HUMAN AND COMPUTER RECOGNITION OF REGIONAL ACCENTS AND ETHNIC GROUPS FROM BRITISH ENGLISH SPEECH

BY

ABUALSOUD HANANI

Supervisors: Prof. Martin Russell and Prof. Michael Carey

A Thesis submitted for the degree of **Doctor of Philosophy**

School of Electronic, Electrical and Computer Engineering

The University of Birmingham

March 2012

Abstract

The paralinguistic information in a speech signal includes clues to the geographical and social background of the speaker. This thesis is concerned with automatic extraction of this information from a short segment of speech. A state-of-the-art Language Identification (ID) system, which is obtained by fusing variant of Gaussian mixture model and support vector machines, is developed and evaluated on the NIST 2003 and 2005 Language Recognition Evaluation (LRE) tasks. This system is applied to the problems of regional accent recognition for British English, and ethnic group recognition within a particular accent. We compare the results with human performance and, for accent recognition, the 'text dependent' ACCDIST accent recognition measure. For the fourteen regional accents of British English in the ABI-1 corpus (good quality read speech), our language ID system achieves a recognition accuracy of 86.4%, compared with 95.18% for our best ACCDIST-based system and 58.24% for human listeners. The "Voices across Birmingham" corpus contains significant amounts of telephone conversational speech for the two largest ethnic groups in the city of Birmingham (UK), namely the 'Asian' and 'White' communities. Our language ID system distinguishes between these two groups with an accuracy of 94.3% compared with 90.24% for human listeners. Although direct comparison is difficult, it seems that our language ID system performs much better on the standard twelve class NIST 2003 Language Recognition Evaluation task or the two class ethnic group recognition task than on the fourteen class regional accent recognition task. We conclude that automatic accent recognition is a challenging task for speech technology, and that the use of natural conversational speech may be advantageous for these types of paralinguistic task.

One issue with conventional approaches to language ID that use high-order Gaussian Mixture Models (GMMs) and high-dimensional feature vectors is the amount of computing power that they require. Currently, multi-core Graphics Processing Units (GPUs) provide a possible solution at very little cost. In this thesis we also explore the application of GPUs to speech signal and pattern processing, using language ID as a vehicle to demonstrate their benefits. Realisation of the full potential of GPUs requires both effective coding of predetermined algorithms, and, in cases where there is a choice, selection of the algorithm or technique for a specific function that is most able to exploit the properties of the GPU. We demonstrate these principles using the NIST LRE 2003 task, which involves processing over 600 hours of speech. We focus on two parts of the system, namely the acoustic classifier, which is based on a 2048 component GMM, and the acoustic feature extraction process. In the case of the latter we compare a conventional FFT-based analysis with an FIR filter bank, both in terms of their ability to exploit the GPU architecture and language ID performance. With no increase in error rate our GPU based system, with an FIR-based front-end, completes the full NIST LRE 2003 task in 16 hours, compared with 180 hours for the more conventional FFT-based system on a standard CPU (a speed up factor of more than 11).

***Keywords*:** Language Identification, Gaussian Mixture Model, Support Vector Machine, GMM Tokenization, Graphical Processing Units, Speech Spectral Analysis, Feature Extraction, Accent Identification, British English, Ethnic Groups Identification.

Acknowledgements

I would like to express my sincere gratitude towards my supervisors Professor Martin Russell and Professor Michael Carey for their unfailing support, guidance, encouragement and advice. This thesis would not have been possible without them. Inspired by their deep understanding and knowledge, our discussions always gave me invaluable insight into problems and techniques. They gave me great suggestions about possible methods and refining ideas. Also I wish to thank Mrs Mary Winkles, the EECE postgraduate secretary, Dr Sridhar Pammu, Computer Officer for DSVP group, for their support and help in many occasions during my study. Also, I wish to thank all of my friends; Saeid Safavi, Gheida Shahrour, Masoud, Mohammad, Maryam and Mahdi in EECE and all people who I met either inside or outside the university of Birmingham for their support. I also wish to thank my examiners; Dr Peter Jancovic and Prof Renals for their detailed review, constructive criticism and advice during the preparation of this thesis. My acknowledgment also goes to the UK Overseas Research Student (ORS) scholarship and to the department of Electronic, Electrical and Computer Engineering at the University of Birmingham for the financial support. Last but not least, a special thanks to my mother and my father for their patience, prayer, support and encouragement. This thesis is especially dedicated to my wonderful wife, Safa and my children Raneem and Ahmad for their love, encouragement and support during the whole period of my study. I also would like to thank my brothers and my sisters and all of my friends and relatives in my village Beit Furik for their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter is an overview of the research presented in this thesis, highlighting the research questions that will be addressed. A brief literature review with references to relevant existing work is also presented.

1.1 Research questions

The title of this thesis is human and computer recognition of regional accents and ethnic groups of British English speech. From this title, many research questions can be formed. The key questions of this research are:

1. It is known in language identification (ID) that the most successful approaches are those which exploit differences between the distributions of sounds in different languages, and those which exploit language differences in the sequences in which these sounds occur. Based on this, to what extent can differences between the distributions and sequences of sounds be used for accent and ethnic group identification? In other words, how well does a state-of-the-art language ID system performs for regional accent and ethnic group identification?

2. How well do these approaches from language ID perform compared with human listeners?

3. In the case of of accent identification, specialist techniques have been described in the literature. How well do language ID techniques perform compared with these specialist techniques?

4. Like many other speech processing applications, language ID requires intensive computation. Multi-core Graphics Processing Units (GPUs) have been successfully used to speedup computation in a number of applications. To what extent can computation in language ID be accelerated using GPUs?

5. Turning to front-end signal processing, to what extent can the best language ID performance, in terms of both accuracy and use of computer resources, be achieved not only by optimizing the mapping of spectral analysis algorithms onto the GPU architecture, but also by choosing a spectral analysis technique where a high degree of optimization can be realized?

1.2 Implementation plan

In order to answer all of the above research questions, we developed and implemented a state-of-the-art language recognition system on the NIST 2003 and 2005 language recognition evaluations and we applied it to accent and ethnic group recognition. Two human perceptual experiments for accent and ethnic group recognition were also conducted to compare the performance of automatic systems with human listeners.

For investigating the effectiveness of using GPUs for language ID, the most computationally expensive components are optimized and implemented for running on the GPU. In addition, different spectral analysis algorithms are optimized and implemented for language recognition using GPUs.

1.3 Language, dialect and accents in speech and language technology

A speech signal contains a wealth of information over and above its linguistic content, including clues to the geographical, social and ethnic background of the speaker. In the case of British English, most native listeners would be more or less aware of the speaker's regional accent, and a listener from the same region might also be aware of the speaker's social or geographical 'subgroup' within the region. In the first volume of "Accents of English", Wells defines 'accent of English' as "a pattern of pronunciation used by a speaker for whom English is the native language or, more generally, by the community or social grouping to which he or she belongs" [1]. This is different from 'dialect' which also includes the use of words that are characteristic of those regions. So, for example, when a speaker from Yorkshire in the North of England pronounces "bath" with the same vowel quality as "cat" rather than "cart" he or she is exhibiting a Yorkshire (or at least north of England) accent, but use of the word "lug" to mean "ear" or "flag" to mean "paving stone" are examples of Yorkshire dialect [2, 3].

In recent years the topics of 'accent' and 'dialect' recognition have become more common in speech and language technology research. A search of the Interspeech 2010 proceedings for the word 'dialect' returns 74 references, of which 64 refer to some extent to work concerned with variations caused by dialect or accent (this is approximately 8% of the total number of papers presented at Interspeech). Of these, 40% are concerned with speech science, referring to 'dialect' in the contexts of 18 different languages, and 60% with technology. In speech technology the most common references are to dialect as a source of variability in speech recognition, however five papers address the problem of dialect recognition directly. There is some ambiguity in the speech technology literature between the terms 'dialect' and 'accent' and some authors also use the term 'variety' (for example, Koller, Abad, Trancoso and Viana discuss varieties of Portuguese in [4]).

In these terms, most speech technology is concerned with accent. A search for the word 'accent' returns 1771 instances in 117 documents in the Interspeech 2010 proceedings, but of course, accent certainly has more than one meaning in the context of speech and language science.

Automatic accent recognition from speech has a number of potential applications. Accent is a major source of variability for Automatic Speech Recognition (ASR) [5, 6, 7], and recognizing a speaker's accent prior to ASR could enable a system to accommodate this variation more effectively, for example by choosing appropriate acoustic and lexical models. However, even within a 'homogeneous' population of subjects who were born in the same town or city and have lived there all of their lives, and therefore notionally speak with the same regional accent, there are likely to be significant differences. Typically these will include speakers whose accent is close to standard British English and other variations in accented speech associated with different social, geographical or ethnic groups. Therefore a more interesting challenge is to develop a continuous space representation of speakers and accent, such that subjects who are close in this space speak in a similar manner and, from the perspective of automatic speech recognition, can be characterized by similar sets of model parameters.

Coupled with a capability to synthesize regionally accented speech (for example, [8]), automatic accent recognition could also be used to select appropriately accented synthetic speech in the context of an interactive dialogue system.

Much of the existing work on automatic accent recognition takes its lead from Language ID, and as in language ID the different approaches can usefully be partitioned into acoustic methods, which exploit differences between the distributions of sounds in different accents, and 'phonotactic' approaches which exploit accent-dependent differences in the sequences in which these sounds occur [9]. An early example of the latter is Zissman's work [10] on the application of Phone Recognition followed by Language Modelling (PRLM) to accent recognition. The performance of PRLM can be further improved by the use of discriminative methods that focus on phones or phone sequences that are characteristic of an accent [7, 11]. Another technique borrowed from language ID is the use of Gaussian Mixture Models (GMMs) and Support Vector Machines (SVMs) to model the acoustic properties of accented speech, and this has also achieved some success (for example [11]). Other acoustic-based approaches include the use of phone durations and average cepstra [12], phone and word-level Hidden Markov Models (HMMs) [13, 14, 15, 16], and stochastic trajectory models [17].

Some more recent research has exploited specific properties of accents. The approach described in [7] uses the fact that, at least to a first approximation, accents share the same phoneme inventory, but the realization of these phonemes may differ. They report improved performance compared with a conventional utterance level GMM-SVM system by using phoneme-dependent GMMs and creating 'supervectors' at the phoneme level. Huckvale [18] took this a step further with his ACCDIST (Accent Characterization by Comparison of Distances in the Inter-segment Similarity Table) measure, by exploiting the fact that British English accents can be characterized by the similarities and differences between the realizations of vowels in specific words [1, 19]. For example, for our speaker from Yorkshire the distance between the realizations of the vowels in "bath" and "cat" is small, but it is large between those in "bath" and "cart", whereas for a subject with a Southern English accent the opposite is the case. Huckvale reported an accent recognition accuracy of 92.3% on the 14 accents of British English in the ABI-1 corpus [18].

Studies of human accent recognition, and the effects of the linguistic backgrounds of the listeners involved, have been reported for six regional accents of French [20], three regional accents of British English [21], and non-native accented English [13]. The ability of human listeners to distinguish between African American English and Standard American English has also been studied [22, 23].

The comparison of different approaches to accent recognition is difficult because of the absence of standard corpora or evaluation methodologies. Corpora that have been used include various collections of non-native English [13, 14, 16, 17], British and American English [15], different corpora of regional accents of British English [16, 18, 21, 24], five varieties of spoken Arabic [7], and six regional French accents [20].

In this research we apply a state-of-the-art language ID system to extract two types of paralinguistic information from English[1] speech, namely the speaker's regional accent and, in the case of Birmingham accented speech, the ethnic group to which the speaker

[1]Unless otherwise stated, in this paper 'English' refers to 'British English' speech.

belongs. As well as measuring the overall performances of our language ID system, we report the performance of its acoustic and phonotactic subsystems on these tasks. These are compared with human listener performance, and in the case of regional accent recognition, with two systems based on Huckvale's ACCDIST measure [18].

1.4 Computational load

Automatic language ID is a very expensive computational task. In language ID, the feature vectors that represent the acoustic speech signal, are considered to be independent of each other. This means that the elements of the sequence of speech frames can be processed in parallel. For these types of tasks, parallel computation based processors outperform modern central process units (CPU). This encourages the use of GPUs to speedup the computation. Hence, one of the goals of this thesis is to investigate the implementation of language ID systems on a GPU.

GPUs provide a huge amount of processing power for very little cost. This alone justifies the growing interest in their use in a wide range of scientific and engineering disciplines. For many years, engineers have struggled against the limitations imposed by the cost and availability of powerful processors, and this has discouraged the exploration of algorithms of high complexity. There are of course exceptions. For example Atal and Schroeder's [25] first code excited linear prediction (CELP) algorithm ran at about 100 times real-time on a Cray 1, a processor capable of up to 250 MFlops. Due to simplification of the algorithm and the availability of more powerful processors CELP has become the standard coding technique used in mobile phones, with several billion instantiations. Optimal performance was achieved by handcrafting algorithms in machine code on processors designed to execute digital signal processing algorithms, such as the Texas Instruments TMS 320 family.

Over the last five years the situation has changed as PC-based computer games players have demanded and obtained increasingly more powerful processors to enhance the realism of their games. This has led to the development of multiprocessor graphics cards with arrays of processors, each capable of performing more than 1000 Mflops. Over the last 30 years the price of one megaflop of computing power has fallen from $36,000 to about 0.035c, a factor of 10^8. It has been suggested that this is the third wave of change in the semiconductor/computer industry following the first wave, the integrated microprocessor, and the second, the digital signal processor [26].

GPU capabilities have been explored in different areas which need intensive computations and real time operation, such as computer vision and signal processing, including speech processing technology. In computer vision Fung [27] uses the GPU to speed up signal processing algorithms such as low-pass filtering, blurring and downsampling. By using a GPU they achieved factor of 3.5 (or 3.5$\times$) speedup over the CPU implementation. In [28], Erra implemented a fractal image compression on an NVidia GeForces FX 6800 GPU and achieved a 280$\times$ speedup over the CPU implementation. Many other applications and algorithms implemented on the GPU, such as geometric computations, collision detection, and particle tracking, are discussed in [29]. These results motivate the use of GPUs to speedup the computation in our language ID system.

In this research we investigate the potential impact of the GPU on language ID by examining their impact on the problem of probability calculation and speech-spectrum estimation in the context of automatic language ID. Language ID systems, typically comprise several algorithmic units of which two normally absorb the majority of the computer's resources. These are spectral analysis and the calculation of the conditional probability of the input speech given some pre-trained acoustic model, typically a GMM. Several papers [30, 31, 32, 33] have already considered the use of a GPU to calculate these conditional probabilities. This calculation is a weighted sum of Gaussian Probability Density Function (PDF) calculations, and while rearrangement of the algebra is possible the algorithm used is fixed. Training these systems usually involves an iterative technique such as the Expectation Maximization (EM) algorithm, which dominates the use of the computer resources.

The optimizations for this calculation and the spectral analysis calculation are discussed at length in chapter 9.

1.5 Organization of the thesis

The remainder of the thesis is organized as follows:

Chapter 2 provides an overview of language ID systems. **Chapter 3** describes in detail the process of extracting cepstral features for language ID, with a focus on four different algorithms for spectral analysis. Four feature normalization techniques, which could help to remove unwanted distortion while keeping important speech information, are also discussed in chapter 3.

Chapter 4 and **5** describe the theory and implementation of the main components of our phonotactic and acoustic language ID systems. Four score normalization techniques are also described in chapter 5. This chapter also describes a powerful technique to compensate for inter-session variability within a single language.

Chapter 6 discusses popular methods of fusing the outputs of several different language ID systems to improve performance. It also presents our complete language ID system which is the fusion of a number of phonotactic and acoustic sub-systems.

Chapter 7 describes all of the the speech data used in this research. It describes the NIST and CallFriend corpora which are used in all of the language ID experiments, the Accents of the British Isles (ABI) corpus, which is used in all of our accent recognition experiments, and finally the "Voices across Birmingham" (VaB) corpus which is used for the ethnic group ID.

Chapter 8 describes the development and evaluation of our language ID systems on the NIST 2003 and 2005 language recognition evaluations task. In this chapter, phone recognizers trained on different languages are used to build phonotactic language ID systems. Different combination of the acoustic features, and feature and score normalization techniques are examined experimentally to find the best configuration for the language ID task. In addition, inter-session compensation techniques are applied to both GMM and SVM acoustic systems. The results in this chapter set up a baseline system for the work described in this thesis and demonstrate that it achieves a level of performance comparable with the state-of-the-art [34, 35].

Chapter 9 describes the development and optimization of various algorithms to speed up computation for language ID. The probability computation is optimized for running on a CPU and on a GPU using Matlab, GPUmat and the NVidia CUDA toolkits. This chapter also investigates alternative spectral analysis algorithms for language ID and the optimization process for mapping each algorithm onto the GPU. Four algorithms, namely FFT, FIR, IIR and LPC, are investigated in terms of computational speed and language

ID performance.

Chapter 10 investigates the effectiveness of standard state-of-the-art techniques from language ID for accent recognition on fourteen regional accents of British English. The performances of these systems are compared with variations of the text-dependent ACCDIST-based method proposed by Huckvale [18] and also with the performance of human listeners.

Chapter 11 investigates whether techniques used for language and accent ID are able to distinguish between speakers from different social (or ethnic) groups within a single regional accent. Specifically, 'White' and second generation 'Asian' groups in the city of Birmingham. These groups are well represented in the "Voices across Birmingham" corpus of recordings of telephone conversational speech between individuals in the city. In this chapter we apply the techniques developed in the previous chapters for language and accent ID to ethnic group ID. Obviously we are not recognizing ethnicity, but the pattern of speech for the two ethnic groups. The performance of the automatic systems is compared with human listeners on this task.

Chapter 12 highlights the achievements of the thesis with respect to the language, accent and ethnic groups ID from speech. It also discusses opportunities and directions for future work.

1.6 Major contribution

The research described in this thesis provides original contributions to the field of automatic recognition of speaker language, regional accent, and ethnic group within a single accent, based on a short segment of speech. The major contributions can be summarised as follows:

- First investigation of the effectiveness of the standard language ID techniques for identifying regional accents of British English, and for identifying the two main ethnic groups within the Birmingham accent in the “voices across Birmingham” corpus. Refer to the publications **(III)**, **(II)** and **(IV)** in the next section.

- Extension of Huckvale’s ACCDIST system to use vowel tri-phones (phone-vowel-phone) instead of a complete word context, which removes the need for subjects’ speech to correspond to the same text. Improved accent recognition performance by modeling ACCDIST speaker distance tables with SVMs. Refer to the publications **(III)** and **(IV)** in the next section.

- Implementation of GMM conditional probability computation and inter-session compensation for language ID with multi-core GPUs. This allows new techniques to be tried (such as Multi-Language Model (MLM)) which would otherwise not be possible due to their high computational load. Refer to the publication **(V)** in the next section.

- Comparison of four spectral analysis algorithms for language ID, in terms of compatibility with GPU, and language ID performance. A filter bank with Finite Impulse Response (FIR) filters has been found to outperform the classical Fourier transform-based front-end in terms of computational speed on the GPUs and with no degradation of language ID performance. Refer to the publication **(V)** in the next section.

- The first use of discriminative weighting with a GMM tokenization technique for language, accent and ethnic groups identification. The proposal of the Multi-Language Model (MLM) for building an efficient language-independent GMM tokenizer. Refer to the publication **(I)** in the next section.

1.7 List of publications

(I) **A. Hanani**, M. Carey and M. Russell, "Improved Language Recognition using Mixture Components Statistics, Proc. Interspeech 2010, Tokyo, Japan, 741–744, 2010.

(II) **A. Hanani**, M. Russell, and M. Carey, "Speech-Based Identification of Social Groups in a Single Accent of British English by Humans and Computers", Proc. IEEE ICASSP 2011: Prague, 4876–4879, 2011.

(III) **A. Hanani**, M. Russell, and M. Carey, "Computer and Human Recognition of Regional Accents of British English", Proc. Interspeech 2011, Florence, Italy, 729–732, 2011.

(IV) **A. Hanani**, M. Russell, and M. Carey, "Human and Computer Recognition of Regional Accents and Ethnic Groups from British English Speech", accepted subject to modifications for a Special Issue of Computer Speech and Language journal on Natural Paralinguistics, August 2011.

(V) **A. Hanani**, M. Carey and M. Russell, "Language Recognition Using Multi-core Processors", accepted subject to modifications for Computer Speech and Language journal, October 2011.

(VI) M. Russell, **A. Hanani,** and M. Carey, “Some Thoughts on Language, Dialect and Accents in Speech and Language Technology”, IEEE Speech and Language Technical Committee (SLTC) Newsletter, February 2011.

(VII) M. J. Carey, **A. Hanani** and M. J. Russell ,“The Next Big Thing in Computing”, IEEE Speech and Language Technical Committee (SLTC) Newsletter, July 2011.

1.8 Summary

In this chapter,the research focus of the thesis was introduced with a list of research questions and the implementation plan of the research. A brief literature review covering the main topics of the thesis was presented.

The organization and the major contributions of the thesis were also outlined. The papers which are already published and and which will be published were also listed.

Chapter 2

Background

Spoken language ID, or language recognition, is a technique which recognizes the language of speaker from his or her speech. Although the target of the language recognition task is different from speech and speaker recognition, which aims to recognize the linguistic contents of a speech signal and the speaker identity respectively, many of the techniques are similar. In this chapter some background knowledge on language ID will be introduced, including the most common and successful approaches to language ID.

2.1 Language Identification

The language ID problem can be viewed as a language recognition or a language detection (or verification) task. The objective of language recognition is to identify the language being spoken from a short sample of speech by an unknown speaker, whereas the task of language detection is to decide if the claimed language is the true language from a short speech sample. Therefore, the evaluation metric for the two tasks is different. The percentage accuracy, the percentage of total utterances correctly classified, is usually used as a performance measure for a language recognition system. The Detection Error Trade-off (DET) curve [36], Receiver Operation Characteristic (ROC) [37] and the Equal

Error Rate (EER), where miss and false alarm probabilities are equal, are usually used as a performance measure for language detection (or verification). In the context of this thesis, language ID is done by machine, therefore it is automatic. Throughout this thesis, the two terms language ID and language recognition carry the same meaning and unless otherwise stated they refer to language detection.

As for speech recognition, humans are more accurate than machines in recognizing the languages that they know (or speak) but much less accurate with languages they do not know, or are less familiar with [38, 39]. In general, there are a variety of cues that humans and machines can use to distinguish between different languages [9]. We know that the following characteristics differ from language to language:

- *Phonetics*: a phone is an abstract representation of a basic sound unit in speech. A phoneme in language, is an equivalence class of phones which are not contrastively significant in that language. For example, the phones /b/ and /p/ are different phonemes in English because they distinguish between words like "bet" and "pet", but in Arabic /b/ and /p/ are allophones of the same phoneme. The sets of phones differs from one language to another, but many languages share a common subset of phones. Phone frequencies may differ, i.e., a phone may occur in two languages, but it may be more frequent in one language than in the other.

- *Phonotactics*: In each language there are different rules governing the allowable sequences of phones. Therefore the statistics of phoneme combination or sequence can also be used to distinguish languages. Some combinations that occur frequently in one language are illegal in another.

- *Morphology*: Each language has its own vocabulary, and its own manner of forming words.

- *Syntax*: The sentence patterns are different among different languages.

- *Prosody*: duration characteristics, pitch and stress patterns are different from one language to another.

Most current automatic language identification systems take advantage of one or more of these characteristics in discriminating between languages.

The applications of language identification mainly fall into three fields. Its use by national security services for monitoring communications is probably the most dominant. The main focus is to route the call to an appropriate large vocabulary speech recognition (LVCSR) system, keyword spotting (KWS) system, or an appropriate human listener. Language ID is also used in the domains of Information Retrieval (IR), automatic translation services and education services. Alternatively, language ID might be used to route an incoming telephone calls to human switchboard operator fluent who is in the detected language.

2.1.1 Structure of the language ID system

There are many approaches to language ID. In general, the structure of the most common approaches (figure 2.1) can be described as follows:

- **Front-end signal processing**– The main aim of front-end processing is to extract from a speech signal a stream of features that can be used to distinguish between languages. These features should capture the salient aspects of the speech signal and be perceptually meaningful, if possible. They also should be robust in the sense that the language ID task should not be affected by the distortions that can appear, due to among other things environmental aspects or the transmission medium.

- **Classification**– There are lot of approaches to language recognition. Generally, a state-of-the-art language ID system is obtained by fusing the outputs of a number of subsystems. These subsystems are normally of two types, acoustic and phonotactic. In a phonotactic system the test speech signal is first processed by a phone recognizer, whose output is passed to a set of statistical phone-level language models, one for each language, resulting in a set of candidate language probabilities. Often several such systems are run in parallel and their outputs fused. This is the Parallel Phone Recognition Language Modeling (PPRLM) approach from [9]. A detailed description about the phonotactic language ID systems is found in chapter 4. The acoustic component of a language ID system is based on the assumption that the distributions of acoustic feature vectors in each candidate language are different. Acoustic language ID systems typically use GMMs to characterize these distributions, or SVMs to discriminate between them, or, most likely, a combination of both. More details about the acoustic language ID systems are found in chapter 5.

- **Thresholding and decision** is used for making hard decisions. This is "true" or "false" in the case of detection task, or the hypothesized language in the case of recognition task.

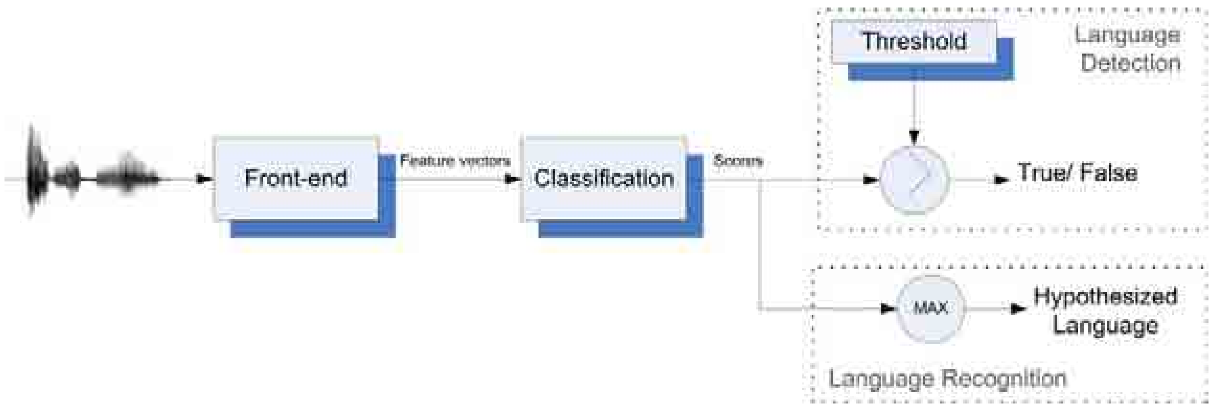


Figure 2.1: General Language ID Structure

2.2 NIST evaluation metric

The National Institute of Standards and Technology (NIST)[1] runs regular Language Recognition Evaluations (LRE), so that systems from different laboratories can be evaluated on common data. The first NIST LRE took place in 1996 and the second evaluation was in 2003. After that NIST conducted evaluations every two years.

The performance of language ID system is evaluated separately for test segments of three durations, 3s, 10s and 30s according to NIST [40, 41] per-language. A standard DET curve [36] is evaluated as a plot of the probability of false alarms against the probability of misses, with the detection threshold as parameter and equal priors for target and non-target languages. The EER is the point where these probabilities are equal. NIST recognizes two overall EERs. The 'pooled' EER is the average of the language-dependent EERs, and the 'average' EER is the weighted average of the language-dependent EERs, where the weights are the language priors.

During this research, DET curves, 'average' EER percentage with 90% confidence interval and (occasionally) the accuracy rate (i.e. percentage of correctly classified tests)

[1]http://www.nist.gov

are used as a measure of classification performance.

2.3 Summary

This chapter presented background knowledge on language ID. The main clues and characteristics that can be used to distinguish between languages are discussed, together with the main existing and potential applications for automatic language ID systems. The general structure of a typical language recognition system was also introduced with a brief explanation of each component. The most popular and successful approaches to language ID, particularly phonotactic and acoustic systems, were also introduced in this chapter. The evaluation metric proposed by NIST for language recognition evaluation was presented.

Chapter 3

Front-end Analysis

Front-end analysis is the first stage in most of speech applications, whereby the input speech signal is converted to a sequence of acoustic feature vectors which contain information necessary for the recognition task. This chapter will present the theory and methods of extracting features vectors which have been shown empirically to be an effective for language ID. Since language recognition and speech recognition are two different tasks, the front-end of the two tasks is also different. With more focus on the acoustic features for acoustic language recognition, the front-end of state-of-the-art language and speech recognition systems will be presented in this chapter. The front-end for speech recognition will be used in building a phone recognizer, the basic component in the phonotactic language ID systems.

3.1 Overview

Speech signals have time varying spectra occupying a 10 kHz bandwidth. In telephony, the speech is low-pass filtered to restrict the bandwidth to 4 kHz and sampled at 8kHz. The spectrum is assumed to be quasi-stationary over 10 to 20 ms periods. The power envelope of the speech signal occupies a band of frequencies from approximately 0 Hz

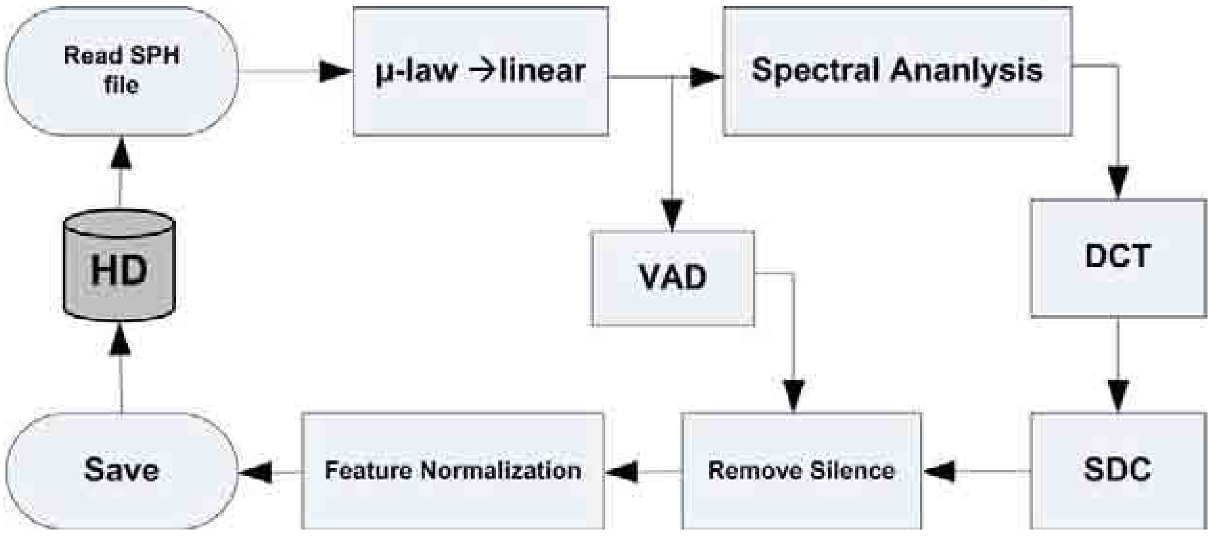


Figure 3.1: Block diagram of language ID feature extraction process

to 50 Hz. Hence information can be extracted by spectral analysis at a frame rate of between 50 and 100 Hz.

The basic process of feature extraction in language ID is shown in figure 3.1. The following sections give details about each block in figure 3.1.

3.2 μ-law $\rightarrow$ linear

In the applications considered in this thesis, the speech data are usually stored in NIST SPHERE format (SPH), 16-bit μ-law PCM [42]. The first step in feature extraction is to convert the μ-law PCM to linear PCM encoding. The speech waveform is then partitioned into a sequence of overlapping frames defined by a set window size, typically 20-25ms. The window size is chosen based on the understanding of the human speech production system, where the vocal tract is considered to be relatively constant for the duration of the window resulting in a relatively constant frequency spectrum. In addition, the length of the window should be short enough to give the required time resolution and also should be long enough to provide adequate frequency resolution. The overlap between windows is typically set to half the window size, e.g. a 10ms interval between 20ms windows to produce 100 feature vectors per second.

3.2.1 Spectral analysis

In any pattern recognition task, the objective of feature extraction is to convert the raw patterns into feature vectors which emphasise the information that is relevant for classification and attenuate that which is not. In the case of speech, knowledge of the human speech production system and the perceptual auditory system suggest that a key stage in this process is spectral analysis.

The purpose of the 'spectral analysis' stage in figure 3.1 is to convert the speech waveform into a sequence of feature vectors, whose entries represent the energies in particular frequency bands. It has been shown that performance improves if the the frequency scale is linear up to about 1 kHz and logarithmic from 1 kHz to the upper frequency limit, with a commensurate increase in the frequency band width to match the critical bands of the human auditory system. This is referred to as the Mel scale [43].

This can be achieved in two ways. Either the spectrum is estimated (for example, using Discrete Fourier Transform (DFT) or Linear Prediction (LP)) and the frequency axis is quantized, with quantization bins defined according to a mel scale, or the speech is passed through a bank of filters whose centre frequencies and bandwidths are defined according to mel scale.

Thus there are several algorithms to perform these analyses, but in this thesis we chose the following algorithms:

1. Infinite Impulse Response (IIR) Filter Bank,
2. Finite Impulse Response (FIR) Filter Bank,
3. Discrete (Fast) Fourier Transform (DFT),
4. Linear Prediction.

Each of the above algorithms outputs a vector from which the Mel scale power spectrum of the speech can be estimated.

3.2.1.1 The IIR filter bank

The individual analogue filters in the original filter banks had Butterworth or Bessel responses [44]. The equivalent digital form is usually realised as a recursive filter, which can be designed using the bilinear transform on the coefficients of the analogue filter. Each filter is typically a fourth order Bessel filter, realised using the following recurrence relation

$$Y(n) = \sum_{i=1}^{I} a_i Y(n-i) + \sum_{j=0}^{J} b_j X(n-j), \tag{3.1}$$

where X and Y are the input and filtered signal, respectively, and $a_1, \ldots, a_I$ and $b_j, \ldots, b_J$ are the filter coefficients. The stages in the processing for a single channel are shown in figure 3.5(b). The recursive nature of the algorithm causes restrictions on the computation, since each calculation of the output cannot complete until all its predecessors have been calculated.

The filtering operation is followed by inputting the absolute value of the samples into a two-stage single pole filter to produce a smoothed estimate of the instantaneous channel power. The output of this filter is then decimated by sub-sampling from the

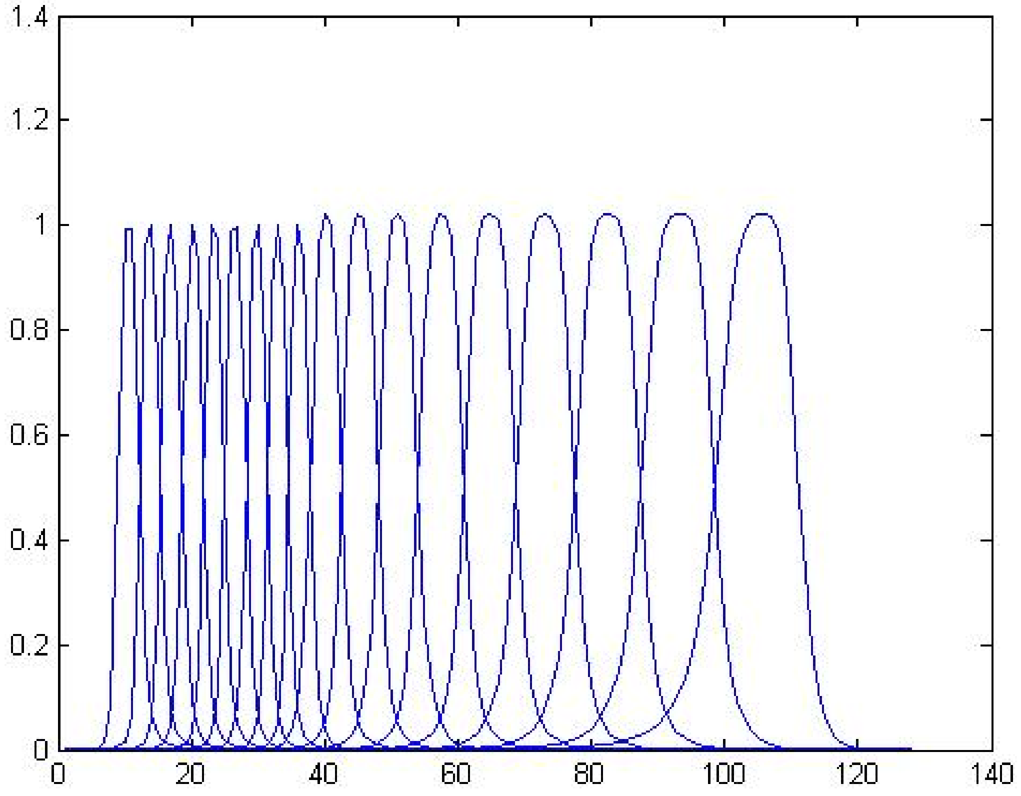


Figure 3.2: Mel-scaled IIR filter bank.

speech sampling rate to the frame rate of 100 frames per second. A bank of IIR filters (channels) is shown in figure 3.2.

3.2.1.2 FIR filter bank

Unlike the recursive filter described above, the outputs of a non-recursive, FIR can each be computed independently of the others. Figure 3.5(d) shows a single channel of the FIR filter bank. It comprises two linear phase non-recursive filters, the first of which has an In-phase response (I) while the second has a Quadrature response (Q). Each filter has a maximum impulse response of length w. The output of each filter is computed in w

window and the sum of the squares of the outputs of the in-phase and quadrature filters are computed. To give a smoothed energy estimate, successive pairs of outputs are added together. If a single filter only is used the output is sensitive to the relative phase of the input and the filter.

FIR filters are normally equiripple filters designed using an optimization technique, the Remez exchange algorithm [45]. However, while this works extremely well when the filter required has a pass band which is a significant fraction of the sampling frequency, it does not produce good results when the filter pass band is relatively narrow, as is the case in this application. A window-based technique was therefore used to design the filters. Samples of a pair of sinusoids with a frequency equal to the filter centre frequency are multiplied by a window function to give the I and Q sets of filter coefficients, $h_i(n)$ and $h_q(n)$ as follows:

$$p(n) = \cos(kn) \tag{3.2}$$

$$q(n) = \sin(kn) \tag{3.3}$$

where $k = \frac{2\pi f_c}{f_s}$, f_c is the centre frequency of the filter, f_s is the sampling frequency and $n = \frac{N}{2} + 1, \ldots, N$, where N is the filter order. For symmetry $p(n) = p(N - n + 1)$ and antisymmetry, $q(n) = -q(N - n + 1), n = 1, \ldots, \frac{N}{2}$. then

$$h_i(n) = p(n)w(n) \tag{3.4}$$

$$h_q(n) = q(n)w(n) \tag{3.5}$$

Gaussian [46], Hamming [47] and Kaiser [48] windows were used in the filter design but no significant performance differences were apparent. Consequently a Hamming window was retained for subsequent experiments. Figure 3.3 shows the frequency response of FIR filter bank consisting of seventeen channels.

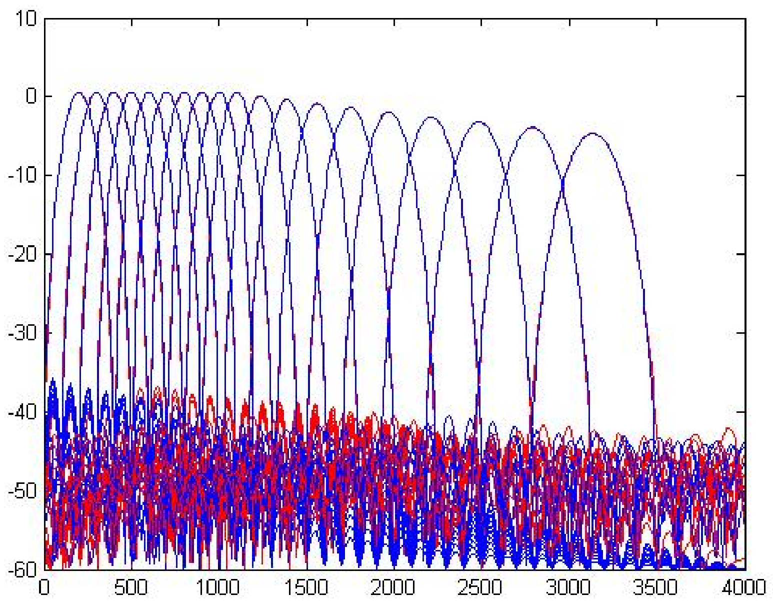


Figure 3.3: Mel-scaled FIR filter bank.

3.2.1.3 DFT filter bank

An alternative way of estimating the power spectral density of the speech signal is to use a DFT. This has become the standard approach because it has a highly optimized implementation, namely the Fast Fourier Transfer (FFT)[49]. The FFT filter bank is illustrated in figure 3.5(c).

The log amplitude of the FFT is then computed to produce the log power spectrum of each speech frame. Davis and Mermelstein [43] implemented a Mel scale filter bank using the spectral power estimates over a limited part of the frequency range to approximate to the power outputs of one of the filters in the filter bank. Each filter output is calculated as a triangular weighted sum of the spectral power coefficients, where the triangular

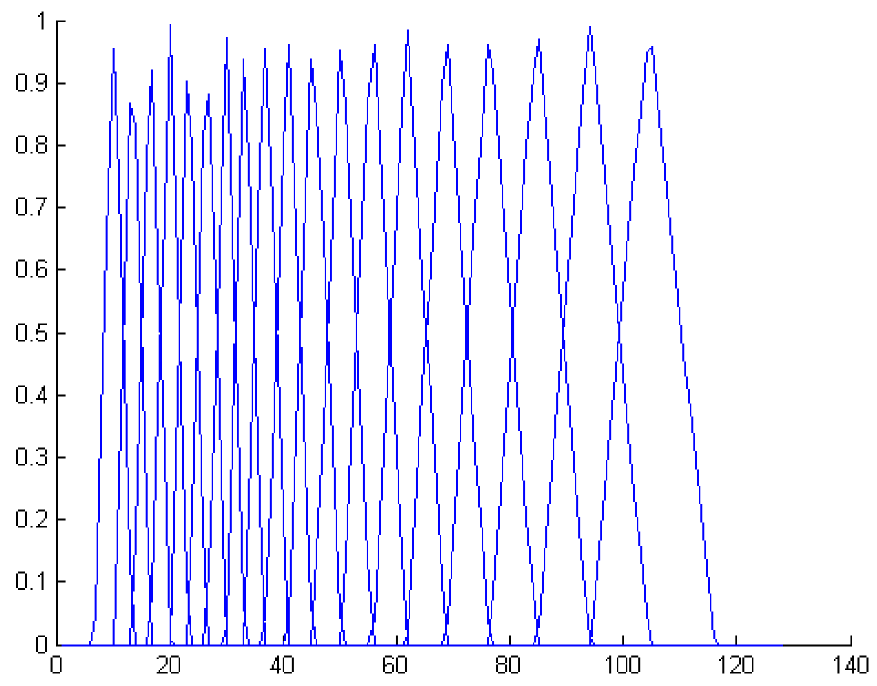


Figure 3.4: Mel-scaled DFT filter bank with 19 triangular filters.

windows are equally spaced on a Mel frequency scale [43], as shown in figure 3.4. The i^{th} Mel frequency spectral power coefficient, m_i, of the i^{th} window, win_i, with spectral power s is calculated as follow:

$$m_i = \sum_f win_i(f)s(f) \tag{3.6}$$

3.2.1.4 Linear prediction

Linear Prediction (LP) is another way to estimate the spectral power coefficients form speech signal. Given a speech samples at time n , $S(n)$ can be modeled as a linear combination of the past p speech samples, such that:

$$S(n) \approx \hat{S}(n) = \sum_{k=1}^{p} a_p(k) S(n-p+k-1), \tag{3.7}$$

where, $\hat{S}(n)$ is the estimate or prediction of $S(n)$, and $a_p = a_p(1), \ldots, a_p(p)$ are unknown LP coefficients. The Levinson-Durbin algorithm [50] is used to compute the prediction coefficients, a_p, for LP analysis (figure 3.5(e)) of the autocorrelation sequence of speech samples. The Levinson-Durbin algorithm provides solution to the linear equations through a recursive procedure that exploits the symmetry property. A cepstral representation is derived from the linear prediction coefficients [51] of every speech frame:

$$c(k) = a_p(k) - \sum_{i=1}^{k} -1(1-\frac{1}{k}) a_p(i) c(k-i), \quad k = 1, \ldots, p, \tag{3.8}$$

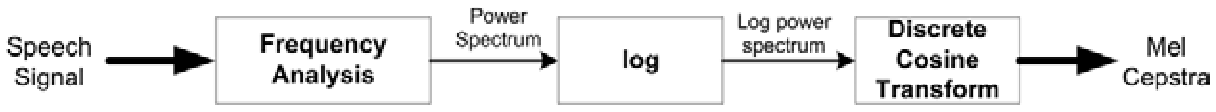
where $c(k)$ is the k^{th} cepstral coefficient, but in the linear frequency scale.

An alternative to the cepstral representation above is Perceptual Linear Prediction (PLP) [52]. PLP is successfully incorporates a non-linear frequency scale, which is very similar to MFCC analysis.

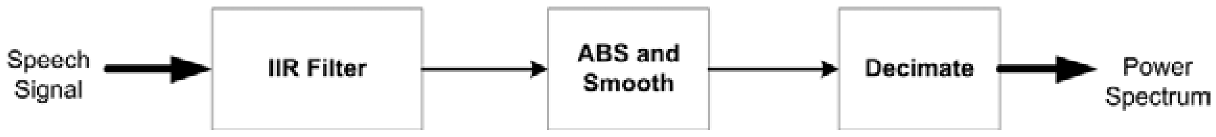
Although it is not as commonly used in language recognitions as MFCCs, PLP is frequently used in state-of-the-art speech recognition systems, therefore, it is used in building our phone recognizer for the phonotactic language ID systems which will be described later in chapter 10.

3.3 Voice activity detection (VAD)

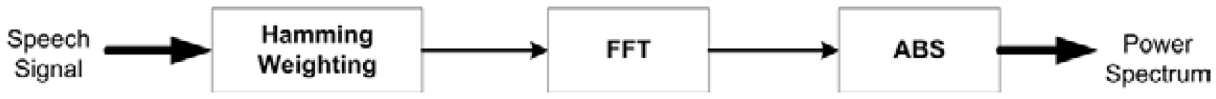
A speech recording contains many non-speech (silence) periods which usually include environmental and channel noise only. Detecting these non-speech periods and excluding them from some parts of speech processing (e.g. language ID processing) improves the



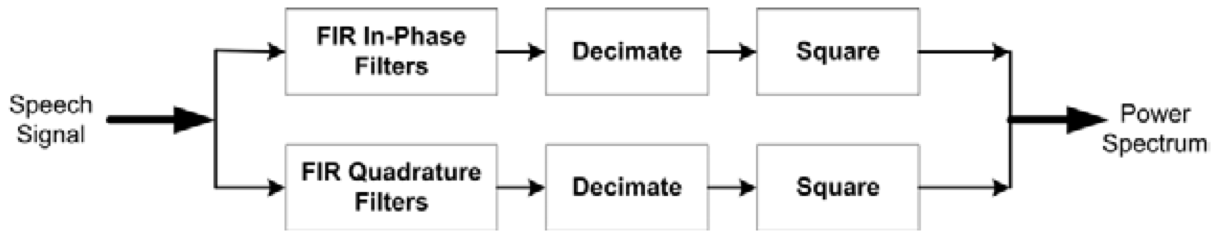
(a) The stages of Spectral Analysis



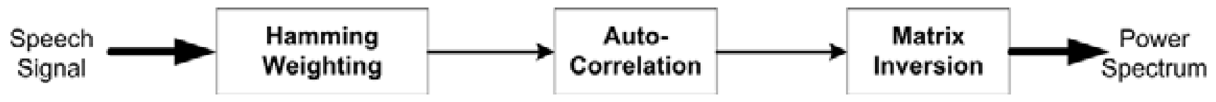
(b) IIR Filter Bank



(c) FFT Filter Bank



(d) FIR Filter Bank



(e) Linear Prediction

Figure 3.5: Schematics Diagrams of a) The stages of spectral analysis, b) IIR Filter Bank, c) FFT Filter Bank , d) FIR Filter Bank and e) Linear Prediction.

recognition performance and reduces the computation overhead. There are two ways to exclude the non-speech periods. One way is by determining the speech/non-speech periods in the speech waveform, removing non-speech periods and then re-concatenating the speech periods to get a ‘speech-only’ waveform prior to the front-end processing. The second way is by labeling the feature vectors constructed from the whole speech waveform (including non-speech) as speech or non-speech and discarding the non-speech vectors at the end of the front-end processing. The later way is essential when calculating delta and shifted-delta cepstra which will be described in section 3.5, whereas, the earlier way has the advantage of reducing front-end computation load. There are many algorithms for voice (or speech) activity detection including those which depend on an energy threshold [53],

$$threshold = \frac{1}{\alpha \times N} \sum_{i=1}^{N} (E_i + \beta \times E_{min}) \tag{3.9}$$

where,

$$E_{min} = min_{i=1}^{N} E_i.$$

E_i is the energy of the i^{th} frame, N is the total number of frames in a given utterance, and α and β are constants estimated experimentally. The energy of each frame is then compared with the energy threshold which is calculated for each utterance. Frames with energy below the threshold are labeled as non-speech and frames with energy above than the threshold are labeled as speech frames.

The alternative to energy based VAD is to use pitch [54] or Zero-Crossing Rate (ZCR) [55]. The ZCR is defined by the average number of times the speech signal crosses the zero line, therefore, it can be calculated by the number of times the speech signal changes its polarity, as shown below:

$$ZCR = \frac{1}{N} \sum_{n=0}^{N-1} |sgn(s(n)) - sgn(s(n-1))| \tag{3.10}$$

where,

$$sgn(s(n)) = \begin{cases} 1, & s(n) \geq 0 \\ -1, & s(n) \leq 0 \end{cases}$$

The ZCR of each frame is compared with thresholds tuned empirically, and frames with very high or very low zero-crossing rate are labeled as non-speech.

3.4 Discrete Cosine Transform (DCT)

Unfortunately for all of the spectral analysis techniques in section 3.2.1, the filter bank outputs are highly correlated, hence the components of the GMMs in the subsequent pattern-processing stage, which will be described in chapter 5, require full covariance matrices to ensure that the probability estimates are accurate. In addition to increasing the computation required to calculate the Gaussian probabilities, more data is required to estimate the model parameters. This can be avoided if the power spectrum estimates are used to calculate a set of cepstral coefficients by means of a DCT.

Coefficients so formed are referred to as Mel Frequency Cepstral Coefficients (MFCCs). It can be assumed that the covariance matrix of the MFCCs is diagonal. The use of MFCC's has two further advantages, the higher MFCC's do not need to be retained since discarding them reduces the pitch induced variability in the spectra and this in turn reduces the feature length of the MFCC vector and the concomitant computation required.

3.5 Shifted-Delta Cepstra (SDC)

For most speech classification tasks the MFCCs are augmented with their delta (velocity), Δ, and double-delta (acceleration), $\Delta\Delta$, parameters [56] which provide local information about feature dynamics. For language ID it is shown in [57] that improved performance can be achieved by incorporating broader temporal information using SDC.

SDC coefficients are obtained by concatenating the delta cepstra computed across multiple frames of speech, as shown in figure 3.6. The SDC features are specified by a set of four parameters, namely N-d-P-k, where N is the number of cepstral coefficients computed at each frame, d represents the time advance and delay for the delta computation, k is the number of blocks whose delta coefficients are concatenated to form the final feature vector, and P is the time shift between consecutive blocks. Accordingly, the SDC coefficients for a cepstral frame i at time t, are computed as follow:

$$\Delta^{SDC} c_n(t,i) = c_n(t+iP+d) - c_n(t+iP-d) \tag{3.11}$$

$$n = 0 \cdots N-1, i = 0 \cdots k-1,$$

where, n is the n^{th} cepstral coefficient and $i = 0 \cdots k-1$ is the SDC block number. The final SDC feature vector is obtained by concatenation of k blocks of N parameters. For example, setting N-d-P-k to 7-1-3-7 [57, 58] results in a sequence of 49-dimensional SDC features. A block diagram in figure 3.6 illustrates the process of computing the SDC coefficients.

SDC coefficients are then concatenated with the static cepstral coefficients.

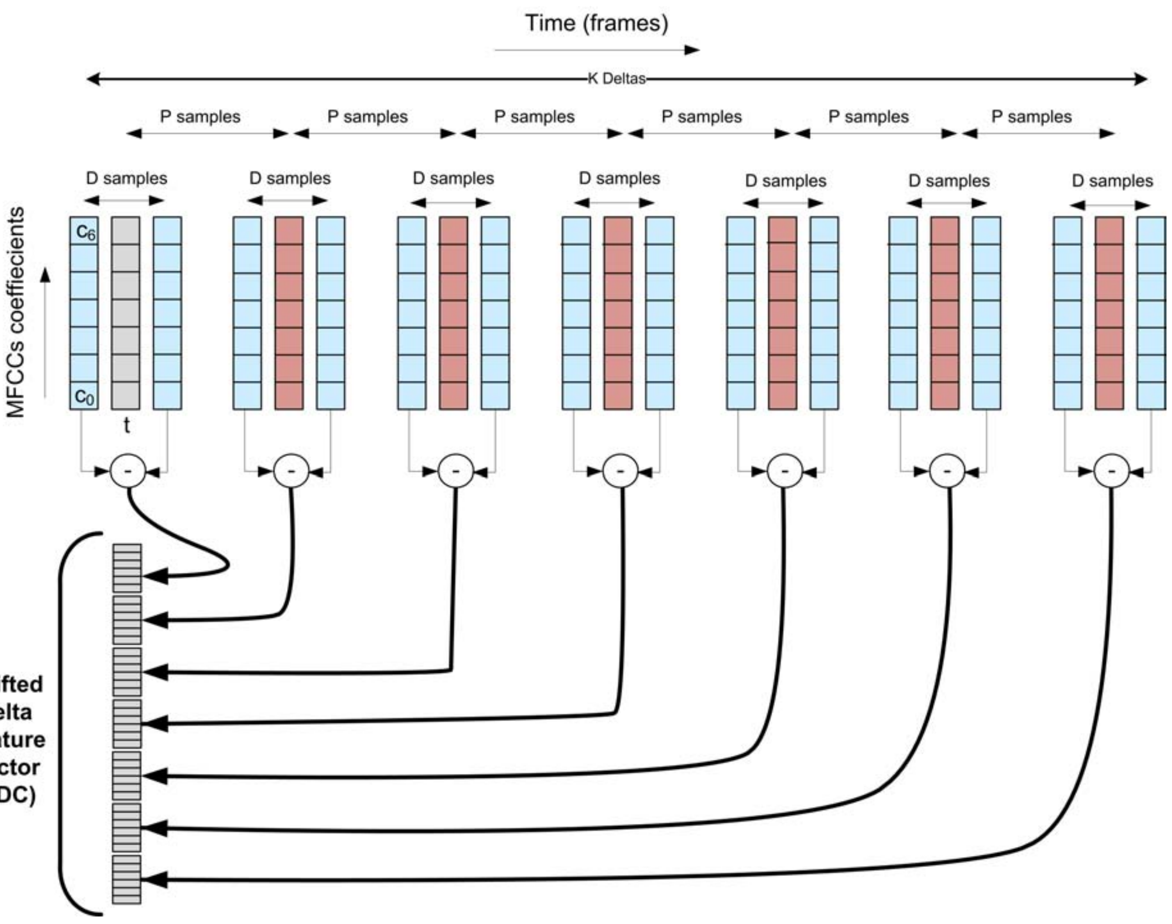


Figure 3.6: SDC coefficients (7-1-3-7) calculation

3.6 Removes Silence

The frame labels, speech or silence, which are generated by VAD are used to discard the silence frames, as they do not carry any useful information.

3.7 Feature normalization

When the speech is collected from the telephone or cellular networks, it may contain noise and distortion due to channel effects and interference. The noise and channel effects have a great impact on all speech technology fields, including language recognition. To reduce the effect of noise and channel from the feature vectors generated from the front-end processing, a number of techniques are typically applied to the raw features before classification. The most commonly used techniques include Cepstral Mean Normalization (CMN) [59], *RASTA* filtering [60], Mean and Variance Normalization (MVN) [61] and Feature Warping (Gaussianization) [62].

1. **Cepstral Mean Normalization (CMN)**: If we assume that most channel distortions are stationary (for example that caused by different microphones, telephone handsets and audio channels), or at least slowly time-varying, then the effect of the channel appears as convolutive noise in the time domain and hence becomes an additive constant in the log cepstral domain. Hence, subtracting the mean of each cepstral coefficient over the whole utterance removes the channel induced offset and any other stationary speech components. CMN can also be applied to variable lengths of speech to remove the varying channel effects within an utterance, such as change of microphone positions.

2. ***RASTA* filtering**: *RASTA* relies on the fact that the rate of change of non-linguistic components in speech often lies outside of the typical rate of change of the vocal tract shape. So, *RASTA* suppresses the spectral components that change more slowly or quickly than the typical range of change of speech. *RASTA* is very similar to the CMN for stationary noise, but in addition it filters noise in the high frequency range (i.e. smooths the feature waveform). *RASTA* can be thought of as a band-pass filter, as shown in figure 3.7 and given in equation 3.12, which filters

out the very low frequency, including DC offset, and the very high frequency.

$$H(z) = 0.1 \times \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{z^{-4}(1 - 0.98z^{-1})} \tag{3.12}$$

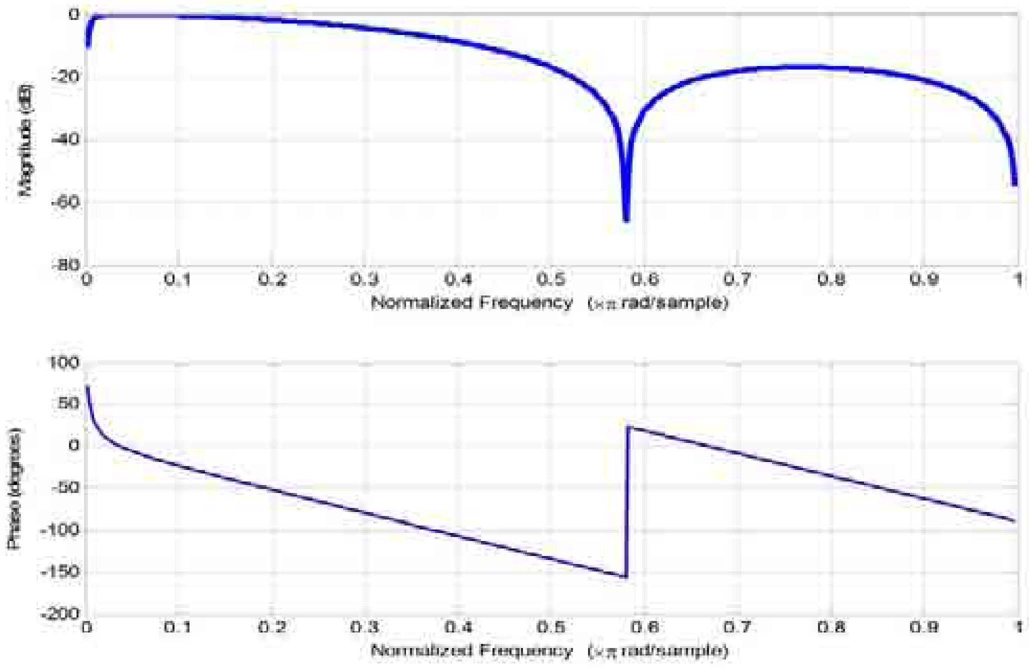


Figure 3.7: Frequency Response of *RASTA* bandpass filter

3. **Mean and variance normalization**: Mean and Variance Normalization (MVN) [61] is the same as cepstral mean subtraction, but in addition each feature coefficient is normalized by the estimated variance of that feature over the whole utterance. By subtracting the mean and dividing by the variance, the distribution of the features will be zero mean and unit variance. This is to remove different cepstral coefficient distributions due to variable channel distortions.

4. **Feature warping**: Applying CMN on cepstral features is proven to provide a robustness against convultive noise such as linear effects of the channel. In addition,

RASTA filters out the low frequency noise and some of the high frequency noise. However, under additive noise conditions, the cepstral feature estimates degrade significantly. Mapping the raw features to a predetermined distribution, such as the standard normal distribution, appears to be a good way to make the features more robust to different channel and noise effects. This can be done via cumulative distribution function (CDF) matching, which warps a given feature so that its CDF matches a desired distribution [62], e.g. standard normal distribution with zero mean and unit variance, $N(0,1)$. The warping can be viewed as a nonlinear transformation from the original feature, x, to a warped feature, $\hat{x}$. This method is performed separately for each cepstral coefficient and assumes that the features of the MFCC vector are independent. CDF matching is performed over a sliding window, of length N. Only the central frame, the point in the middle, of the window is warped based on CDF matching. The features in a given window of the utterance are sorted in descending order. Suppose the central frame has a rank R (between 1 and N). Its corresponding CDF value is approximated as

$$\Phi = \frac{N + \frac{1}{2} - R}{N}. \tag{3.13}$$

Then the warped value $\hat{x}$ should satisfy

$$\Phi = \int_{z=-\inf}^{\hat{x}} f(z)dz, \tag{3.14}$$

where $f(z)$ is is the probability density function of standard normal distribution. The value of $\hat{x}$ can be quickly found by lookup in a standard normal CDF table. The sliding window is advanced one frame and a new entry is calculated, and so on.

In our experiments, this warping technique was found to be most efficient when applied over a sliding window of 3 seconds, which is consistent with the result reported in [62]. Feature warping is applied on short segments (typically 3s) based on the assumption that the underlying distribution is known (typically standard normal distribution) and any deviation is due to a distortion that requires normalization. A possible reason for this assumption is that the distribution of cepstral features extracted from clean speech over 3 seconds window, which corresponds to a period of one sentence, is approximately similar to the normal (or Gaussian) distribution. Noise and channel effects distort the shape of the cepstral feature distribution, therefore, normalizing the distribution removes the distortion caused by noise and channel.

In addition to mean and variance, feature warping normalizes the flatness and skewness of the feature distribution (see figure 3.8).

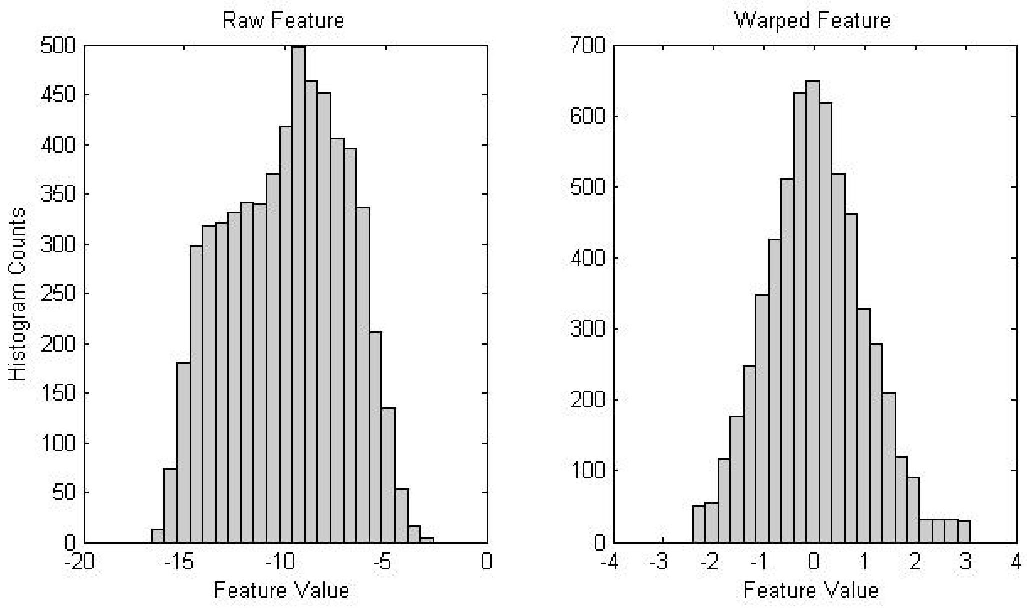


Figure 3.8: Histogram of raw and warped cepstral feature

3.8 Summary

This chapter introduced the basic techniques used in front-end processing of speech. Cepstral feature vectors were introduced as the most commonly used parametric representation of speech signals. The process of extracting cepstral features for language ID is described, with a focus on four different algorithms for spectral analysis. Although spectral analysis with DFT is the most widely used algorithm in speech technology, other algorithms are investigated because of their potential advantages for GPU implementation.

Silence does not carry useful information to discriminate between languages. Thus, labeling the input frames as speech or silence with VAD was described in this chapter. The method of calculating SDC coefficients, which have been proven to improve the performance of the language ID systems, was also described.

In real life the speech signal is usually accompanied by noise which distorts the speech cepstral features, and hence, degrades the performance of language ID systems. Different recording equipments such as microphones or handsets and communications channels such as landlines or wireless can cause different kinds of distortion. Ambient noise can also lead to corrupted speech and hence affect the extracted cepstral features. Some feature normalization techniques, which could help to remove unwanted distortion while keeping important speech information, were discussed.

Chapter 4

Phonotactic language ID systems

The Phonotactic approach is an essential part of most state-of-the-art language ID systems. It exploits language-dependent differences in the sequences of sounds in different languages. This chapter presents our phonotactic language ID system by describing in detail its main components, including 'tokenization', 'vectorization' and building Language Models (LMs) with SVMs.

4.1 Overview

Every language has a characteristic set of basic sounds – phonemes– such that each word of that language can be expressed as a sequence of phonemes from that set. Different languages have different phone sets, all are subset of the International phonetic Alphatic (IPA) [1]. The structure of phone sequences is determined by the words and syntax of the language, and hence contains useful information for language ID.

[1]http://www.langsci.ucl.ac.uk/ipa/handbook.html

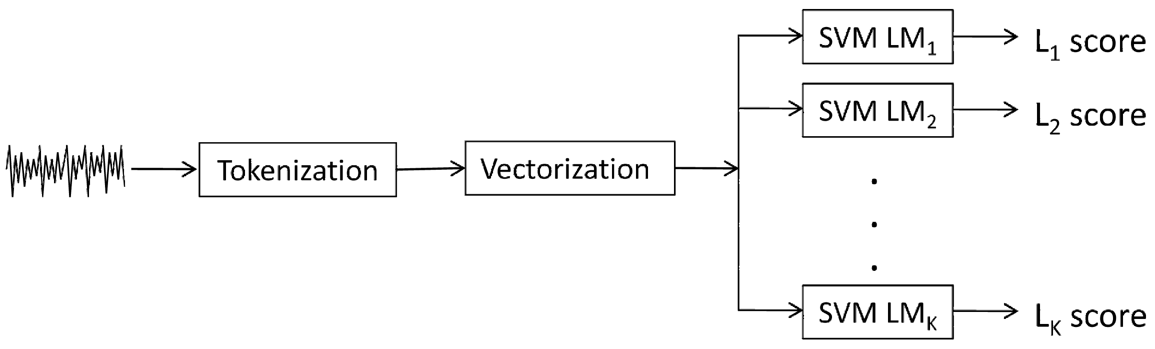


Figure 4.1: A simple phonotactic language ID system.

4.2 General framework

A typical structure of the phonotactic language ID system is shown in figure 4.1. A description of each component in the diagram is presented in the following sub-sections.

4.2.1 Tokenization

In order to extract the discriminative phonotactic information from speech signals and use it to distinguish between languages, a tokenizer is required to produce a sequence of symbols from the sequence of feature vectors. In a conventional PRLM approach the symbols are phones and the tokenizer is a phone recognizer, but other configurations are possible (for example, the GMM- n-gram systems in section 5.8).

Ideally, using a language-independent phone recognizer which incorporates the total set of phones in all languages as a tokenizer is the best for language ID, however a phone recognizer trained on any language can be used to label sections of speech with phones of that language. This is possible because, in principle, a good phone recognizer trained on a particular language will produce a systematic error for the phones which do not exist in that language, and hence capture useful phonotactic differences between languages. The problem of using a language-dependent phone recognizer is that it is limited by the phones used in that language,and therefore it can not model unknown phones which are used in the other languages. One way to overcome this limitation is by using multiple phone recognizers trained on different languages in parallel to generate a set of phone sequences. If some phones are not defined in one phone recognizer, there is a possibility that they may be defined in the other phone recognizers. The output sequences are then used to build a statistical model for each language.

4.2.2 Vectorization and weighting

The sequence of symbols is then used to estimate the probabilities of a predefined set of n-grams. The n-gram components of the sequence of symbols generated from an utterance Utt can be represented as a D-dimensional vector $p = (p_1, p_2, \ldots, p_D)$ where, D is the number of n-grams, C_j is the j^{th} n-gram and the probability p_j of C_j is estimated using counts of n-grams,

$$p_j = \frac{Count(C_j)}{\sum_i Count(C_i)} \tag{4.1}$$

where sum in 4.1 is performed over all n-grams and $Count(C_j)$ is the number of times the n-gram C_j occurs in the symbol sequence produced from the utterance. The result of the tokenization and 'vectorization' processes is that each utterance of speech is converted to a D-dimensional vector.

Each probability in the resulting vector is then weighted according to the utility of the corresponding n-gram for classification. In [63] the Inverse Document Frequency (IDF) used in Information Retrieval (IR), and the Log-Likelihood Ratio (LLR) weighting proposed in [64] are applied at the phone and word levels. According to [65], the LLR weighting technique outperformed IDF, and therefore is used in our systems.

The LLR weighting is applied to the n-gram probabilities in order to emphasize the most discriminative components (i.e. those which are common in one language but not in others), and de-emphasize the n-grams that are common in all languages. The weight w_j for the j^{th} n-gram component C_j is given by:

$$w_j = g_j(\frac{1}{p(C_j|All)}), \tag{4.2}$$

where g_j is a function used to smooth and compress the dynamic range (for example, $g_j = \sqrt{x}$) and $p(C_j|All)$ is the probability of n-gram component C_j across all languages (i.e. the prior probability of C_j).

The n-gram components which have zero occupancy in the training data of all languages are removed since they do not carry any useful information. A benefit of discarding these non-visited components is that it reduces the feature dimension dramatically, particularly for the high order n-gram system, as the dimension of the n-gram increases exponentially (S^n) where S is the total number of symbols (or phones).

Those n-gram components which have a very small probability have a very high weighting, allowing a minority of components to dominate the scores. To prevent this, a maximum threshold $T1$ on the weighting w_j is applied. According to Zipfs law, the rank-frequency distribution of words in a typical document follows a decaying exponential. The high ranking words with high probability are not useful for discrimination because they appear in most of the documents. Conversely, the low-rank words are too rare to gather useful statistical information. The area of interest is somewhere in the middle. This motivates us to apply a second, minimum, threshold $T2$ on the weighting vector to de-emphasise the common components. The values of $T1 = 200$ and $T2 = 40$ were determined empirically on development data set.

4.2.3 SVM LM

The traditional PRLM approach, which is very popular in language ID, uses a phone recognizer as a tokenizer followed by a set of language-specific n-gram LMs to capture language specific sound patterns. These n-gram LMs are typically trained with the ML criterion. They compromise the conditional probabilities of a phone given the history of the $n-1$ previous phones. Therefore the LM for a given language can be built by collecting n-gram statistics (counts) from 1-best phone strings and computing their conditional probabilities. Furthermore, Gauvain [66] showed, that n-gram statistics can be computed from the n-gram posterior probabilities taken from the phone lattices generated by the phone recognizer. This way, alternative phone recognition paths are taken into account, giving better estimates of the n-gram probabilities and improving the performance.

However, it has been shown in [67] that using SVMs to learn a discriminatively trained n-gram model is very effective for automatic language ID and outperforms the traditional n-gram approach. Therefore, the weighted n-gram probability vectors from utterances in the training sets are used to train a set of SVM n-gram LMs, one SVM for each language.

In recognition, a symbol (or phone) sequence is extracted from the test utterance; an n-gram probability vector is computed and weighted with the weight factor above. Then the weighted n-gram vector is evaluated using the SVMs for the different languages.

It has been shown previously (for example [9]) that using parallel PRLM systems with multiple phone recognizers trained on different languages and combining them in the back end improves the performance of language, dialect and accent ID systems. Therefore we have adapted this approach in our language ID system.

4.3 Summary

This chapter described the structure of our phonotactic language ID system. First, the ‘tokenizer’ converts the speech waveform into a sequence of symbols. In a conventional phonotactic language ID system the symbols are phones and the tokenizer is a phone recognizer. In vectorization this sequence is used to estimate a vector of probabilities of a predefined set of symbol n-grams. Each probability is weighted according to the utility of the corresponding n-gram for classification using LLR weighting. Finally, a set of SVM n-gram LMs is trained on the vectors from utterances in the training sets, using one SVM for each language.

Chapter 5

Acoustic Language ID Systems

In this chapter the theory of the acoustic-based language ID will be presented, with a focus on the most two popular and successful methods: GMMs and SVMs. Then some important techniques used in language ID, such as score normalization and language adaptation, will be introduced. Finally, state-of-the-art techniques which have been developed recently and successfully applied to language ID will be described.

5.1 Overview

Acoustic modeling methods have so far been proved to be among the most successful classification methods in speech technology, including language ID. Acoustic language ID systems are based on an assumption that the distributions of the acoustic features of languages are different. Therefore, the difference between these distributions for different languages can be used to distinguish between languages. The acoustic features extracted from speech signals in the front-end pre-processing are used to build an acoustic model for each language. The parameters of these models are estimated from a set of training speech data from that language. The language-dependent acoustic models are then used to produce a set of scores which can be used in recognition.

The acoustic features for language ID are often derived from short-term spectra, their time trajectories or SDC features (see chapter 3), prosodic information such as fundamental frequency, intensity, intonation and other possible features. GMMs which are extensively used in speech and speaker recognition, have also been successfully applied to language ID [34, 35, 68, 69, 70]. Recently other techniques have also been successfully used to discriminate between the acoustic information for language ID, such as SVMs [34, 58, 71, 72] and language models built on GMM tokenization [65, 67, 73, 74].

5.2 Theory of GMM

A GMM is a weighted sum of M Gaussian distributions called the components of the GMM, where each component distribution has a different mean, μ and covariance matrix, Σ. Provided that there are a sufficient number of mixture components, any shape of distribution for a specific language can be approximated very closely by the combination of these Gaussian mixture components. By assuming that observation vectors are independent, the probability that a specific language GMM model, λ_l, generates the sequence of T observation vectors $X = x_1, x_2, \ldots, x_T$, $p(X|\lambda_l)$ can be defined as

$$p(X|\lambda_l) = \prod_{t=1}^{T} p(x_t|\lambda_l), \tag{5.1}$$

$$p(x_t|\lambda_l) = \sum_{i=1}^{M} w_i G(x_t; \mu_i; \Sigma_i) \tag{5.2}$$

where G is D-dimensional Gaussian function of the form

$$G(x_t; \mu_i; \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp\left\{-\frac{1}{2}(x_t - \mu_i)^T \Sigma_i^{-1}(x_t - \mu_i)\right\} \tag{5.3}$$

with mean vector μ_i and covariance matrix Σ_i. The mixture weights w_i satisfy the constraint that $\sum_{i=1}^{M} w_i = 1$, $0 \leq w_i \leq 1$, $i = 1, 2, \ldots, M$. The complete Gaussian mixture density with M components is parameterized by the mean vectors, covariance matrices and mixture weights from all components. These parameters are collectively represented by $\lambda = \{w_i, \mu_i, \Sigma_i\}, i = 1, \ldots, M$.

GMMs are used in state-of-the-art language ID system to model the distribution of acoustic feature vectors in a given language. For example, thousands of Gaussian components can be trained for each language on features which provide information for discriminating between languages, usually static cepstra with SDC features .

While the general model form supports full covariance matrices, i.e., a covariance matrix with all its elements, we use only diagonal covariance matrices in all of our systems. This is done for three reasons. First, the density modeling of an M^{th} order full covariance GMM can equally well be achieved using a larger order diagonal covariance GMM. Second, diagonal-matrix GMMs are more computationally efficient than full covariance GMMs for training, since repeated inversions of a $D \times D$ matrix are not required. Third, estimating full covariances required more training data.

Given training speech from a language, the goal of acoustic language model training is to estimate the parameters of the GMM, λ, which in some sense best matches the distribution of the training feature vectors. There are several optimization criteria for estimating the GMM parameters [75]. However, the most popular and well-established technique is Maximum Likelihood (ML) estimation. The aim of ML estimation is to find model parameters which maximize the likelihood of the GMM, given the training data. The GMM likelihood of a sequence of T training vectors $X = x_1, \ldots, x_T$, as shown in equation 5.1, is a nonlinear function of the parameters λ and direct maximization is not possible. However, ML parameter estimates can be obtained iteratively using a special case of the Expectation-Maximization (EM) algorithm [76].

The basic idea of the EM algorithm is, beginning with an initial model λ, to estimate a new model $\bar{\lambda}$, such that $p(X|\bar{\lambda}) \geq p(X|\lambda)$. The new model then becomes the initial model for the next iteration and these two steps, expectation and maximization, are repeated until some convergence threshold is reached. On each EM iteration, the following re-estimation formulae are used, which guarantee an increase (or at least no decrease) in the model's likelihood value:

Mixture weights:

$$w_i = \frac{1}{T}\sum_{t=1}^{T} P(i|x_t, \lambda) \tag{5.4}$$

Means:

$$\mu_i = \frac{\sum_{t=1}^{T} P(i|x_t, \lambda)x_t}{\sum_{t=1}^{T} P(i|x_t, \lambda)} \tag{5.5}$$

Variances:

$$\Sigma_i^2 = \frac{\sum_{t=1}^{T} P(i|x_t, \lambda)x_t^2}{\sum_{t=1}^{T} P(i|x_t, \lambda)} - \bar{\mu}_i^2, \tag{5.6}$$

Where A *posteriori* probability for a GMM component i is given by

$$P(i|x_t, \lambda) = \frac{w_i G(x_t, \mu_i, \Sigma_i)}{\sum_{k=1}^{M} w_k G(x_t, \mu_k, \Sigma_k)} \tag{5.7}$$

There are two critical factors in the ML training of GMMs; selecting the number of Gaussian components, M, and initializing the model parameters prior to application of the EM algorithm. The number of GMM components is dependent on the amount of training data. Particularly in the NIST LREs, 2048 to 4096 components are usually used and the GMM parameters are initialized with several iterations of the k-means cluster algorithm [77].

The conventional ML training technique has two main drawbacks: first, it may converge to a local optimum point rather than to the global optimum and second, because each language model is trained independently of the others, it cannot obtain model parameters which explicitly minimize classification errors. A solution for the second of these ML drawbacks is to use discriminative training techniques, which focus on the boundaries between languages by taking into account data from the competing languages, as well as the target language. This is related to optimizing the posterior probability of languages given all training data. SVMs, Minimum Classification Error (MCE) and Maximum Mutual Information (MMI) are among the most popular discriminative training techniques [35, 78, 79]. In our study, we focused on the conventional ML training technique and SVM discriminative training, which is described in the following sections.

5.3 Maximum A Posterior (MAP) adaptation

Training a high order GMM for each language requires a large amount of training data from each language and extensive computation. For this reason, the approach of training one Universal Background Model (UBM), using training data from all languages (a language-independent model), and adapting it to create language-dependent GMMs using a more modest amount of language-dependent training data, is taken from speaker identification and successfully applied to the language ID application. MAP, or Bayesian, adaptation [80], with relevance factor, is one of the most common techniques used to adapt the UBM GMM parameters to language-dependent GMMs.

Like the EM algorithm, MAP adaptation is a two step estimation process. The first step is identical to the expectation step of the EM algorithm for GMMs, where estimates of the statistics of the language-dependent training data are computed for each component in the GMM. Unlike the second step of the EM algorithm, for adaptation these new statistic estimates are then combined with the old statistics from the UBM parameters using data-dependent mixing coefficients. These coefficients are designed so that the GMM components with high occupancy of training data for the language rely more on the new statistics for final parameter estimation, while components with low occupancy of data rely more on the UBM for final estimation of language-dependent GMMs, as shown in the following equations [81]:

$$\hat{w}_i = (\alpha_i^w w_i + (1 - \alpha_i^w) w_i^{ubm})\gamma \tag{5.8}$$

$$\hat{\mu}_i = \alpha_i^m \mu_i + (1 - \alpha_i^m)\mu_i^{ubm} \tag{5.9}$$

$$\hat{\Sigma}_i^2 = \alpha_i^v \Sigma_i^2 + (1 - \alpha_i^v)(\Sigma_i^{(ubm)2} + \mu_i^{(ubm)2}) - \hat{\mu}_i^2, \tag{5.10}$$

where, $\alpha_i^w, \alpha_i^m, \alpha_i^v$ are the adaptation coefficients controlling the balance between old and new estimates for the weights, means and variances, respectively. The scale factor, γ, is computed over all adapted GMM weights to ensure they sum to unity. $\hat{w}, \hat{\mu}$ and $\hat{\Sigma}^2$ are the adapted parameters, where w, μ and Σ^2 are the estimated parameters as shown in equations (5.4 - 5.6), and w^{ubm}, μ^{ubm} and $\Sigma^{(ubm)2}$ are the UBM weights, means and covariance, respectively.

For each GMM component and each parameter, a data-dependent adaptation coefficient α_i^ρ, $\rho \in \{w, m, v\}$ is used in the above equations. This is defined as

$$\alpha_i^\rho = \frac{P(i|x_t, \lambda)}{P(i|x_t, \lambda) + r^\rho}, \tag{5.11}$$

where, r^ρ is a fixed 'relevance factor' [81] for parameter ρ, and $P(i|x_t, \lambda)$ is the *posteriori* probability for a component i of GMM (λ).

5.4 GMM-UBM language ID system

In a GMM-UBM language ID system, a UBM is built using utterances from the combined training sets of all languages.

Language-dependent GMMs are obtained by MAP adaptation of the UBM parameters using language specific training data. We have found experimentally that there is only a minor gain of in using parameter-dependent adaptation coefficients. Therefore, our GMM-UBM system uses a single adaptation coefficient factor for all parameters ($\alpha_i^w = \alpha_i^m = \alpha_i^v$) with a relevance factor. We have also found experimentally that there is no benefit of adapting the covariance parameter, therefore in our GMM-UBM system, we adapt means and weights only. The result is one UBM and L language-dependent GMMs which are used to score the feature vectors of the unknown utterance. The resulting scores are then used in identifying the language of the unknown utterance, which could be a recognition or detection task, as explained in chapter 2.

5.5 SVMs

SVMs [82], which are a popular tool for discriminative classification [78], have recently been introduced for language ID [83]. An SVM is a discriminative binary (two-class) classifier. Unlike other classification approaches, which consider all feature vectors to model the probability distributions of the languages (generative classifiers), SVM only considers those feature vectors at the boundary between, for example a language and a set of impostor languages as they contain more discriminative information (discriminative classifier).

5.5.1 Basics

Since SVM is a binary classifier, the training data should have labels identifying training data for each class. Suppose the size of the training data is N with dimensionality of D, and the two classs labels are either -1 or $+1$. I.e. the form of the training data is as follow: $\{x_i, y_i\}$, where $i = 1, 2, \ldots, N$, and $y_i \in \{-1, +1\}, x \in \Re^D$. If we assume the data is linearly separable, a line can be drawn to separate the two classes in two dimensional data and hyperplane for higher dimensions ($D > 2$). A hyperplane can be described by the general equation: $w.x + b = 0$ where, w is normal vector to the hyperplane specifying the orientation of the hyperplane, and the bias constant b determines the distance from the origin. Although many hyperplanes might be found separating the two classes, SVM aims to find the hyperplane which has the maximum margin. The SVM margin is the separating hyperplane's equidistance from the two boundary hyperplanes ($d1$ and $d2$ in figure 5.1).

For simplicity, if we look at the two dimensional data example in figure 5.1, we will see that the training data can be described by:

$$y_i(x_i.w + b) - 1 \geq 0 \qquad \forall_i \tag{5.12}$$

The data points which lie close to the separating hyperplane are called support vectors

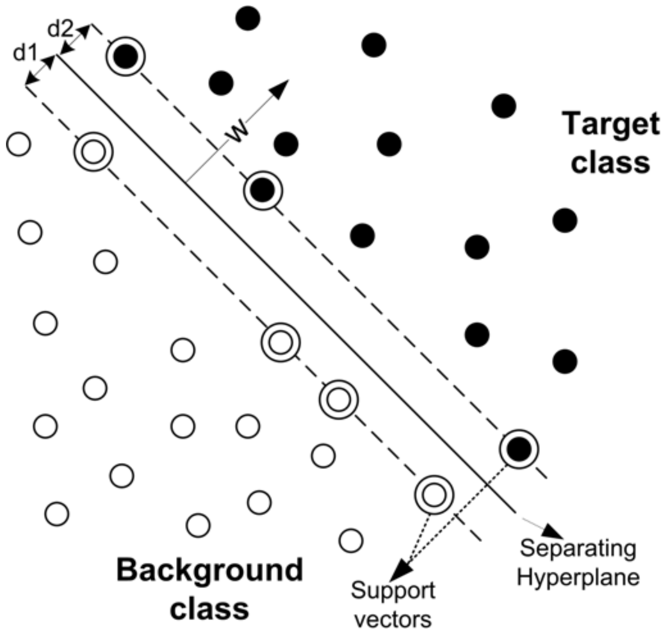


Figure 5.1: SVM — two-dimensional data illustration

(shown in circles in the diagram). In order to orientate the hyperplane to be as far from the support vectors as possible, geometrically the margins ($d1$ and $d2$) need to be maximized. Since the geometric margin is proportional to $\frac{1}{\|w\|}$, maximizing the margin is same as minimizing $\|w\|$ such that $y_i(x_i.w + b) - 1 \geq 0 \quad \forall_i$. The simplest way to solve this optimization problem with this constraint is by introducing Lagrange multipliers α_i, where $\alpha_i \geq 0$ as in the following equation [78]:

$$L_p \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i[y_i(x_i.w + b) - 1] \tag{5.13}$$

Equation 5.13 is well-known as the *primal* form of the SVM. Now the target is to find w and b which minimize α_i (keeping $\alpha_i \geq 0$) and maximize this *prime* equation (5.13). One way of solving this optimization is by differentiating L_p with respect to w and to b and set the derivatives to zero.

$$\frac{\partial L_p}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{5.14}$$

$$\frac{\partial L_p}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{5.15}$$

This indicates that the vector w is a linear combination of the support vectors. By substituting the optimum value of w into the *prime* form (L_p) above, we will get the *dual* form, L_D, of the *primal* form:

$$L_D \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i.x_j \quad s.t. \;\; \alpha_i \geq 0 \;\; \forall_i, \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{5.16}$$

As we see in the *dual* form above, two constraints should be kept: $\alpha_i \geq 0$ and the new constraint for the optimum bias, $\sum_{i=1}^{N} \alpha_i y_i = 0$. Another important thing to note in the *dual* form is that only the dot product of the input vectors is required to be calculated.

$$x_i.x_j = x_i^T x_j = k(x_i, x_j)$$

where, $k(x_i, x_j)$ is called $KernelFunction$ ($k(x_i, x_j) = x_i^T x_j$ is known as a $LinearKernel$).

The only variables available in the *dual* form are the Lagrange multipliers α_i, so the target now is to find the values of these variables which maximize the dual form L_D, and keep the two constraints. The best way to solve this quadratic optimization is by using Quadratic Programming (QP). By finding the optimal value of α_i, the w vector can be found using equation 5.14, and the scalar b can be found using $y_s(x_s.w + b) = 1$ where x_s is a support vector satisfying constraint in 5.15. The SVM parameters, w and b are only required to specify the maximal margin separating hyperplane, and therefore, are used to define the SVM cost function which is used to classify new input data, x',(test data points):

$$f(x') = x'.w + b = \sum_{i=1}^{N} \alpha_i y_i k(x', x_i) + b, \quad (5.17)$$

where, x_i is the support vector and $k(x', x_i) = x'^T x_i$ is a linear kernel function. The more positive the value of $f(x')$ the more likely x' belongs to the target class (+1), and conversely, the more negative $f(x')$, the more likely x' belongs to the background class (-1).

In most cases, the situation is not as simple as described above. Particularly in speech applications, acoustic features are not linearly separable, so the linear SVM cannot be used as described above. To deal with the data that is not linearly separable, there are two methods; the first one is by introducing a slack factor which relaxes the constraints of SVM slightly allowing for misclassified points, and then trading-off between the slack variable and the size of margin. The second way of dealing with non-linearly separable data is by projecting the data points into a high-dimensional space in which the data becomes linearly separable, hence a linear SVM can be applied. This means that if the kernel function can be recast into a higher dimensionality space by some potentially non-linear feature mapping function $x \rightarrow \phi(x)$, only inner products of the mapped input data points in the high-dimensional space need to be determined, without needing to explicitly calculate ϕ. This is useful because in some practical classification problems, such as language ID, the classes are not linearly separable in the input feature vectors, but they might be linearly separable in a higher dimensionality feature domain given a suitable mapping function $x \rightarrow \phi(x)$. Thus, the SVM cost function in equation 5.17 can be written as:

$$f(x') = \sum_{i=1}^{N} \alpha_i y_i k(\phi(x'), \phi(x_i)) + b, \quad (5.18)$$

Not any function can be used as kernel function. The Kernel function must satisfy a specific condition called the Mercer condition, which guarantees convergence to the iterative QP process for optimizing the dual form in equation 5.16. The Mercer condition is that the kernel matrix, whose entries are the evaluation values of the kernel function for all input feature vector pairs, must be a positive definite matrix. The Radial Basis Kernel, Polynomial Kernel and Sigmoidal Kernel are among the most popular non-linear kernel functions which satisfy the Mercer condition. All of the SVM models in this thesis were trained and evaluated using the SVM-KM SVM MATLAB toolbox [84].

5.5.2 SVM for language ID

Since language ID is a multi-class problem, SVM (two-class classifier) can not be used directly. There are normally two strategies to address this problem. Firstly, the one-to-one strategy can be used, where for every two languages an SVM is trained to identify one language from the other. Secondly, the one-against-others strategy can be used, where for every language an SVM is trained to separate it from the others. The second strategy requires fewer SVMs and it is more efficient than the first strategy while achieving comparable performance. The training process for the English language using this strategy is depicted in figure 5.2.

The second issue for applying SVMs to the language ID task is choosing a kernel to handle the sequence of acoustic feature vectors. A sequence kernel, $k(x_i^a, x_i^b)$, is a kernel that compares the two sequences of feature vectors x_i^a and x_i^b. The main issues in constructing a kernel are: making the kernel a relevant comparison between two sequences and satisfying the Mercer condition described earlier.

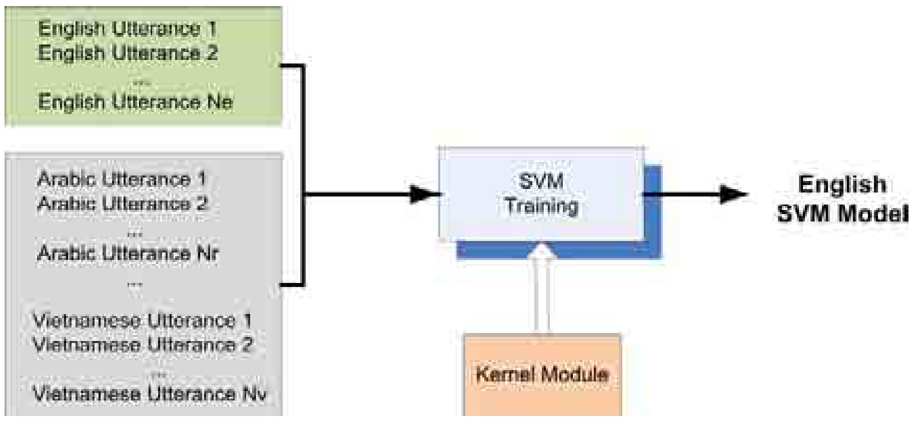


Figure 5.2: SVM training strategy — English SVM model

One way of constructing a sequence kernel for speech applications is by training on one sequence and testing on another one. This idea is firstly developed by Campbell in [83, 85] and leads to a sequence kernel named the Generalized Linear Discriminant Sequence (GLDS) Kernel. The GLDS kernel achieves promising results in language ID experiments.

Another way of handling this situation is by training statistical models from the input sequences, and defining the kernel as a similarity between the two estimated distributions. The Kullback divergence [86, 87] and Bhattacharyya affinity [88] are of this kind. The main drawback of this kind of kernels is that the usual shortness of test sequences (no more than 30 seconds) prevents a robust estimation of distribution parameters. The most successful approaches of this type are those which use GMMs to model the two input sequences, as described in the following sub-section.

5.5.3 GMM-SVM language ID system

The basic idea of this approach is to model the sequence of acoustic feature vectors of every utterance with a GMM, and then to find a kernel function that measures the similarity between the GMMs and satisfies the Mercer conditions. For simplicity, the GMM parameters are usually combined into one high dimensional vector called a 'supervector'.

Since the acoustic features of a single utterance are usually not sufficient to train a GMM, MAP adaptation is used to adapt UBM GMM parameters, in the same way described for the GMM-UBM language ID system in section 5.3, but per utterance rather than per language. Adapting the GMM weights, w_i, and covariances, Σ_i, does not show any gain for language ID performance [65], therefore only GMM mean vectors, μ_i, are adapted and concatenated to form supervectors. Consequently, if we assume M is the number GMM components and D is the dimensionality of the acoustic feature vectors, the dimensionality of the supervectors will be $M \times D$. Figure 5.3 illustrates the process of constructing supervector from GMM mean vectors.

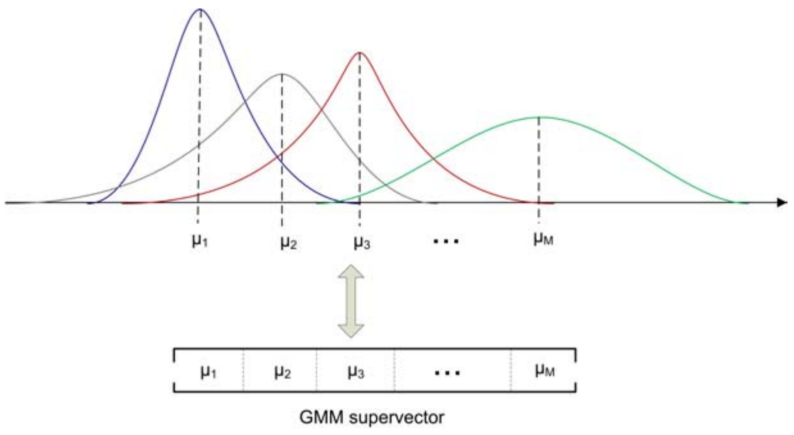


Figure 5.3: An illustration for constructing supervectors from GMM means

One possible way of measuring the similarity between the GMMs is the KL-divergence. Unfortunately, the KL-divergence does not satisfy the conditions to be a SVM kernel because a matrix of kernel distances directly based on symmetric KL divergence does not satisfy the Mercer conditions, i.e., it is not a positive definite matrix. However, an approximation of the KL-divergence which represents the distance between two GMMs can be used as a kernel function [87], as shown in the following equations.

$$k(utt_a, utt_b) = \sum_{i=1}^{M} w_i \mu_i^a \Sigma_i^{-1} \mu_i^b \tag{5.19}$$

$$= \sum_{i=1}^{M} (\sqrt{w_i} \Sigma^{-\frac{1}{2}} \mu_i^a)^t (\sqrt{w_i} \Sigma^{-\frac{1}{2}} \mu_i^b) \tag{5.20}$$

$$= \phi(utt_a)^T \phi(utt_b) \tag{5.21}$$

where, μ_i^a and μ_i^b are the adapted means of GMM component i, with utterances utt_a and utt_b respectively, w_i and Σ_i are weights and variances of the UBM, $k(.,.)$ is the kernel function. The resulting supervectors are then used with the kernel defined above to train one SVM for each language with the one-against-others strategy.

During recognition, each test utterance is mapped into the supervector domain by MAP adapting the UBM means. The resulting supervector is then input to each of the language-dependent SVM models, and a score is calculated with the SVM cost function shown in equation 5.17. Positive output scores indicate that the test utterance belongs to the target language, and negative scores that the utterance belongs to the non-target (background) languages. Thus, SVM output scores can be interpreted as log-likelihood scores (i.e. GMM scores) and the same score normalization techniques can be applied. We refer to this as GMM-SVM language ID system from here onwards.

5.6 GMM-SVM-GMM language ID system

Recently the possibility of transferring the SVM parameters back to a GMM which is then used for scoring, has been explored [72, 89]. The challenge in this paradigm is to understand the exact role of the SVM parameters in GMM scoring.

Campbell and Karam in [90] illustrated the connection between SVM scoring and GMM scoring in the context of GMM-SVM language ID system described in the previous section, and showed that GMM scoring yielded better results. As we have seen earlier, SVM depends on the support vectors at the boundary of the target and background classes for discriminating between them. We also know that support vectors in GMM-SVM system are in the supervector domain, i.e. a concatenation of GMM mean vectors. Therefore, weighted average of both the target, $\mathbf{m}_{sv}^{+}$, and background, $\mathbf{m}_{sv}^{-}$, support vectors can be split into a set of mean vectors (reverse operation in figure 5.3) which are then used to create target, λ_{target}, and background, $\lambda_{background}$, GMMs. UBM weights and covariance vectors are used for the the two GMMs.

$$\mathbf{m}_{sv}^{+} = \frac{1}{\sum_{\{i|y_i>0\}} \alpha_i} \sum_{\{i|y_i>0\}} \alpha_i x_i \tag{5.22}$$

$$\mathbf{m}_{sv}^{-} = \frac{1}{\sum_{\{i|y_i<0\}} \alpha_i} \sum_{\{i|y_i<0\}} \alpha_i x_i \tag{5.23}$$

where, x_i is the support vectors, α_i is the Lagrange multiplier, and y_i is the data labels ($+1$ or -1). x_i and α_i are obtained from SVM training.

The result of this technique is two GMMs, λ_{target} and $\lambda_{background}$, for each language, which are used to calculate a score:

$$score = \sum_t \log p(x_t|\lambda_{target}) - \sum_t \log p(x_t|\lambda_{background}) \tag{5.24}$$

where, x_t is acoustic feature vector of test utterance at time t. We refer to this as a 'GMM-SVM-GMM' language ID system from here onwards, to distinguish it from our previous GMM-SVM language ID system.

5.7 Inter-session compensation (ISC)

Amongst the most significant factors that impact on the performance of speech classification applications, particularly language ID, are variability caused by changes in channel, speaker and noise.

There are many techniques in speech technology that are specifically used to compensate variability between channel conditions, such as the feature normalization techniques described in chapter 3. Another technique was proposed by Reynolds in [91]. The main idea of this technique is that channel mismatch can be reduced by estimating a model to represent each type of channel (e.g. handset type carbon-button or electrets for landline, cellular network; GSM analogue or digital etc). This estimation can be done by building a channel-independent model (root model) and then adapting it to each channel type by using labeled training data for each channel. In recognition, the channel type of the test segment is identified first by comparing the scores of the channel-dependent models. The channel-dependent model which gives the maximum score is selected and used in the evaluation. The drawback of this method is that each utterance in the training data should be labeled with the channel type, which is not easy to get.

In addition, there are other techniques for dealing with inter-speaker variability within a single language, such as Vocal Tract Length Normalization (VLTN) [92], and for compensating for noise such as noise masking [93] and Parallel Model Combination (PMC) [94].

Recently, in state-of-the-art language ID systems, a powerful technique called Inter-Session Compensation (ISC) has been proven to improve language ID performance dramatically [95]. For the perspective of language ID, inter-session variability can be defined as anything that makes one utterance in a particular language different from another. Many factors can cause inter-session variability within a single language. The most important factors include inter-speaker variability, channel conditions, background noise and length of utterances. The basic idea behind this technique is that the distortions due to inter-session variability in the high-dimensional supervector space can be summarized by a small number of parameters in a lower dimensional subspace, which are called the channel factors [96]. The ISC technique can be applied in the model domain and in the acoustic features domain. In the case of the GMM-UBM approach, model domain compensation is done by shifting the means of the UBM and all of the language-dependent GMMs towards the inter-session variability direction estimated from the test utterance as shown in equation 5.25

$$\hat{\mu}_{sv} = \mu_{sv} + UX_c \tag{5.25}$$

where $\hat{\mu}_{sv}$ and μ_{sv} are the compensated and the original means of UBM and all language-dependent GMMs in the supervector domain. X_c is an R-dimensional vector comprising the channel factors for the test utterance. U is a low rank matrix projecting the channel factors X_c from low-dimensional inter-session variability domain to the high-dimensional supervector domain. U is called the 'Eigen-channel subspace matrix'.

Eigen-channel subspace estimation : We adopted the term eigen-channel as used in Speaker Recognition Evaluation (SRE) from Kenny [96]. It was introduced to the NIST SRE by Spescom Datavoice (SDV) in 2004 [97], revisited by Kenny and Vogt [98] in NIST SRE 2005, and again by several sites in various forms.

Before eigen-channel adaptation can be applied, we must identify directions in which a supervector is mostly affected by inter-session variation. These directions (eigen-vectors), are defined by the columns of an $MD \times R$ matrix U, where M, D and R are the number of GMM components, the dimension of the feature vectors and the chosen number of eigen-vectors that represent major directions of inter-session variability (R is usually $\geq$50), respectively. The matrix U is given by R eigen-vectors of an average within class covariance matrix, where each class is represented by supervectors estimated on different utterances spoken in the same language.

For each language, l, and each of its conversations, $\{j = 1, \ldots, J_l\}$, the UBM is adapted to obtain a supervector, sl_j . The corresponding average language supervector given by

$$\bar{s_l} = \frac{1}{J_l} \sum_{j=1}^{J_l} sl_j \tag{5.26}$$

is subtracted from each supervector, sl_j , and the resulting vectors form columns of an $MD \times J$ matrix S, where J is the number of all conversations from all languages, $J = \sum_l J_l$. The assumption is that when $\bar{s_l}$ is subtracted from sl_j, the resulting vector is due to inter-session variability. The columns of matrix U are given by the R eigen-vectors of the $(MD \times MD)$ covariance matrix SS^T corresponding to the R largest eigenvalues. Unfortunately, for our system, where supervectors are very high dimensional vectors (e.g. $MD = 4096 \times 68 = 278528$), using Principal Component Analysis (PCA) directly to compute these eigen-vectors is unfeasible. A possible solution is to use an iterative algorithm which estimates U matrix from the eigen-vectors of $J \times J$ matrix $S^T S$.

Eigen-channel adaptation : Once the eigen-channels are identified, a GMM for each language (or UBM) can be adapted to the channel of the test utterance by shifting its supervector in the directions given by eigen-channels, to better fit the test utterance data. Mathematically, this can be expressed as finding the channel factors, X_c, that maximize the following MAP criterion:

$$p(X|s + UX_c)N(X; 0; I), \tag{5.27}$$

where s is the supervector representing the model to be adapted, $p(X|s+UX_c)$ is the likelihood of the test utterance, X, given the adapted supervector (model) and $N(; 0; I)$ denotes a standard multivariate Gaussian. Assuming fixed occupation of Gaussian mixture components by the test utterance frames, $X = x_1, x_2, \ldots, x_T$, it can be shown [97] that the value of X_c that maximizes the criterion (Equation 5.27) is given by:

$$X_c = A^{-1} \sum_{i=1}^{M} U_i^T \sum_{t=1}^{T} \gamma_i(t) \frac{x_t - \mu_i}{\sigma_i}, \tag{5.28}$$

where U_i is the $D \times R$ part of the U matrix corresponding to the i^{th} mixture component, $\gamma_i(t)$ is the probability of occupation mixture component i at time t, μ_i and σ_i are the mixture components mean and standard deviation vectors and

$$A = I + \sum_{i=1}^{M} U_i^T U_i \sum_{t=1}^{T} \gamma_i(t). \tag{5.29}$$

In our implementation, the occupation probabilities, $\gamma_i(t)$, are computed using the UBM and assumed to be fixed for a given test utterance. This allows the matrix A^{-1} to be pre-computed only once for each test utterance.

This method uses a very simple scheme of modeling channel variability that affects only the recognition phase. It cannot be applied to a GMM-UBM language ID system with different number of mixture components because a new U matrix needs to be estimated, nor to other language ID systems such as GMM-SVM or GMM-SVM-GMM. However, ISC technique can be applied in the acoustic feature domain.

The adaptation of a feature vector $\hat{x}_t$ is obtained by subtracting a weighted sum of compensation offset values from the original feature vector x_t according to:

$$\hat{x}_t = x_t - \sum_{i=1}^{M} \gamma_i(t) U_i X_c \tag{5.30}$$

where U_i and x_c are estimated in the same way as for the adaptation in the model domain. ISC can be applied to all acoustic feature vectors as a part of front end before different types of language ID.

5.8 GMM Tokenization

GMMs, which are generally employed to measure the acoustic match between the input feature vectors, $X = x_1, x_2, \ldots, x_T$, and the language l, can also be used to act as speech tokenizers [65, 67, 73, 74], i.e. to convert the acoustic vector stream into a discrete symbol stream. This is done by replacing each vector x_t of X by the label of the Gaussian component which produces the highest contribution in equation 5.1. This then makes it possible to apply phonotactic approaches to the language ID problem at the acoustic level by replacing the sequence of linguistic tokenizing symbols (phones) by a sequence of GMM component indices.

The resulting sequence is used to train n-gram LMs for each language using SVMs, as described in chapter 4 and depicted in figure 4.1. Compared with the PRLM system described in chapter 4, the phone recognizers are replaced by a language-independent GMM which produces a sequence of Gaussian component indices instead of a sequence of phones. The other parts of these two types of system are the same, including the use of discriminative weighting to emphasize the GMM component n-grams which represent the language specific features and de-emphasize the components which represent features that are common in all languages [65]. To the best of our knowledge this is the first use of LLR weighting in a GMM tokenization system. We refer to this system as 'GMM-n-gram' throughout the rest of this thesis.

The advantage of the GMM tokenizer is that it does not require phonetically transcribed data, and it can be trained on the same acoustic data that is used to train the acoustic-based language recognition system. This is computationally less intensive than the phone recognizer.

There are two ways to use a GMM as a tokenizer for the GMM-n-gram system. The first way is to use a language-independent GMM (e.g. UBM) trained on the data of all languages, as a tokenizer, and the second way is to use multiple language-dependent GMMs, trained individually on the language specific training data, as tokenizers and then combine them together at the back end.

We suggest that the most discriminative n-gram components are common in one language and rare in others. Increasing the order of GMM allows these features to occupy separate components. We achieve this with low computational cost and less training data by replacing the traditional UBM with a Multi-Language Model (MLM), which is a concatenation of multiple language-dependent GMMs, as described in the following section.

5.8.1 Multi-Language Model (MLM)

We hypothesize that increasing the number of UBM components will cause language-specific information to be represented in separate components. In n-gram systems these components contain the most discriminative information. This is the motivation for our high order MLM. For a conventional EM-trained UBM, increasing the model order necessitates more training data and computation. These problems are alleviated in our MLM.

The MLM model is a concatenation of language-dependent GMMs, each trained separately on language specific training data using the EM algorithm. The resulting GMMs are combined to form a single large MLM. The MLM gives more space to represent language specific information, which is emphasized with the discriminative LLR weighting, where the common information is de-emphasized. Each language-dependent GMM can be of different order, depending on the available enrollment data. Training a MLM also requires less computation than training a comparable UBM.

If we assume $\lambda_l = \{w_l, \mu_l, \Sigma_l\}$ are the GMM parameters for language l, the MLM model is formed by concatenating the parameters of all language-dependent GMMs,

$$\lambda_{MLM} = \left\{ \frac{[w_1, \ldots, w_L]}{L}, [\mu_1, \ldots, \mu_L], [\Sigma_1, \ldots, \Sigma_L] \right\} \tag{5.31}$$

where λ_{MLM} is the set of MLM model parameters and L is the total number of language-dependent models. To date we have built gender dependent MLMs of order up to 24,576.

5.9 Score normalization

An important issue in the statistical approaches to language ID is score normalization, which scales the log-likelihood score's distribution. Scaling the score distributions of different languages is used to find a global language independent threshold for the decision making process, and to reduce the unwanted environmental effects.

There are several commonly used score normalization techniques such as Zero Normalization (Z-Norm) and Test Normalization (T-Norm) which have been successfully applied to speaker identification [99]. In Z-Norm, the mean and standard deviation of the impostor scores are estimated off-line, and then the estimated mean is subtracted from each score, which is then divided by the estimated standard deviation. The normalization has the form:

$$\hat{S}_l = \frac{S_l - \mu_I}{\sigma_I} \tag{5.32}$$

where, μ_I and σ_I are the estimated impostor parameters for language model l. S_l is the log-likelihood score and the $\hat{S}_l$ is the Z normalized score.

T-Norm also depends on mean and variance estimation for distribution scaling. During testing, a set of impostor models is used to estimate the impostor log-likelihood scores for a test utterance, and the mean and variance parameters are estimated from these scores. These parameters are then used to perform the score normalization by using the formula in equation 5.32.

However, unlike speaker identification, in language ID there are not sufficient impostor models to estimate the parameters required for T-norm (in our case only 11 impostor languages are available for each target language). Alternatively, a score calibration and normalization technique called log-likelihood ratio, has been proven to be useful for language recognition [58].

$$\hat{S}_l = S_l - \log\left(\frac{1}{L-1}\sum_{j\neq l} e^{S_j}\right) \tag{5.33}$$

where L is the total number of target languages. A similar method to this normalization technique is the 'max-loglikelihood' score normalization, which has also been applied successfully to language ID systems [90]. In this normalization, the log-likelihood score of each target language is normalized with the score of the most competitive language model, instead of the average score of all competitor (impostor) languages.

$$\hat{S}_l = S_l - max_{i \neq l} S_i \tag{5.34}$$

5.10 Summary

This chapter presented the acoustic modelling methods which are typically used in state-of-the-art language ID systems. A brief theoretical description of the probabilistic models, GMMs and the discriminative models, SVMs, and a combination of both were presented.

Three components of our language ID system, GMM-UBM, GMM-SVM and GMM-SVM-GMM were described in this chapter. The GMM-UBM approach adapts a language-independent model, the UBM, to language-dependent GMMs using MAP adaptation. The GMM-SVM approaches uses the UBM and MAP adaptation to map each utterance of the target languages from the acoustic feature vector sequence domain to a high dimensional supervector domain where an SVM is used to find the best separating hyperplane between each target language and its impostor languages. The resulting language-dependent SVM models are used to evaluate the GMM-SVM sub-system. They are also 'pushed back' from supervector domain to the GMM domain to form the GMM-SVM-GMM sub-system.

Using GMMs in a different way, called GMM tokenization, was also presented in this chapter. The idea of GMM tokenization is to use a language-independent GMM as a tokenizer to produce a sequence of GMM component indices from the feature vectors. This allows a system similar to the phonotactic language ID system described in chapter 4 to be built, by using a GMM instead of phone recognizer as a tokenizer. The resulting system is called GMM-n-gram which is another component of our language ID system.

A powerful technique to compensate the inter-session variability within a single language, called ISC, was described in detail in this chapter together with four different techniques to normalize the score distributions.

Chapter 6

Fusion

State-of-the-art language ID systems typically make use of several acoustic and phonotactic sub-systems. Combining the outputs of these sub-systems generally improves the language ID performance. This chapter gives a brief description about the most common fusion techniques used in language ID, and the general structure of our fused systems.

6.1 Overview

Combining disparate systems which use different features or different modeling approaches has proven to be a very successful and popular method in language ID.

The term 'fusion' has been widely used to refer to the method of combining different systems together. In most cases, 'fusion' simply refers to a more specific category, namely 'score fusion', which produces a new set of likelihood scores from the original likelihood scores produced by the existing individual language ID systems.

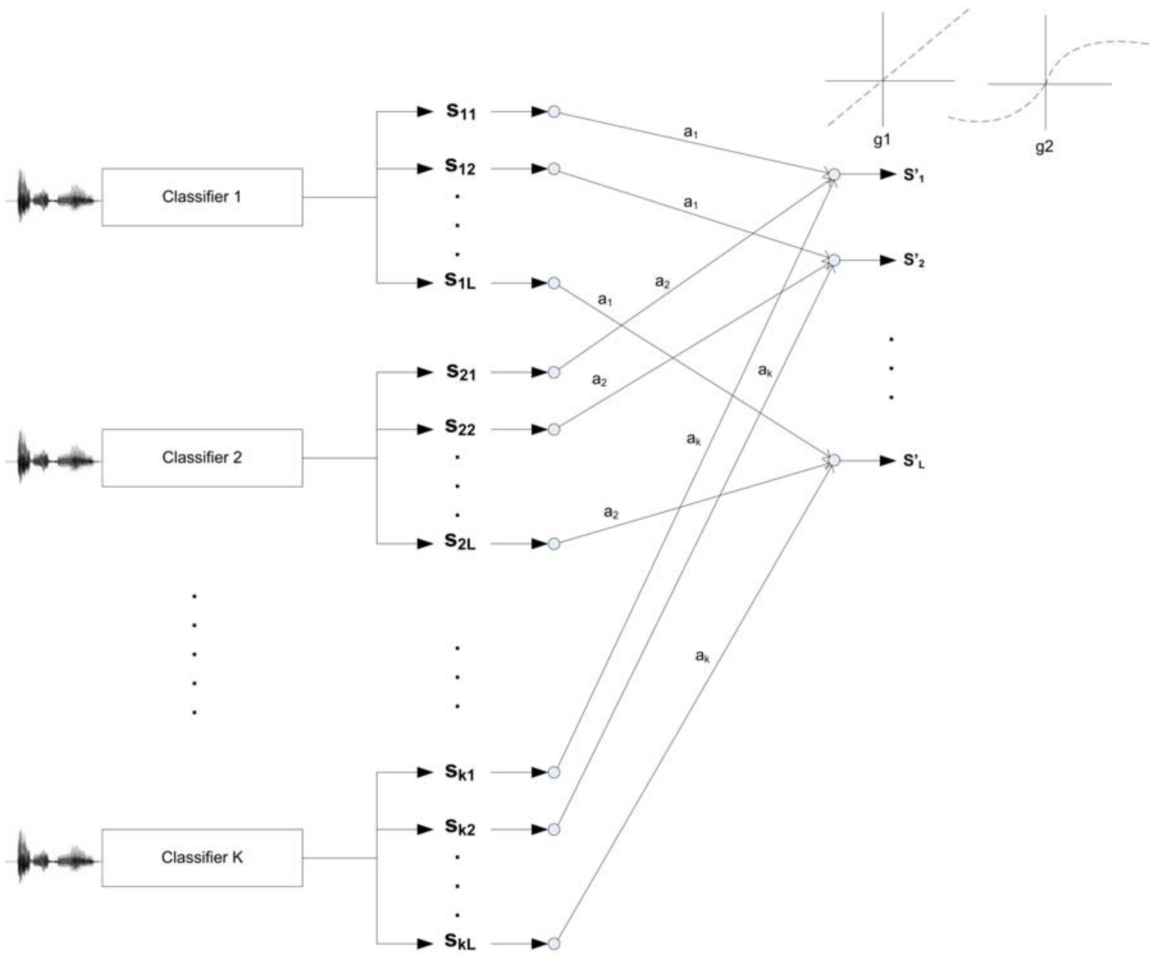


Figure 6.1: Within-class Linear fusion

6.2 Fusion techniques

Multi-class classifiers can be fused in two possible ways: within-class fusion and cross-class fusion. Within-class fusion means that the output fused scores are obtained by fusing scores of the corresponding classes from all classifiers (see figure 6.1), whereas in the cross-class fusion, the output scores are obtained by fusing the scores of all classes from all classifiers. These two fusion methods can be linear or non-linear.

Linear fusion is one of the most widely used techniques in language ID [100]. The output fused score, s', as shown in figure 6.1, is a linear combination of the scores, s, produced by the individual sub-systems. Assuming the final fused system consists of K sub-systems, with L classes, the fused score, s'_i is calculated as:

$$s'_i = a_0 + \sum_{j=1}^{K} a_j s_{ji} \tag{6.1}$$

where, a_j, $(j = 1, \ldots, K)$ is the weighting coefficients for output scores of classifier j, a_0 is a bias coefficient, and s_{ji} is the score of class i of classifier j. The problem now is finding the optimal weighting coefficients, a_j, which optimizes the system performance. There are many ways to solve this problem. Assuming there are sufficient training examples, the weighting coefficients can be calculated with linear algebra based on the Least Squared Error (LSE). Assuming we have N training utterances. Each utterance will produce $K \times L$ scores. Ideally, the output scores of the fused system are L-dimensional vectors of zeros, for the impostor classes and ones, for the true classes. By adding a vector of L ones to the classifier's scores, the linear relationship between the input scores and the fused scores can be written in matrix notation in terms of the unknown coefficients,

$$S^T A = S' \tag{6.2}$$

where, S is a matrix of scores of dimension $L \times N \times (K+1)$, S' is an $L \times N$ matrix of output scores and A is a vector of $K+1$ coefficients a_k. Therefore, the direct solution for this linear equation is $A = S^{-1}S'$, where S^{-1} is the inverse of S^T matrix. Unfortunately, S is not, in general a square matrix or invertible, so that the inverse cannot be calculated directly. However, A can be estimated by minimizing $\sum_{n=1}^{N} ||s_n A - s'_n||^2$, which can be solved with the pseudo inverse of S^T, $S^\dagger$

$$A \approx S^\dagger S' \tag{6.3}$$

Brummer in [101] suggested that replacing the linear activation function, $g1$ in figure

6.1, by an exponential function such as logistic regression function, $g2$ in figure 6.1, de-emphasises the large scores, and is more sensitive to the boundary scores, which are more important for fusion. This technique is called linear logistic regression [101, 102]. A toolkit known as the FoCal Multi-class toolkit [1]that implements linear logistic regression fusing is available and very popular in language ID. Therefore, it was used to do all fusion in this thesis.

The system depicted in figure 6.1 is an example of a Multilayer Perceptron (MLP). Hence, the weighting coefficients can be estimated using the standard training algorithms for the MLP. However, due to the limited amount of available training examples, this often results in non-robust estimates of the weightings parameters. One possible solution for this is to approximate the distribution of the scores, S, by a particular parametric distribution, for example a Gaussian distribution, and to estimate the distribution parameters. The weights can then be calculated in terms of the distribution parameters [103]. In other words, correctly estimating approximate distributions often yields better results than approximating exact distributions.

Besides the commonly-used linear logistic regression technique, another popular fusion technique is GMM fusion [34]. In GMM fusion , a GMM with a small number of mixture components classifier for each class is deployed for score fusion. The input vector consists of the output scores from all individual language ID systems. For each class, the GMM is trained on score vectors produced by the individual systems using the development data. Given a new vector for an unknown test utterance, the class-dependent GMMs are used to estimate the output likelihood scores, which are used for the final decision.

[1]http://niko.brummer.googlepages.com/focalmulticlass

Recently, several score fusion techniques have been proposed for the language ID task, including SVMs [104]. In this reference, a comparison of most of these techniques can be found.

6.3 The fused language ID system

Our final language ID system is obtained by fusing a number of phonotactic and acoustic language ID systems, as depicted in figure 6.2.

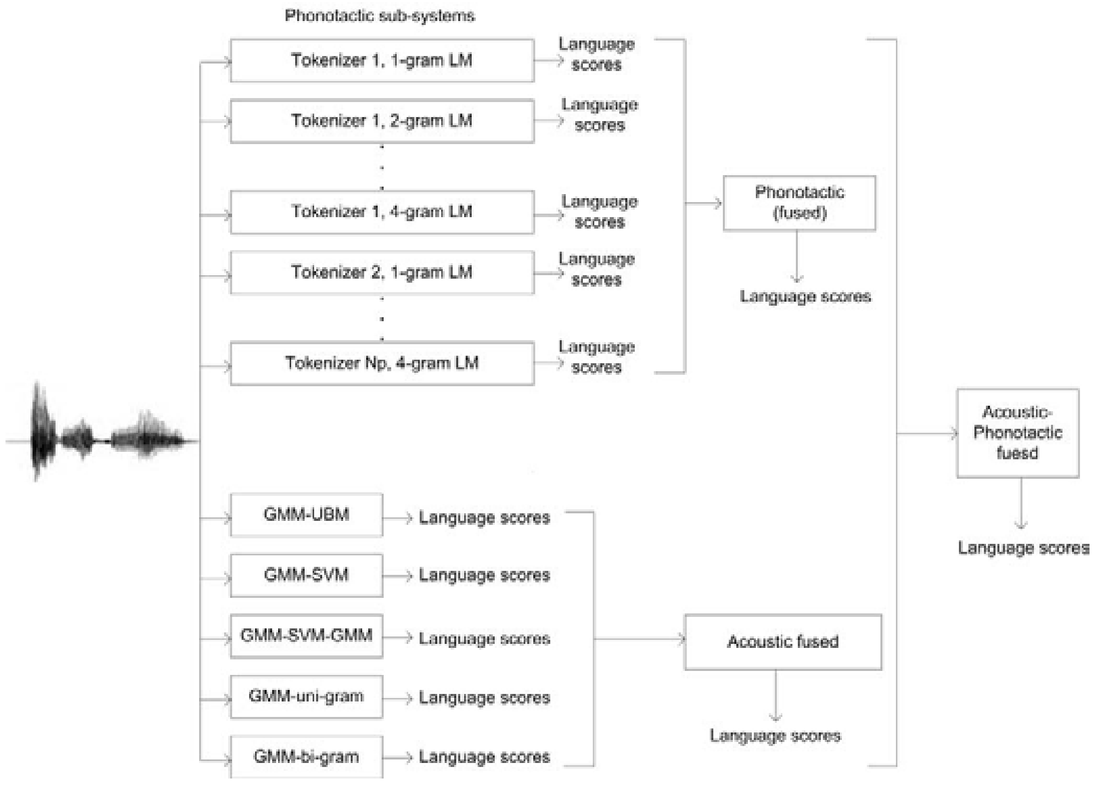


Figure 6.2: Phonotactic and acoustic fused language ID system.

The different phonotactic systems are built by either changing the tokenizer (e.g. phone recognizer trained on different languages) or modeling the sequence of symbols generated by a particular tokenizer with different order of n-grams (usually n=1,2,3 and

4). If we assume N_p is the number of tokenizers, $N_p \times 4$ different phonotactic systems can be built with $1, 2, 3, 4$-grams. In our system $N_p = 3$ and the different languages are Czech, Hungarian and Russian. Fusing these phonotactic sub-systems together gives our phonotactic fused language ID system (phonotactic fused block in figure 6.2).

The second category of language ID systems is the acoustic based systems. As described in chapter 5, using the same acoustic features, the differences between the distributions of these features for different languages can be exploited for language ID with methods such as GMMs and SVMs. In our system we use GMM-UBM, GMM-SVM, GMM-SVM-GMM , GMM-*uni*-gram and GMM-*bi*-gram sub-systems. Fusing these five acoustic sub-systems together gives our acoustic fused language ID system (acoustic fused block in figure 6.2).

Our final fused language ID system is obtained by fusing all the phonotactic and acoustic sub-systems together (this is the 'acoustic-phonotactic fused' block in figure 6.2). The best fusion coefficients are obtained by linear logistic regression training using development data.

6.4 Summary

A single language ID system is less capable of providing robust and accurate performance than the fusion of multiple systems. Using different features (or a different tokenizer in the case of phonotactic systems) or modeling particular features with different approaches results in different language ID systems, which might lead to improved performance when fused together. The most popular fusing techniques for language ID were briefly presented in this chapter. A diagram (figure 6.2) depicting the different components of our language

ID systems, the fused phonotactic and acoustic sub-systems, and the overall fused system were also presented in this chapter.

Chapter 7

Speech Corpora

In this thesis, three different classification tasks are conducted: language ID, regional accent ID and ethnic group ID. The speech data used in training and evaluating those three classifiers are described in this chapter.

7.1 CallFriend and NIST data

7.1.1 CallFriend corpus

The CallFriend corpus[1] of telephone speech was collected by Linguistic Data Consortium (LDC)[2] in 1996, primarily to support projects on language recognition and was sponsored by the U.S. Department of Defense. There are fifteen languages and dialects (see table 7.1) with conversations lasting around 30 minutes. The CallFriend corpus consists of three sets; train, development and evaluation. Each set contains twenty two-sided conversations for each of the fifteen languages and dialects. English, Mandarin and Spanish have two dialects. All speakers were aware that they were being recorded but they were given no guidelines concerning what they should talk about. Once a caller was recruited to

[1]http://www.ldc.upenn.edu/catalog

[2]http://www.ldc.upenn.edu

participate, he or she was given a free choice of whom to call. Most participants called family members or close friends. Unless otherwise stated, both the train and development sets are used as training data for the language ID experiments in this thesis. The data of the two dialects of English, Mandarin and Spanish are combined and used to build one model for each of the corresponding languages. This means that each of these three languages has 80 conversations for training whereas the other nine languages each has only 40 conversations for training.

7.1.2 NIST 1996 data set

The NIST 1996 data set [3] is used as a development data set for our language ID systems. This data set consists of two subsets – development and evaluation sets consisting of the 12 languages shown in table 7.1, and 3, 10 and 30 second audio files. The development subset consists of approximately 1200 files for each of the three evaluation durations (3s,10s and 30s), with roughly 160 files each for English, Mandarin, and Spanish and 80 for each of the other nine languages. The evaluation subset consists of approximately 1500 files for each of the three durations: 480 for English, 160 each for Mandarin and Spanish, and 80 for each of the other nine languages. English messages were obtained from both the CallFriend corpus (160) and other English corpora (320) [34]. The NIST 1996 evaluation set (lid96e1) is used for fusing different language ID systems together, and NIST 1996 development set ('lid96d1') was used for estimating the parameters which are required for the Z-norm score normalization, and for tunning some other parameters.

[3]http://www.itl.nist.gov/iad/mig/tests/lre/1996

English (South and North USA)	Arabic	French	Farsi
German	Hindi	Japanese	Korean
Mandarin (Mainland and Taiwan)	Spanish (Caribean and non-Caribean)	Tamil	Vietnamese

Table 7.1: Twelve target languages with two dialects for three languages

7.1.3 NIST 2003 LRE data set

This data set [40] consists of 80 segments with duration of 3, 10 and 30 seconds in each of 12 target languages (table 7.1). This data comes from conversations collected for the CallFriend Corpus, described in section 7.1.1, but not included in its publicly released version. In addition, there are four additional sets of 80 segments of each duration selected from other LDC supplied conversational speech sources, namely Russian conversations of CallFriend type, Japanese conversations from the CallHome Corpus, English from the Switchboard-1 Corpus and cellular English from the Switchboard Cellular Corpus. The NIST 2003 30-second subset was mainly used to evaluate our language ID systems and to present comparative results for some useful techniques common in the language ID.

7.1.4 NIST 2005 LRE data set

The NIST 2005 LRE [4] data set contains test segments with three nominal durations of speech: 3 , 10 , and 30 seconds, from a set of 7 languages and two dialects (English-American, English-Indian, Hindi, Japanese, Korean, Mandarin-Mainland, Mandarin-

[4]NIST 2005 Language Recognition Evaluation: http://www.nist.gov/speech/tests/lang/2005

Taiwan, Spanish and Tamil) [5]. For the NIST 2005 language ID evaluation, we do not distinguish between English-American and English-Indian or Mandarin–Mainland and Mandarin-Taiwan. Actual speech durations varied, but were constrained to be within the ranges of 2-4 seconds, 7-13 seconds, and 25-35 seconds of actual speech contained in segments, respectively. However, only the 30 seconds subset was used to evaluate our language ID systems. This subset consists of about 3662 speech segments, with around 360 segments for each the target languages and dialects.

7.2 The "Accents of the British Isles" (ABI-1) corpus

The Accents of British Isles (ABI-1) speech corpus [24] was used in all of our regional accent recognition experiments reported in chapter 10. The ABI-1 speech recordings represent 13 different regional accents of the British Isles , plus standard British English. These were made on location in 13 different regions as shown in Figure 7.1.

Table 7.2 shows the fourteen regional accents and their abbreviations that will be used through out this thesis. In each case, twenty people were recorded (normally ten women and ten men) who were born in the region and had lived there for all of their lives. The standard southern English speakers were selected by a phonetician. Each subject read twenty prompt texts, ranging from 'task oriented' texts which are representative of generic applications of automatic speech recognition, to 'phonetic' texts chosen for their phonetic content. The later includes the "Sailor Passage" [105], which was split into three

[5] All of these languages and dialects are subset of the CallFriend languages and dialects except Indian English

parts of approximately equal length. The first section, referred to as SPA (Sailor Passage A) comprised ninety-two words and its recordings varied between 30 and 45 seconds in duration. The ABI-1 corpus also includes recordings of a list of /hVd/ syllables, where each syllable begins with 'h' and ends with 'd' and the vowel 'V' varies. This list of syllables, referred to as "Careful Words", was read five times by the participants. Word-level transcriptions, aligned at the sentence level, are available for all of the ABI-1 recordings.

Accent	**Abbrev.**	**Accent**	**Abbrev.**
Birmingham	brm	Liverpool	lvp
Truro (Cornwall)	crn	Newcastle	ncl
Lowestoft (East Anglia)	ean	Denbigh (North Wales)	nwa
Hull (East Yorkshire)	eyk	Dublin (Republic of Ireland)	roi
Glasgow	gla	Elgin (Scottish Highlands)	shl
Inner London	ilo	Standard Southern English	sse
Burnley (Lancashire)	lan	Belfast (Ulster)	uls

Table 7.2: Accents of the ABI corpus and corresponding abbreviations

The ABI-1 recordings were made using head mounted and desk microphones, and sampled at 22.05kHz. For the accent recognition experiments reported here, the head-mounted microphone recordings were bandpass filtered (0.23 - 3.4 KHz) to simulate a telephone channel, and downsampled to 8KHz. The speakers were divided into three subsets; two with 93 and one with 94 speakers. Gender and accent were distributed equally in each subset. A "jackknife" training procedure was used in which two subsets were used for training and the remaining subset for testing. This procedure was repeated

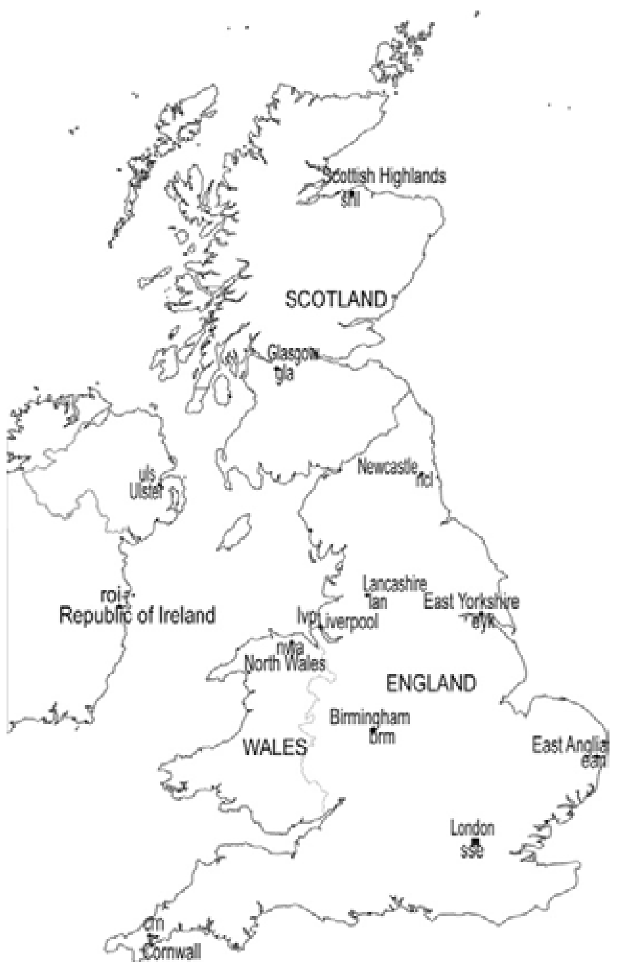


Figure 7.1: ABI regional accents of British Isles

three times with different training and test sets, so that each ABI-1 speaker was used for testing, and no speaker appeared simultaneously in the training and test sets.

Two separate evaluations of the text-independent accent recognition systems were conducted, one using 30-seconds extracts from all test recordings, and the other using the SPA utterances. The first test set, 1504 30-second cuts from all speakers in the ABI-1 test sets, was used to enable comparison with standard language identification performance on the NIST 2003 and 2005 evaluation sets (where we also used 30s test utterances). The second test set, comprising approximately 280 SPA utterances, was used to evaluate and compare the text-independent and text-dependent automatic systems, and human listeners on the accent identification task.

7.3 The "Voices across Birmingham" (VaB) corpus

The "Voices across Birmingham" (VaB) corpus was used in the "ethnic group" recognition experiments reported in chapter 11. The goal of the VaB project is to capture variations in conversational speech across the people of the city of Birmingham in the UK. It currently comprises approximately 175 hours of recordings of telephone conversational speech between participants who were born in or around the city. Each participant made up to one hour of free telephone calls, which were routed through an Aculab Prosody X telephony card for automatic recording. Both participants in the call were aware that they were being recorded and of the purpose of the recording.

Significant immigration into Birmingham from Asia began in the 1960s. According to the 2001 census of England and Wales, which included questions about the ethnicity of residents[6], approximately 70.4% of Birmingham's population categorized themselves as 'White' and 19.5% as 'Asian'. Twenty nine percent of Birmingham's British Asian population gave their ethnicity as Indian, 53% as Pakistani, 11% as Bangladeshi and the remaining 7% as 'Other Asian'. The VaB project asked its participants similar questions about ethnicity. For the 'White' and 'Asian' groups there is sufficient data to conduct an experiment to study whether or not an individual can be classified automatically into the correct ethnic group from his or her speech (the VaB corpus does not distinguish between the different ethnic subgroups within the Asian community).

The Asian group can be further sub-divided into those subjects who were born in

[6]2001 Population Census in Birmingham (http://www.birmingham.gov.uk/community)

Birmingham (second generation) and those who were not. Only recordings from White and second generation Asian participants were included in the current experiments. The recordings from these two groups were divided into training and test sets. The training set consists of recordings from 242 different speakers (165 Asian and 77 White). The test set consists of 315 utterances from different speakers, each with maximum duration of 40 seconds. Of these, 175 are Asian (69 male, 106 female) and 140 are White (53 male, 87 female).

All of the speech corpora described above are summarized in table 7.3.

Corpus	CallFriend	NIST LRE 1996	NIS LRE2003	NIS LRE2005	ABI-1	VaB
Style	Conv.	Conv.	Conv.	Conv.	Read	Conv.
Channel	Telephone	Telephone	Telephone	Telephone	Head mic	Telephone
Sample rate	8kHz	8kHz	8kHz (resamp. 8kHz)	8kHz	22.05kHz	8kHz
Classes	12 languages	12 languages	12 British Eng.	7 groups	14 accents	2 ethnic
Use	LID training	LID development	LID testing testing	LID testing & testing	Accent ID training &	Ethnic group ID training

Table 7.3: Summary of speech corpora used in the studies (Conv. = conversational speech).

7.4 Summary

All of the speech data used in this thesis was described in this chapter. The CallFriend corpus consists of telephone conversations collected by LDC for twelve languages. The training and development sets were used as training data for all of our language ID systems presented in this thesis. The NIST 1996 data set was used for as development data for our language ID systems, e.g. fusing different language ID systems together. The NIST 2003 and 2005 evaluation data sets were used to develop and evaluate our language ID systems.

The ABI-1 corpus is a read speech recordings collected from 13 different regions in the British Isles plus standard British English. All of the recordings, with their word level transcriptions, were used for all of our regional accent recognition experiments in this thesis. Because the amount of ABI-1 speech data is not sufficient to be divided into training, development and testing sets, three rounds of 'jackknife' procedure was used to overcome this limitation.

The VaB corpus consists of around 175 hours of recordings of telephone conversational speech between participants who were born in or around the city Birmingham in the UK. The participants were asked to indicate their ethnicity. All recordings from the two main ethnic groups in the city of Birmingham, 'White' and second generation 'Asian', were used for all of our ethnic group ID experiments.

Chapter 8

Language Recognition Experiments

After evaluations in 1996, NIST LRE 2003 was the second evaluation NIST conducted to establish a current baseline of performance capability for language recognition of conversational telephone speech [40]. The NIST LRE 2003 30-second subset was used to develop our language ID systems and to study the effectiveness of different features and techniques for language ID. Although NIST has released evaluation plans for 2005, 2007, 2009 and 2011, we used the NIST 2005 LRE data set to evaluate our language ID systems because the main aim of this work is to study how well the standard techniques used in language ID can be extended and used for recognizing the regional accents of British Isles and the two main ethnic groups in the city of Birmingham which are described in chapters 10 and 11.

This chapter describes the details of our language ID system and presents our results of the NIST 2003 and 2005 language recognition evaluations.

8.1 Language ID systems

8.1.1 Phonotactic systems

Our phonotactic language ID system uses three existing phone recognizers (Czech, Hungarian and Russian), borrowed from a toolkit developed by Brno University of Technology [1] as tokenizers. These phone recognizers are used because it has been shown that they provide good performance for language ID systems and they are available at no cost. These phone recognizers were trained on the SpeechDat-E databases using a hybrid approach based on Neural Networks and Viterbi decoding [106]. More details about these phone recognizers can be found in [106].

For each of these phone recognizers and each target language, 1-gram, 2-gram, 3-gram and 4-gram LMs were built with SVMs, as described in chapter 4. This results in twelve phonotactic language ID subsystems (3 Tokenizers $\times$ 4 n-gram LMs). The final dimensions, after removing the zero occupancy components, of n-gram probability vectors of each subsystem are shown in table 8.1. For each order of n-gram ($n = 1, \ldots, 4$), the three phone recognizer dependent subsystems are fused together using LLR fusion technique, which is described in chapter 6, trained on the NIST 1996 evaluation subset (see chapter 7). Moreover, the twelve phonotactic systems are fused together in the same way, and the result is our phonotactic language ID system.

The performance (EER[%]) of each individual subsystem on the NIST 2003 30-second data set is reported in table 8.2, together with the performance of the fused systems.

The results in table 8.2 show that the phonotactic systems with the Russian phone

[1] www.fit.vutbr.cz/research/groups/speech/sw/phnrec/

phone recognizer \ n-gram	1-gram	2-gram	3-gram	4-gram
Czech	42	1764	44377	422666
Hungarian	58	3364	76939	903069
Russian	52	2704	75844	895428

Table 8.1: Dimensions of n-gram vectors of each phone recognizer.

PRLM system	*uni*-gram	*bi*-gram	*tri*-gram	4-gram	ALL
Czech	14.7	6.3	3.4	4.2	
Hungarian	13.4	5.4	2.9	3.8	
Russian	10.8	4.5	2.8	3.1	
Fusion	7.8	2.6	**1.6**	2	**1.48**

Table 8.2: EER [%] of single n-gram ($n = 1, \ldots, 4$) PRLMs and PPRLM on NIST 2003 LRE 30s evaluation.

recognizer outperform the other systems for each order of n-gram. It is also shown that by increasing the order of the n-grams from 1 to 3, the performances of all systems improve dramatically. However, the performance of all systems decreases with the 4-gram. A possible explanation for this degradation is that in the case of 4-grams, more training data is needed than is available. Another interesting observation from the results in table 8.2 is that combining several phonotactic systems with different phone recognizers outperforms all of the individual systems.

Table 8.3 gives a comparison of our results with the results of the best systems known from the literature on NIST 2003 data. The Massachusetts Institute of Technology (MIT) PPRLM system [34] (row 2) employs HMM phone recognizers trained on 6 languages from OGI stories [107], and the result for the Brno University of Technology (BUT) PPRLM

System	NIST 1996 eval. data	NIST 2003 eval. data
PPRLM MIT[34]	5.6	6.6
PPRLM BUT [106]	1.48	2.42
Our PPRLM	1.36	1.48

Table 8.3: Comparison of EER [%] on NIST 1996 and 2003 evaluations

system [106] (row 3) uses phone recognizers trained on four languages from SpeechDat-E data (the same phone recognizers which we use in our experiments plus a Polish phone recognizer).

Our best results (EER of 1.36% and 1.48% in table 8.3) are obtained by fusing 12 PRLM systems together. This result favorably compares to the PPRLM BUT system. In spite of the fact that our system and the BUT system use the same phone recognizers, the superiority of our system, probably, comes from the discriminative weighting technique applied to the n-gram probability vectors during training and evaluation. In addition, we use more data to train our systems.

We conclude that weighting the n-gram probability vectors with the LLR discriminative technique is very useful for phonotactic language ID, because it emphasise the n-gram components which have the most discriminative information to distinguish between languages and de-emphasise those which do not. We also conclude that fusing multiple phonotactic systems with different phone recognizers improves the performance of the language ID system.

8.1.2 GMM-UBM acoustic system

8.1.2.1 Acoustic features

The most widely used features for language recognition are MFCCs. The extraction process of MFCCs from a speech signal with different spectral analysis algorithms was described earlier in chapter 3. The DFT based spectral analysis algorithm is used to estimate the MFCCs which are used in all experiments described in this chapter.

In order to find the best configurations (parameters) of the front end analysis and the best normalization techniques for our language ID system, several experiments were conducted with our GMM-UBM systems (with 512 and 2048 GMM components) using only the training set of the CallFriend corpus described in chapter 7. Silence frames are labeled using energy based VAD and then discarded after calculating the temporal coefficients.

For the *MFCC35* and *MFCC36* experiments (row 2 and 3 of table 8.4), the feature vectors comprise 12 static MFCCs plus the corresponding delta, Δ, and second delta, $\Delta\Delta$, parameters computed with 5 frame windows. The final coefficients exclude or include c_0, resulting in 35 or 36 features. The purpose of these two experiments is to study the influence of keeping or removing c_0 from the cepstral features. It is evident from the results in table 8.4 that keeping c_0 improves the EER [%] of the GMM-UBM language ID system with 512 and 2048 components by around 5% and 2.7%, respectively. The *MFCC57* features are the same as *MFCC36* but they are calculated on 19 static MFCCs plus the corresponding Δ and $\Delta\Delta$ parameters. The result of this experiment (row 4, table 8.4) shows that increasing the number of MFCCs (and the corresponding temporal information) from 12 to 19 slightly improves the system performance.

Feature	c_0	# of Gauss	
		512	2048
MFCC35	–	29.4	25.9
MFCC36	$\surd$	28	25.2
MFCC57	$\surd$	27.6	24.8
SDC49	$\surd$	22.3	18.3
MFCC7+SDC49	$\surd$	20.5	17.4
MFCC12+SDC49	$\surd$	19.7	17
MFCC19+SDC49	$\surd$	**19.6**	**16.7**

Table 8.4: EER [%] comparison for different kind of features with the GMM-UBM language ID system

In order to compare the conventional method of extracting the temporal information (i.e. Δ and $\Delta\Delta$) with the SDC method, several language ID experiments were carried out with different combinations of static MFCCs and the corresponding temporal features calculated with the two methods. The SDC coefficients are calculated as described in chapter 3, with $N - d - P - k$ parameters. The same parameters reported in [57], $7 - 1 - 3 - 7$, were found to be the best for our system, and therefore were used in calculating SDC coefficients in all our experiments. These parameters result in a 49-dimensional temporal feature vectors. Results of language ID are usually reported only with SDC features, which do not include static coefficients. We studied the influence of adding different number of static coefficients; 7, 12 and 19, to the SDC features. As is shown from the results in table 8.4, adding the 19 MFCCs, including c_0, to the 49-

dimensional SDC coefficients (*MFCC19+SDC49*), which results in 68-dimensional feature vectors, gives the best performance. Therefore, *MFCC19+SDC49* features are used in all of our subsequent experiments unless otherwise stated.

8.1.2.2 Gender Variability

The gender of the speaker is one of the sources of speaker variables that degrade the performance of language ID systems. Several methods are used to normalize the gender variability of the speakers. The most common methods are the Vocal Tract Length Normalization (VTLN) [92] and building a gender identification systems followed by gender-dependent language models. We used the latter method in all of our language ID experiments reported in this chapter. This method is chosen because it is simple and it does not require additional computation. In the gender identification system, two background models (UBM) are trained; one on the conversations of male speakers and one on those of female speakers of all languages. The conversations of male and female speakers in the training data of each target language are then used to MAP adapt means and weights of the corresponding background model, producing two gender and language dependent GMMs for each language. This results in two UBMs and 24 GMMs for our 12 target languages in the CallFriend corpus. In recognition, the two gender-dependent UBMs are used to identify the gender of the speaker first, then the corresponding language dependent models are selected for language recognition.

In order to study the effect of gender variability on language ID, the experiment with the best result in the previous section (row 8 and column 4 in table 8.4) was repeated but with gender identification and gender-dependent models (i.e. two UBMs and 24 language-dependent GMMs). The result of this experiment (EER = 14.3%) shows that by removing the gender variability of the speakers, the performance of our GMM-UBM language ID system (with 2048 components) improves by around 14%. This means that modeling languages for each gender separately avoids the gender variability and improves the language ID performance. Therefore, gender identification and gender-dependent models are used in all of our subsequent experiments on language ID.

8.1.2.3 Feature normalization

In all of our previous results reported in this chapter, the speech frames (after removing the silence frames with VAD) undergo channel normalization using CMN technique. As explained earlier in chapter 3, there are other feature normalization techniques which have been successfully used in language ID, such as RASTA filtering, MVN and feature warping. To study the effect of these techniques on our language ID system, several experiments were carried out with our GMM-UBM system, with 2048 Gaussian components and 68-dimensional feature vectors,(*MFCC19+SDC49*).

The results of these experiments are reported in table 8.5. It shows the performance comparison of the four feature normalization techniques. Normalizing the cepstral features with CMN reduces the EER percentage dramatically by 45%. It is also shown that RASTA filtering has a slight improvement over CMN because it not only normalizes the mean but also filters the high frequency modulation within each component, which may contain some noise. By normalizing the feature distribution with MVN and feature warping, the performance of language ID system improves by 53% and 54%, respectively corresponding to the system result with un-normalized features. In addition to mean and

Features	CMS	RASTA	MVN	Feature Warping	# of Gauss 2048
MFCC19-SDC49	–	–	–	–	26
	√	–	–	–	14.3
	–	√	–	–	14
	–	–	√	–	12.3
	–	–	–	√	11.88
	–	√	√	–	11.7
	–	√	–	√	**11.37**

Table 8.5: EER [%] comparison of different kind of feature normalization techniques on CallFriend training set

variance, feature warping normalizes the skewness and flatness of the feature distribution over a short period (3 seconds in our case), which improves the language ID performance slightly more than MVN. By combining this with RASTA to filter out the high frequency components, a further improvement is obtained with both MVN and feature warping. The best performance (11.37% EER) is obtained by combining RASTA and feature warping, therefore and unless otherwise mentioned, this combination are used in all of our experiments.

8.1.2.4 Score normalization

In order to study the influence of score normalization techniques on language ID, the same language ID system (row 7, table 8.5) was re-evaluated using different combinations of Z-norm, T-norm, LLR and 'max-LogLikelihood' score normalization techniques described

Score Normalization	EER[%]
–	11.37
Z-norm	10.82
T-norm	10.68
ZT-norm	10.68
LLR norm	9.27
max-loglikelihood	8.35

Table 8.6: comparison of different score normalization techniques on GMM-UBM with 2048 components LID system

earlier in chapter 5. The NIST 1996 development data set ('Lid96d1') was used to estimate the Z-norm parameters for each language. Table 8.6 shows the performances of the language ID system using these score normalization techniques. These results show that 'max-loglikelihood' score normalization outperforms the other normalization techniques by improving the system performance 27% relative to the performance without score normalization. A possible explanation for this is that in 'max-loglikehood' we normalize the score of every model with the score of the most competitive (impostor) model, whereas in the other normalization techniques we normalize with the average of competitor impostor scores. Unless stated otherwise, 'max-loglikelihood' score normalization is used in all of the subsequent experiments in this thesis.

8.1.2.5 Increasing the number of GMM components and amount of training data

Training GMM with the EM algorithm requires specification of the number of Gaussian components prior to the training process. Most GMM-based language ID systems use 2048 Gaussian components. Increasing the number of GMM components requires more training data and more computation. However, in order to study the influence of increasing the number of GMM components on the performance of language ID, a GMM-UBM system with different number of Gaussian components (512, 1024, 2048 and 4096) are trained on the train set of the CallFriend corpus and evaluated on the NIST 2003 30-sec segments. Figure 8.1 shows the DET curves for all of these language ID systems. This figure shows that by increasing the number of Gaussian components, the performance of language ID system improves up to 2048 components. We also observe from the DET curves that there is no improvement by increasing the number of components to 4096. A possible reason is that the amount of training data is not enough to train a high order GMM with 4096 components. To verify this and to study the influence of the amount of training data on the system performance, both the training and development data sets of CallFriend corpus were used (i.e. doubling training data) to re-train and re-evaluate the language ID systems with 2048 and 4096 GMM components. The performances of these two systems are shown in figure 8.2. As shown in this figure, doubling the amount of training data improved the performance by about 30% (EER = 5.75%) when using 2048 Gaussian components compared with 38% (EER=5.19%) improvement when using 4096 Gaussian components . This confirms that increasing number of GMM components improves the language ID performance but requires more training data.

The best performance of our GMM-UBM language ID system (EER = 5.19%) is obtained by using 4096 GMM components trained on both the training and development data sets of the CallFriend corpus. This language ID system will be used as a baseline for the other systems and other techniques described later in this chapter.

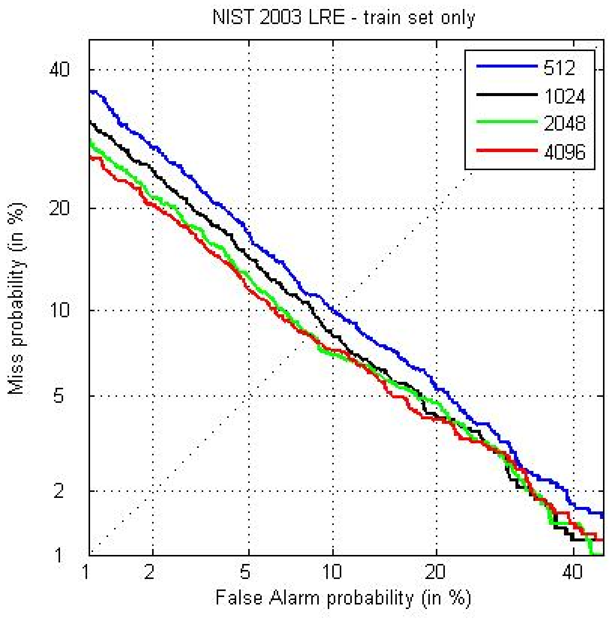


Figure 8.1: Performance (DET curves) of GMM based LID system with different number of Gaussian components trained on Callfriend train set and evaluated on NIST 2003 LRE 30s segments

8.1.3 GMM-SVM and GMM-SVM-GMM acoustic systems

The best front-end parameters and normalization techniques for language ID which have been found earlier in the previous sections with our GMM-UBM system, were also used in our GMM-SVM and GMM-SVM-GMM systems, which were described earlier in chapter 5. The same UBMs used for our GMM-UBM systems of 2048 and 4096 components were also used to map each utterance in the training and development data sets of CallFriend

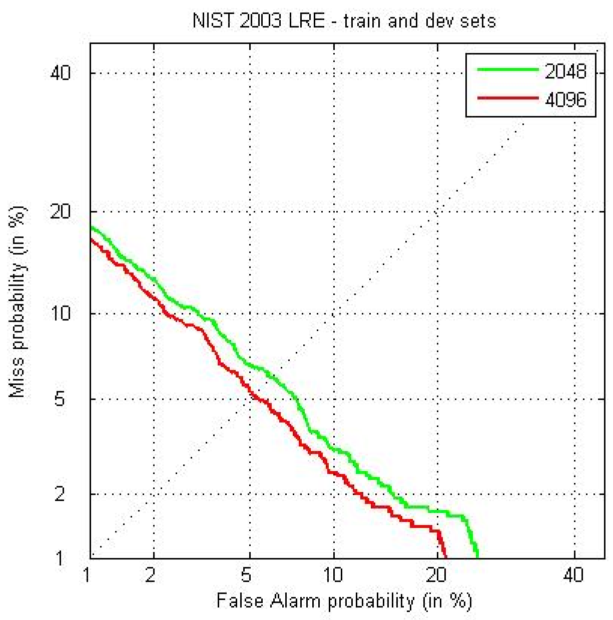


Figure 8.2: Performance (DET curves) of GMM based LID system with 2048 and 4096 Gaussian components trained on Callfriend train and dev sets and evaluated on NIST 2003 LRE 30s segments

corpus to supervector space in the same way described in chapter 5. For each GMM size, 2048 and 4096, two gender-dependent SVM models for each language were trained on the resulting supervectors using 'one-against-others' strategy. Evaluated on NIST LRE 2003 30-sec subset, the performance (EER[%]) of our GMM-UBM, GMM-SVM and GMM-SVM-GMM language ID systems are reported in table 8.7. These results confirm that in perspective of language ID, the discriminative techniques such as SVMs are more effective than the probabilistic modeling techniques such as GMMs. Our GMM-SVM language ID systems with 2048 and 4096 components outperform the corresponding GMM-UBM systems by around 74% and 73% respectively. A further 10% and 9.2% improvements are achieved with our GMM-SVM-GMM systems, with 2048 and 4096 components respectively.

System	# of Gauss	
	2048	4096
GMM-UBM	5.75	5.19
GMM-SVM	1.5	1.41
GMM-SVM-GMM	**1.35**	**1.28**

Table 8.7: Performance (EER [%]) of GMM-UBM, GMM-SVM and GMM-SVM-GMM language ID systems on NIST 2003 LRE 30s segments

8.1.4 Inter-Session Compensation (ISC) for acoustic systems

In order to apply ISC technique, which was described earlier in chapter 5, to our acoustic language ID systems, first we need to estimate the eigen-channel subspace U matrix which represents the inter-session variability in the supervector space. The same supervectors which were used for our GMM-SVM system, are also used to estimate U matrix. The second requirement for ISC is to estimate inter-session variability in each utterance (training and testing) in a low dimensional space (i.e. X_c), in the same way described in chapter 5.

The ISC technique can be applied to the GMM model domain or to the acoustic feature domain or both of them at the same time. Therefore, ISC at model domain can be applied to GMM-UBM and GMM-SVM-GMM systems only, whereas ISC at feature domain can be applied to all systems.

Before we start using ISC with our language ID systems, we need to determine the best number of eigenvectors in U matrix (i.e. the dimension of X_c vectors) which represents the number of removed dimensions. A possible way to find it practically is by evaluating our GMM-UBM system with 4096 components by applying ISC using different number of eigenvectors ranging from 50 to 600.

Figure 8.3 shows the performances of our GMM-UBM system evaluated on the NIST 2003 30-sec data with the ISC technique (applied in model domain) using different number of eigenvectors. This figure shows that compensating for the inter-session variability decreased the EER from 5.19%, when no ISC is applied, to 3.6%, when ISC is applied with only 50 eigenvectors. The best performance, EER = 3.32%, was found when representing the inter-session variability by its major 400 dimensions. Therefore and unless otherwise stated, 400 eigenvectors were used in all of our subsequent experiments which includes ISC technique.

The performances of all our acoustic systems with ISC in the model and feature domains are reported in table 8.8. The general observation from these results that normalizing the inter-session variability improves the performance of the three acoustic language ID systems, GMM-UBM, GMM-SVM and GMM-SVM-GMM dramatically. As expected, applying ISC technique at the acoustic features level is better than in the model domain because in the earlier method the inter-session variability is normalized in both the training and the testing acoustic features, whereas, this variability is normalized in the testing features only with ISC in the model domain. Another interesting observation is that a further improvement on the GMM-UBM and GMM-SVM-GMM systems was obtained by combining both ISC in both features and model domains. This combination is achieved by applying the ISC at the feature level using uncompensated models and then re-training the acoustic models on the compensated features. The ISC is then applied in the model domain to compensate the new models. The best performance (EER=0.5%) was obtained by our GMM-SVM-GMM system with 4096 components and with applying the ISC in the feature and model domains.

We believe that the best result on the NIST 2003 30-sec evaluation set published in 2003 was 2.8% EER [34] and that the overall best performance since then is 0.8% EER [35]. This is comparable with our own best result (0.5%EER) and establishes the credibility of our system.

However, this improvement in performance of acoustic language ID systems was at the cost of computation time. This powerful technique (ISC), especially at the feature level, required a huge computation which takes a very long time.

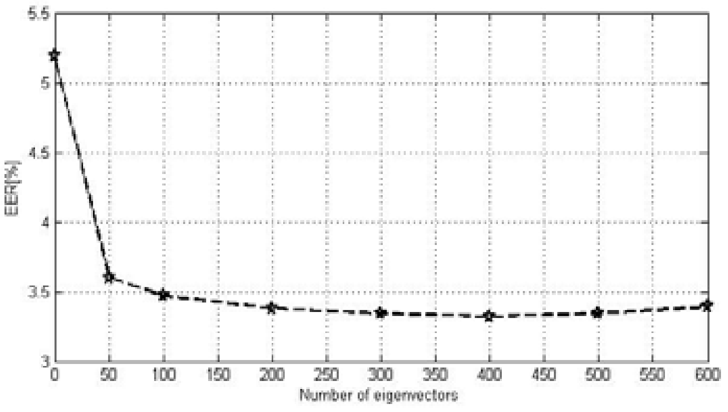


Figure 8.3: Variation of EER of GMM-UBM system with ISC technique using different number of eigenvectors

8.1.5 GMM-n-gram language ID system

Our GMM-n-gram language ID system is the same as the phonotactic system, except a GMM is used as a tokenizer instead of phone recognizer, as described earlier in chapter 5. Twelve language dependent GMMs, with 2048 components, were trained on the language

System	ISC-Feature Domain	ISC-Model Domain	# of Gauss	
			2048	4096
GMM-UBM	–	–	5.75	5.19
	–	√	3.52	3.32
	√	–	1.5	1.5
	√	√	1.4	1.33
GMM-SVM	–	–	1.5	1.41
	√	–	1.25	1.2
GMM-SVM-GMM	–	–	1.35	1.28
	–	√	1.08	1.0
	√	–	0.84	0.55
	√	√	**0.837**	**0.5**

Table 8.8: EER [%] of acoustic language ID systems with the ISC using 400 dimensions to represent the inter-session variability.

specific data, and then used to build our GMM-*uni*-gram system in two different ways:

- First, each language-dependent GMM was used as a tokenizer for all languages and then the twelve *uni*-gram systems are combined at the back-end (similar to parallel PRLM with different phone recognizers) (this is the 'Parallel lang-dept GMMs-*uni*-gram system in table 8.9).

- Second, the twelve language-dependent GMMs are concatenated together to form our proposed MLM language-independent model with 24,576 GMM components (this is the MLM-*uni*-gram system in table 8.9).

The the same UBM which was trained on the whole training data and used for our GMM-UBM and GMM-SVM systems, was also used as a tokenizer for our GMM-*bi*-gram system (system *bi*-gram in table 8.9). The MLM-*uni*-gram and GMM-*bi*-gram systems were fused together with the GMM-UBM and GMM-SVM-GMM acoustic systems to get the 'acoustic-fused' system. All of the acoustic systems were also fused with the phonotactic systems to get our final fused language ID system ('phono-acoustic-fused' in table 8.11).

Moreover, to investigate the effectiveness of the ISC technique on the GMM tokenization systems, all of our GMM tokenization language ID systems described in this section were re-trained and re-evaluated on the compensated feature vectors, as shown in the third column of table 8.9).

System	EER[%]	
	Un-compensated features	Compensated features (ISC)
Parallel lang-dept GMM-*uni*-gram (12×2048)	3.87	3.34
MLM-*uni*-gram (24,576)	1.66	1.34
GMM-*bi*-gram	3.51	3.18

Table 8.9: Performance (EER[%]) of GMM-*n*-gram language ID systems on the NIST 2003 LRE 30sec subset.

The results in table 8.9 show that concatenating the twelve language dependent GMMs into a high order MLM and using it as a tokenizer outperforms the parallel language-dependent GMM tokenizers. The reason for this superiority is that, with high number of GMM components in our MLM, the language specific features are modeled in separate components, which allows emphasizing them with the LLR weights in a better way.

To study the difference between the traditional UBM and MLM, a 6144-component background model was built in three ways: A traditional UBM model (table 8.10, column 2), a concatenation of 12 language-dependent 512-component GMMs built separately for each language (table 8.10, column 2), and a concatenation of 12 language-dependent 512-component GMMs MAP adapted from a 512-component UBM (table 8.10, column 4). Each background model was used in two different systems: A GMM-UBM system, and a discriminative GMM-*uni*-gram system (rows 2 and 3 in table 8.10, respectively).

System/Background model	UBM	MLM	MLM-adapt
GMM-UBM	5.4	5.8	6.6
GMM-*uni*-gram	2.7	2.3	3.6

Table 8.10: Performance [EER%] of the GMM-UBM and GMM-*uni*-gram systems using background model built in three different ways.

It is clear from the results that the MLM background model is advantageous for the GMM-*uni*-gram system but not for the probabilistic GMM-UBM system. A possible explanation is that the areas of interest of the two systems in acoustic space are different. The discriminative *n*-gram systems focus on the language-specific boundaries of the background model, where use of a component is indicative of a particular language. By contrast, the probabilistic GMM-UBM system relies on differences in the probabilities from components of the language specific GMMs which arise from MAP adaptation of the same components from the cross-language middle of the background model. The smaller traditional UBM appears to result in more reliable and robust Gaussian probabilities, but has fewer language-specific components that can be exploited by the *uni*-gram model, whereas the larger MLM method has enough space to accommodate the language specific components that the *uni*-gram model requires. Thus, the MLM is more biased towards the language specific components than the traditional UBM. This is useful for the discriminative approaches but not for the generative approaches.

8.2 Fusion

Table 8.11 shows the results of our fused language ID systems. The 'phonotactic-fused' results was obtained by fusing the 12 phonotactic sub-systems together. The 'Acoustic-fused' was obtained by fusing the four acoustic sub-systems: GMM-UBM, GMM-SVM-GMM, MLM-*uni*-gram and GMM-*bi*-gram. The best result (0.34% EER) was obtained by our overall fused system, 'phono-acoustic-fused', which is fusion of 16 sub-systems (12 phonotactics plus 4 acoustics). In all fused systems, the NIST 1996 evaluation set was used as training data to optimize the fusion coefficients with the LLR fusion technique (described in chapter 6).

According to the results presented in this chapter, the configurations of our final language ID system can be summarized as follow:

- Three phone recognizers (Czech, Hungarian and Russian) are used as tokenizers for the phonotactic systems.
- Four order of n-grams (1-gram, 2-gram, 3-gram and 4-gram) are used to build language models for phonotactic systems, whereas only 1-gram and 2-gram are

System	EER[%]	
	Un-compensated features	Compensated features (ISC)
phonotactic-fused	1.48	1.48
Acoustic-fused	0.93	0.41
Phono-acoustic-fused	0.7	0.34

Table 8.11: Performance (EER[%]) of fused language ID systems on the NIST 2003 LRE 30sec subset.

used for the GMM-n-gram systems.

- Hamming window of 20ms and frame rate of 100 frames per second
- 23 triangular windows with a 3.5 KHz bandwidth
- RASTA filtering in the log power spectral domain
- Acoustic features are 19 MFCCs plus 49 SDCs (68 features).
- Silence frames are labeled and removed with the energy-based VAD
- Feature warping technique is applied for the final feature vectors (after removing silence)
- Gender-dependent models
- ISC technique is applied in the feature and model domains
- A UBM of 4096 components is used for our GMM-UBM, GMM-SVM, GMM-SVM-GMM and GMM-*bi*-gram systems

- MLM of twelve language-dependent GMMs with 2048 components (24,576) is used for the GMM-*uni*-gram system
- Scores are normalized with the max-loglikelihood.
- Linear logistic regression is used for fusion.

8.3 NIST 2005 LRE

In this section we evaluate our language ID system, which was developed in the previous sections using the NIST 2003 evaluation data, on the NIST 2005 30s evaluation data. Since the seven target languages (with three dialects) in NIST 2005 LRE are a subset of the twelve languages (with three dialects) in the CallFriend corpus (except the Indian English dialect), the same systems trained for the NIST 2003 LRE were also used for the NIST 2005 evaluation. Only the language-dependent models of the seven target languages are used for the NIST 2005 language recognition evaluation. The NIST 1996 evaluation data set was used to train the weighting coefficients for fusion.

Table 8.12 shows the performances of the main components of our language ID system and the overall fused system ('Phono-acoustic-fused') on the NIST 2005 data. As it is shown from the results, the performance of each of the acoustic systems is poor compared with the NIST 2003 evaluation. The phonotactic system (5.2% EER) outperforms all of the acoustic systems. The best performance achieved with our language ID system is 4.2% EER , which is obtained by fusing all of the acoustic and the phonotactic sub-systems.

To compare our results with the best results published on the NIST 2005 LRE, table 8.12 includes the results published by the MIT and BUT. The MIT fused system

language ID Systems	NIST 2005 LRE (EER[%])
GMM-UBM	13.1
GMM-SVM	11.75
GMM-SVM-GMM	9.14
MLM-*uni*-gram	11.9
GMM-*bi*-gram	18.2
Phonotactic-fused	5.2
Acoustic-fused	8.4
Phono-acoustic-fused	4.2
MIT fused system	3.3
BUT fused system	2.9

Table 8.12: Performance [EER%] of different language ID systems on NIST 2005 30-second subset.

(row 10) was obtained by fusing two phonotactic systems with a GMM-based acoustic system [70]. The two phonotactic systems are parallel of phone recognizers followed by language models classifiers using phone lattices and parallel phone recognizers followed by binary language models. The result of the BUT fused system (row 11) was obtained by fusing phonotactic and acoustic systems [108], where the phonotactic system is very similar to our system (same phone recognizers) except they used phone lattice and anti-model technique. The BUT acoustic system is GMMs trained with MMI discriminative training. As shown in the results, these two systems outperform our fused system. Both systems, MIT and BUT, used speech data for Indian English from OGI corpus because the CallFriend corpus does not contain Indian English speech, whereas our system was trained on the CallFriend corpus only. This difference in the training data could be a reason for the the relatively poor performance of our system.

The main results for all systems and tasks described in this thesis are also summarized in table A.1 in appendix A.

8.4 Summary and challenges

In this chapter our phonotactic and acoustic language ID subsystems were developed and evaluated on the NIST 2003 LRE data set. The results of the phonotactic systems showed that training the LMs on the phone sequences produced by different phone recognizers with SVMs and using the LLR discriminative weighting achieves results very comparable with the published results on the same data and using the same phone recognizers. The best result of the phonotactic systems was obtained by combining multiple systems with different phone recognizers and different orders of n-gram.

With our acoustic systems, we have verified that the results and conclusions of other labs obtained with MFCC and SDC features are valid and that these features are good for our language ID system. In addition, we have examined the importance of normalizing the acoustic features and the output scores for language ID. The performance of our GMM-UBM system improved by increasing the number of GMM components but this requires more training data. However, the cost of this improvement in the system performance is at the computation time. For instance, training our GMM-UBM system with 512 Gaussian components and 4 EM iterations on a PC with Quad core (Q8200) CPU running at 2.33 GHz and has 4 GB memory, required about 130 hours ($\sim$ 5 days) compared with 195 hours and 300 hours for the 2048 and 4096 Gaussian components, respectively.

The discriminative training with SVMs clearly outperforms the widely used GMMs for language ID. The ISC technique in the model domain and the acoustic feature domain is greatly beneficial for language ID. Moreover, applying this technique in the feature domain has a great advantage, as it allows to apply SVM parameter re-estimation without modifying the training algorithm. However, the ISC technique, particularly in the feature domain, requires a massive computation resources. For example, compensating the feature vectors of all of the conversations in the train and development sets of CallFriend corpus using a GMM with 512 components, took around 21 days on the same PC described earlier. The requirement of this huge computation urged us to divert our research a little bit towards optimization and parallel processing in order to speedup the computation. Optimizing and accelerating GMM and ISC computation on CPU and on multi-core GPU will be addressed in chapter 9.

It has been also shown in this chapter that methods normally applied to sequences of high-level units such as phones or words can be successfully applied to sequences of GMM components. A GMM-*uni*-gram system works surprisingly well, provided that discriminative weighting is applied to the *uni*-gram probabilities. The MLM has been proposed as an alternative to both conventional UBM and parallel of language-dependent GMMs. The MLM appears to have more language-specific components than a UBM, and for this reason works particularly well as the basis of a *uni*-gram system (and potentially as the basis of an *n*-gram system), but less well in a conventional probabilistic GMM-UBM system. The best performance is obtained by fusing the outputs of conventional phonotactic, GMM-UBM and GMM-SVM-GMM systems with those of a discriminatively weighted *uni*-gram system based on a 24,576 components MLM and a 4096 components

UBM *bi*-gram system.

The NIST 2005 data set was used only for evaluating the components of our language ID system. The results of our NIST 2005 language recognition evaluations are comparable with the published results on the same data. This gives credibility to our language ID system, which will be applied to accent and ethnic group recognition later in chapters 10 and 11, respectively.

Chapter 9

Language Recognition Using Multi-core Processors

In this chapter we investigate the potential impact of the GPU on signal processing by examining the impact of GPUs on the problem of probability calculation and speech-spectrum estimation in the context of automatic language ID. Speech processing applications, such as language ID, typically comprise several algorithmic units of which two normally absorb the majority of the computer's resources. These are spectral analysis and the calculation of the conditional probability of the input speech given some pre-trained acoustic model, typically a GMM.

In the first part of this chapter we consider the use of GPU for GMM probability caculation. Several papers [30, 31, 32, 33] have already considered this problem. The calculation is a weighted sum of Gaussian PDF calculations, and while rearrangement of the algebra is possible the algorithm used is fixed. The optimizations possible for this calculation are discussed at length in section 9.2. As described in chapter 5, training these systems usually involves an iterative technique such as the EM algorithm, which dominates the use of the computer resources. We also exploited the benefits of GPU to speedup the computation of the ISC technique, which improved the performance of language ID systems dramatically (see chapter 8).

In the second part of this chapter, we investigate the extent to which the best performance, in terms of both accuracy and use of computer resources, is achieved not only by optimising the mapping of the spectral analysis algorithm onto the GPU architecture, but also by choosing a spectral analysis technique where a high degree of optimisation can be realised. It is important to note that our concern is the performance of the whole feature extraction process, from a waveform to a sequence of MFCC vectors, rather than simply an abstract comparison of spectral analysis techniques in isolation. Thus, although the feature vector dimension and frame rate are fixed, other parameters, such as the analysis window size, are chosen empirically to optimize language ID performance for each approach.

Language ID is an example of a real signal processing problem where restricted computing power is a significant constraint. We have already seen (chapter 8) that the feature vectors in a typical language ID system comprise 'static' MFCCs plus a number of SDCs. All of the experiments in this chapter use 66 dimensional vectors comprising 17 MFCCs plus 49 SDCs and the language ID system is a simple GMM-UBM acoustic system (chapter 5) with 2048 or 4096 components. However, the same basic architecture is applied in many other pattern recognition problems. Hence the results presented in this chapter are more widely applicable. The essential content of this chapter also appears in [109].

9.1 Graphics Processing Units

9.1.1 Hardware description

GPUs are add-on processors which accelerate the performance of PCs applied to the composition of video code for computer games. The experiments described in this paper use the NVIDIA GE Force 260 core 216, a typical mid-range GPU. It comprises twenty seven multiprocessors, each with eight core processors, giving 216 cores operating at a clock rate of 1242 MHz (table 9.1). Each core processor can execute a floating-point multiply/add in a single clock cycle. The resulting GPU has a maximum performance of more than 800 GFlops. Each multi-processor also has two transcendental-function calculation units and shared memory. However it is important to note that the cores have a high latency, so that the result of a computation is not available until after 32 clock cycles. This has implications for algorithms that require access to prior results to complete the current calculation.

9.1.2 Software

GPUs were initially difficult to program due to their restrictive programming model. However, the introduction of NVIDIA's Compute Unified Device Architectures (CUDA) framework has simplified development, allowing the GPU to become a more mainstream tool [110].

Multi-Processors	27
Core processors	216
Core Clock (MHz)	585
Processor Clock (MHz)	1242
Compute Capability	$\tilde{8}$00 GFLOPS
Memory Clock (MHz)	999
Memory Configuration (MB)	1792
Memory Interface Width	448-bit
Memory Bandwidth (GB/sec)	111.9

Table 9.1: NVidia Geforce GTX260 (216 cores) Specifications

9.1.2.1 NVidia CUDA

CUDA is a C like programming language for writing *kernels*, the functions that execute concurrently on GPU. A *kernel* comprises a configurable number of blocks, each consisting of a configurable number of threads [111], as shown in figure 9.1, where each thread applies the kernel function to different data. Built-in variables indicate which thread of which block is currently executing.

A CUDA program comprises a CPU-based host function, which can be a simple function or a full program, plus one or more GPU-based parallel kernel functions. The host transfers data to the GPU memory, calls kernel functions after configuring the execution parameters, and transfers results back to the host memory. Kernel functions run as a grid of blocks of up to 512 threads, which execute concurrently, so that all computation in the grid must end before invoking another grid. All threads in a block run in lock-

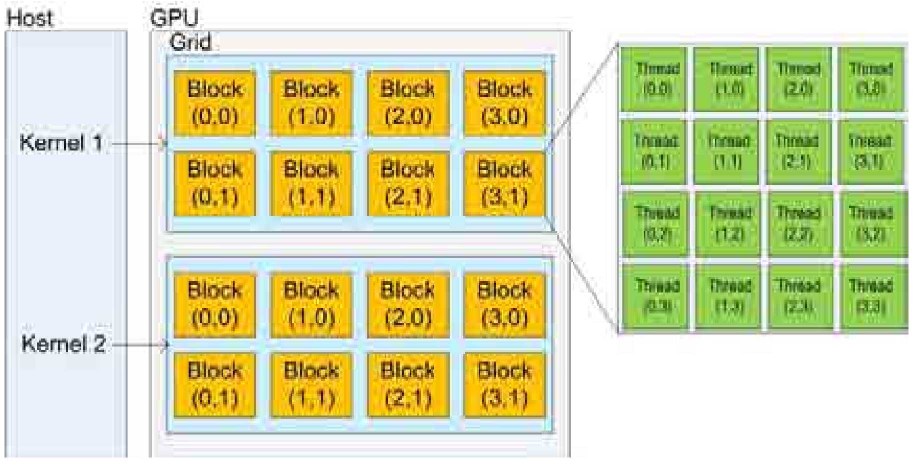


Figure 9.1: Overview of CUDA threads batching [111]

step on the same GPU multiprocessor. There is no communication between blocks, but threads in the same block share fast local cache memory, as shown in figure 9.2, and can be synchronized at programmer specified barriers. Since enormous numbers of blocks can be launched in parallel, a very high number of threads run concurrently.

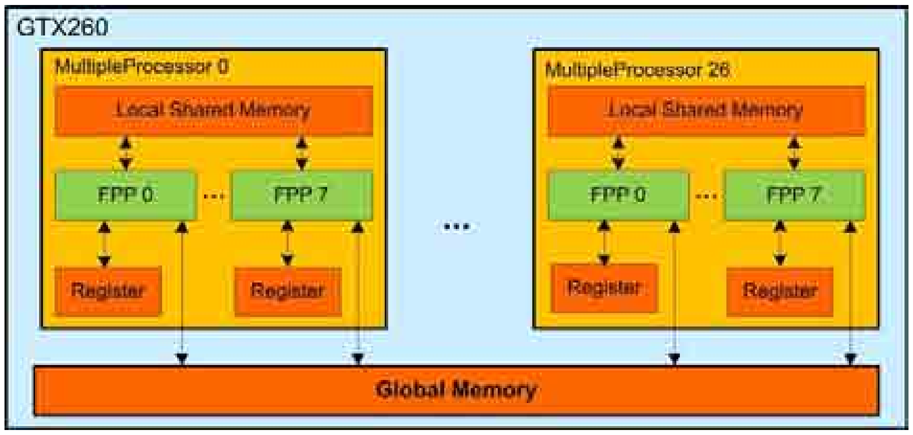


Figure 9.2: logical structure of graphics processor unit (GTX260) [111]

Most data resides in the shared global GPU memory as each multiprocessor's shared cache memory is limited. When calling a CUDA kernel, parameters are passed by value to the local cache, but array pointers will reference the shared global memory, which has a significant penalty for reading. When data is required from the global memory, it is best that adjacent threads in the block request adjacent memory locations, as this results in only a single delay. Kernels may use inter-thread data requests to rearrange memory requests. Processed data is returned to the global memory by the end of the thread's execution, so that the results are accessible from the GPU by the host. The host function can be based on a low-level API, called the CUDA driver API, or on a higher-level API, called the runtime API, as illustrated in figure 9.3.

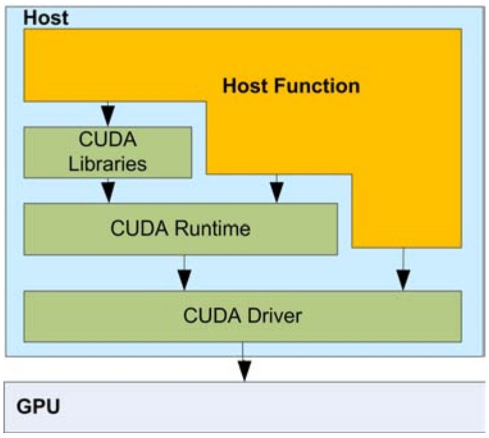


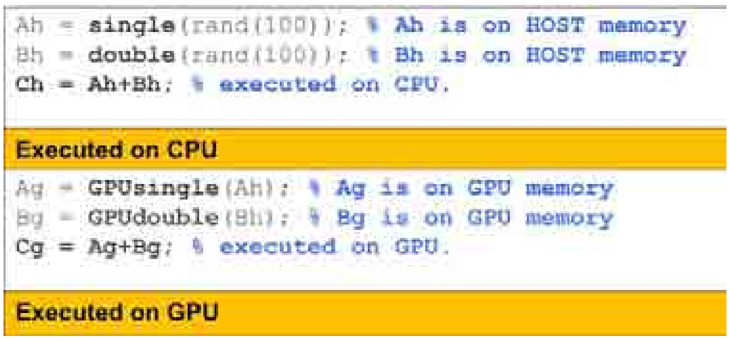
Figure 9.3: Software architecture of CUDA application [111]

9.1.2.2 GPUmat

GPUmat is one of several toolboxes which allows MATLAB to benefit from the computation power of modern GPUs. It is built on the top of NVidia CUDA technology. The computation is directly executed on the GPU transparently. Only the declaration of variables needs to be changed using new specific keywords. So it does not require a good knowledge in the GPU or the CUDA programming. Because of its simplicity and free availability, GPUmat is used as an interface between MATLAB and GPU.

Data is transferred from the host memory to the GPU memory implicitly by declaring the MATLAB variables as *GPUsingle* or *GPUdouble* for single and double precision floating-points respectively. Any operation on variables declared in the GPU memory will be executed directly on the GPU multiprocessors. In the same way, copying back the computation results from GPU memory to the host memory can be done simply by declaring the *GPUsingle* or *GPUdouble* variables as MATLAB *singles* or *doubles*.

Figure 9.4 shows a simple MATLAB example of how to port addition of two vectors from CPU to GPU using GPUmat toolbox. In this example, Ah and Bh vectors are initialized in the host memory. The addition operation is executed on the host CPU and the result is stored in the host memory. To port this piece of code from the host to the GPU, first the two data vectors need to be copied to the GPU memory. With GPUmat, this can be done simply by declaring two variables, Ag and Bg, and initializing them with the host vectors, Ah and Bh respectively. The addition of two vectors; Ag and Bg will be executed on the GPU and the result will be stored in the GPU memory. To copy the result vector Cg from GPU to host memory, the same declaration is used but with MATLAB operators *double* or *single*.



```
Ah = single(rand(100)); % Ah is on HOST memory
Bh = double(rand(100)); % Bh is on HOST memory
Ch = Ah+Bh; % executed on CPU.
```

Executed on CPU

```
Ag = GPUsingle(Ah); % Ag is on GPU memory
Bg = GPUdouble(Bh); % Bg is on GPU memory
Cg = Ag+Bg; % executed on GPU.
```

Executed on GPU

Figure 9.4: Example of using GPUmat

As shown in the example in figure 9.4, porting an existing MATLAB code from the host CPU to the GPU is as simple as changing *single* and *double* to *GPUsingle* and *GPUdouble* respectively. The right-hand variables of any statement should be defined either in host memory or in GPU memory but not in both. In other words, mixing between MATLAB variables and GPUmat variables in one statement is not allowed. A wide range of standard MATLAB functions (e.g. REPMAT) have been implemented in the GPUmat but still some functions have not been implemented (e.g. BSCXFUN). For simple Matlab programs, e.g. matrix multiplication, GPUmat can transfer them efficiently to GPU. For more complex tasks, GPUmat allows users to write their own functions by using low-level CUDA driver API which gives the programmer the best level of control for optimization to achieve the best performance.

9.2 GMM probability computation optimization techniques

GPU technology offers dramatic performance improvements for computationally intensive algorithms that are amenable to parallelisation. An example is the set of algorithms associated with Gaussian Mixture Models (GMMs). A GMM is a PDF p defined as a linear combination of Gaussian PDFs, (combining equations 5.1 and 5.3 in chapter 5)

$$p(x_t|\lambda) = \sum_{i=1}^{M} \frac{w_i}{(2\pi)^{D/2}|\Sigma_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x_t-\mu_i)^T\Sigma_i^{-1}(x_t-\mu_i)\right) \tag{9.1}$$

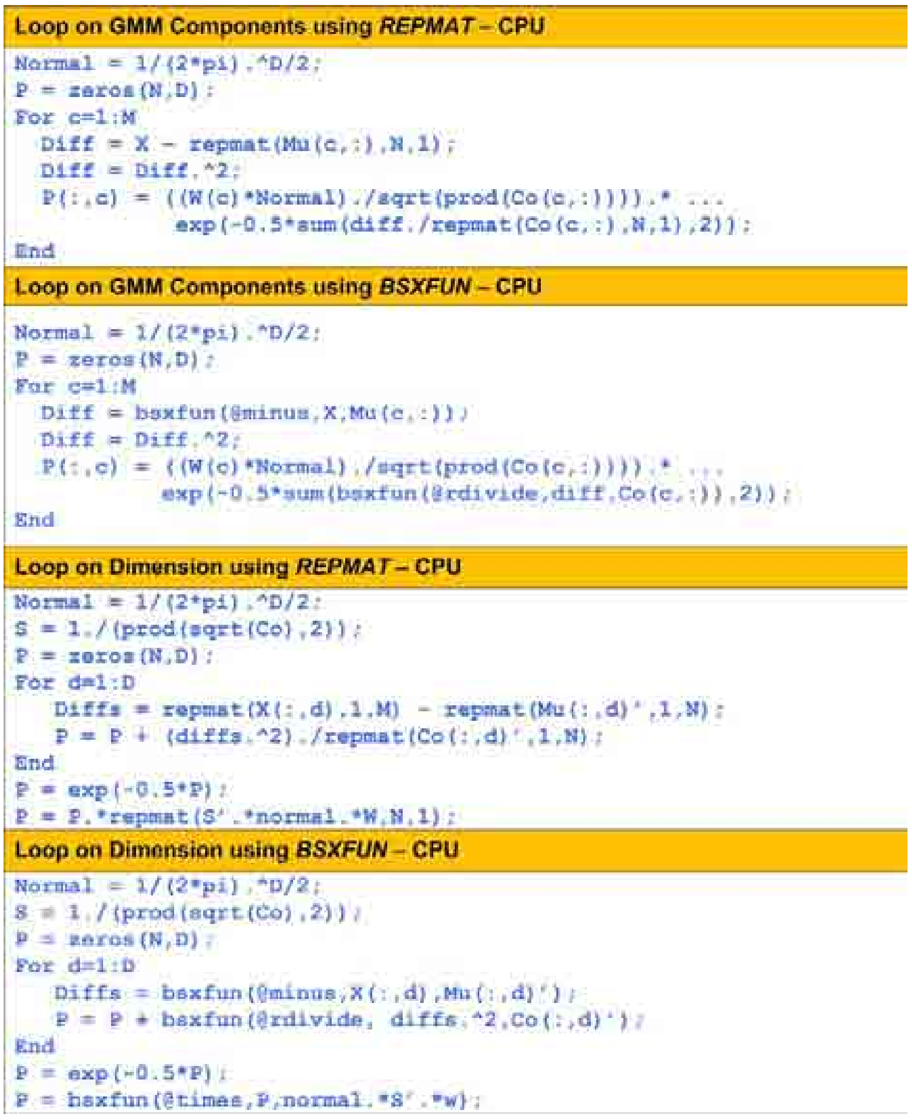
such that, $w_i \geq 0$, and $\sum_{i=1}^{M} w_i = 1$. x_t is an observation vector with dimension D at time t, M is the number of Gaussian components in the mixture and λ is the set of GMM model parameters $\{w_i, \mu_i, \Sigma_i\}$ where, w_i, μ_i and Σ_i are the weight, mean and diagonal covariance of the i^{th} Gaussian component.

As explained in chapter 5, the application of GMMs involves two processes, training, typically using the EM algorithm, and maximum likelihood classification. The high computational cost of these processes for large M and substantial amounts of data has already motivated research into the application of GPUs to this task, for example [30, 31, 32, 33].

9.2.1 Implementation details

Parallelizing the computation of GMM probabilities can be achieved in several ways: Computing the probabilities of one GMM component for all data vectors (loop over GMM components (i in equation (9.1))), computing the probabilities of all GMM components for each feature vector dimension of all data vectors (loop over dimension, $d = 1, \dots, D$ in (9.1) (assuming that the covariance matrices of the GMM components are diagonal)), or computing the probabilities of all GMM components for one vector (loop over data points, t in (9.1)). To exploit the capabilities of the GPU, a large number of threads must be executed concurrently. In a real LID task, the number of data points is much greater than both the number of GMM components and the number of feature dimensions. Thus, the third option, loop on data points, is excluded. If the number of GMM components is greater than the dimension of the features (which is the case for state-of-the-art LID systems), then looping on feature dimensions is best, otherwise, looping on components is best.

Assuming $X = \{x_n : n = 1 : N\}$ is a set of N feature vectors x_n, where $x_n = [x_{n1}, x_{n2}, ..., x_{nD}]$ is a feature vector of dimension D, and a GMM, as defined in equation 9.1, consists of M Gaussian components with μ mean vectors, Σ diagonal covariances and w weights. Two CPU algorithms for GMM probability calculation, 'loop over Gaussian components' and 'loop over feature dimension', are implemented in MATLAB using $REPMAT$ function as shown in figure 9.5. By profiling those algorithms (with $REPMAT$) implementations line-by-line with the MATLAB profiler, statements with the $REPMAT$ function are found to consume 60% of the overall computation. Consequently, an optimized mex-function $BSXFUN$ is used as an alternative to the $REPMAT$ function. Therefore, the two algorithms are re-implemented with $BSXFUN$ (figure 9.5).



Loop on GMM Components using *REPMAT* – CPU

```
Normal = 1/(2*pi).^D/2;
P = zeros(N,D);
For c=1:M
  Diff = X - repmat(Mu(c,:),N,1);
  Diff = Diff.^2;
  P(:,c) = ((W(c)*Normal)./sqrt(prod(Co(c,:)))).* ...
             exp(-0.5*sum(diff./repmat(Co(c,:),N,1),2));
End
```

Loop on GMM Components using *BSXFUN* – CPU

```
Normal = 1/(2*pi).^D/2;
P = zeros(N,D);
For c=1:M
  Diff = bsxfun(@minus,X,Mu(c,:));
  Diff = Diff.^2;
  P(:,c) = ((W(c)*Normal)./sqrt(prod(Co(c,:)))).* ...
            exp(-0.5*sum(bsxfun(@rdivide,diff,Co(c,:)),2));
End
```

Loop on Dimension using *REPMAT* – CPU

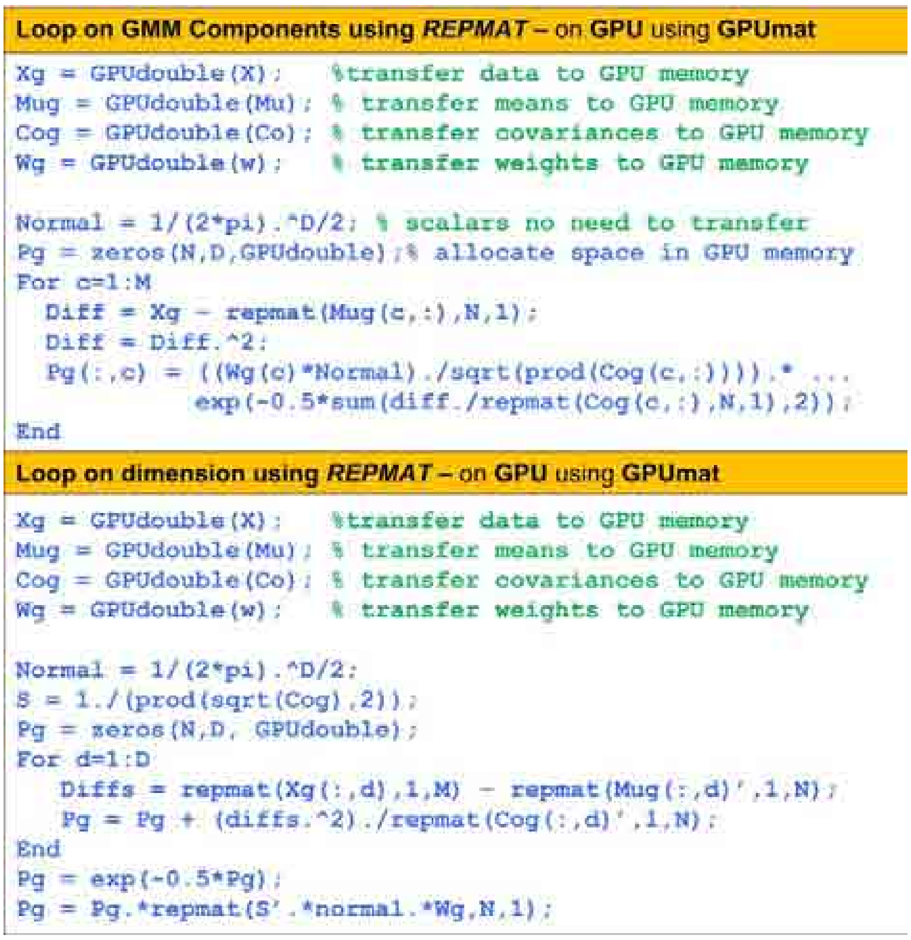
```
Normal = 1/(2*pi).^D/2;
S = 1./(prod(sqrt(Co),2));
P = zeros(N,D);
For d=1:D
   Diffs = repmat(X(:,d),1,M) - repmat(Mu(:,d)',1,N);
   P = P + (diffs.^2)./repmat(Co(:,d)',1,N);
End
P = exp(-0.5*P);
P = P.*repmat(S'.*normal.*W,N,1);
```

Loop on Dimension using *BSXFUN* – CPU

```
Normal = 1/(2*pi).^D/2;
S = 1./(prod(sqrt(Co),2));
P = zeros(N,D);
For d=1:D
   Diffs = bsxfun(@minus,X(:,d),Mu(:,d)');
   P = P + bsxfun(@rdivide, diffs.^2,Co(:,d)');
End
P = exp(-0.5*P);
P = bsxfun(@times,P,normal.*S'.*w);
```

Figure 9.5: Four different MATLAB implementations for GMM PDF computation on CPU

The $BSXFUN$ function has been implemented in GPUmat . Thus, only the algorithms with using *REPMAT* are converted to GPUmat as shown in figure 9.6.



Loop on GMM Components using *REPMAT* – on GPU using GPUmat

```
Xg = GPUdouble(X);   %transfer data to GPU memory
Mug = GPUdouble(Mu); % transfer means to GPU memory
Cog = GPUdouble(Co); % transfer covariances to GPU memory
Wg = GPUdouble(w);   % transfer weights to GPU memory

Normal = 1/(2*pi).^D/2; % scalars no need to transfer
Pg = zeros(N,D,GPUdouble);% allocate space in GPU memory
For c=1:M
  Diff = Xg - repmat(Mug(c,:),N,1);
  Diff = Diff.^2;
  Pg(:,c) = ((Wg(c)*Normal)./sqrt(prod(Cog(c,:)))).* ...
            exp(-0.5*sum(diff./repmat(Cog(c,:),N,1),2));
End
```

Loop on dimension using *REPMAT* – on GPU using GPUmat

```
Xg = GPUdouble(X);   %transfer data to GPU memory
Mug = GPUdouble(Mu); % transfer means to GPU memory
Cog = GPUdouble(Co); % transfer covariances to GPU memory
Wg = GPUdouble(w);   % transfer weights to GPU memory

Normal = 1/(2*pi).^D/2;
S = 1./(prod(sqrt(Cog),2));
Pg = zeros(N,D, GPUdouble);
For d=1:D
   Diffs = repmat(Xg(:,d),1,M) - repmat(Mug(:,d)',1,N);
   Pg = Pg + (diffs.^2)./repmat(Cog(:,d)',1,N);
End
Pg = exp(-0.5*Pg);
Pg = Pg.*repmat(S'.*normal.*Wg,N,1);
```

Figure 9.6: Porting MATLAB implementations of GMM PDF computation to GPU using GPUmat toolbox

As we have shown in figures 9.5 and 9.6, GPUmat does not offer the high flexibility of CUDA APIs. In our low-level implementation for the GMM computation, GPUmat is used only to transfer data between the host and the GPU memory, and the CUDA driver API is used to configure parameters of grids, blocks and threads when invoking the kernel function. The same kernel function is executed many times concurrently by each thread but with different input parameters. Feature vectors are divided into streams of maximum size 65535 blocks. Each block contains a maximum of 512 threads. Each

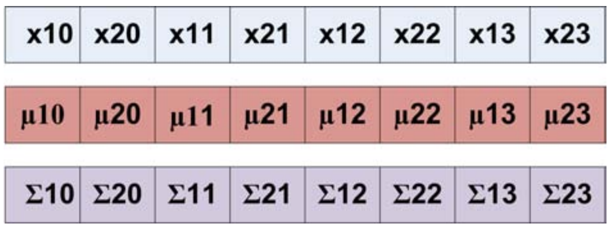
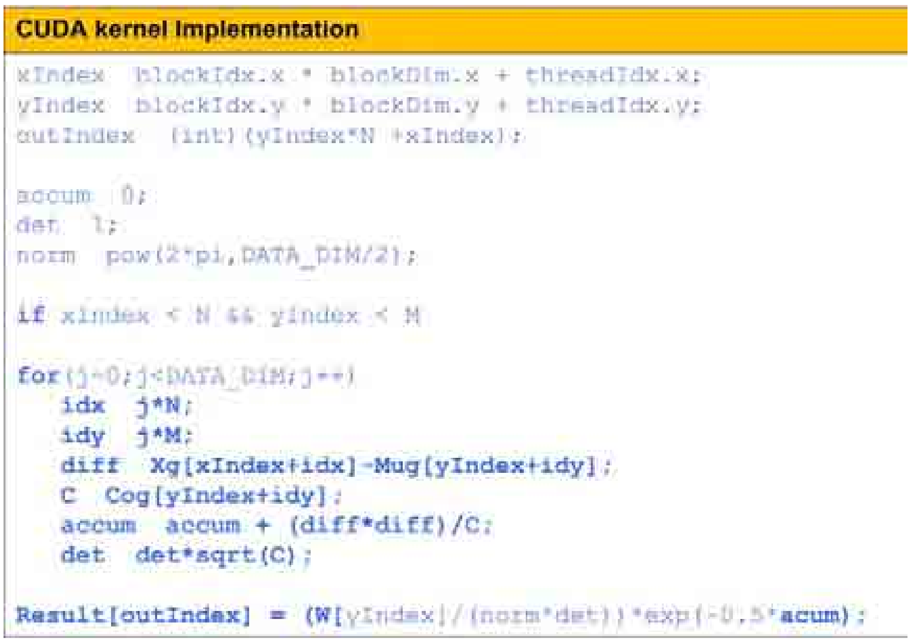

x10 x20 x11 x21 x12 x22 x13 x23
μ10 μ20 μ11 μ21 μ12 μ22 μ13 μ23
Σ10 Σ20 Σ11 Σ21 Σ12 Σ22 Σ13 Σ23

Figure 9.7: Example of data organization in the GPU memory

thread computes the probability of one feature vector given one Gaussian component and writes back the result into a specific space in the global GPU memory.

To ensure efficient access to the GPU global memory during kernel execution, the feature vectors and model parameters are re-organized at the feature level, such that the first elements are the first feature of each data vector and each GMM component parameter followed by the second feature and so on. Figure 9.7 shows an example of two 4-dimensional feature vectors and the parameters (μ and Σ) of two Gaussian components. In this figure, x_{td} denotes feature dimension d in vector x_t, and μ_{id} and Σ_{id} denote dimension d in mean vector and diagonal covariance vector of GMM component i, respectively.

The algorithm in figure 9.8 shows the implementation of the kernel function which is executed by every thread. *DATA_DIM* is the dimension of the feature vectors (66 in our case), N is the number of vectors and M is the number of GMM components. In this implementation, the probability of a single feature vector given a single Gaussian component is computed by one thread. Copying feature vectors and model parameters from the global memory to the shared static memory speeds up the access time 100x because global memory access time is around 100 clock cycles, whereas shared memory access time is only one clock cycle. In our case, the high dimensional feature vectors (66) and model parameters cannot be transfered to the shared memory because of the limited size of the shared memory, 64 KB.



CUDA kernel Implementation

```
xIndex  blockIdx.x * blockDim.x + threadIdx.x;
yIndex  blockIdx.y * blockDim.y + threadIdx.y;
outIndex  (int)(yIndex*N +xIndex);

accum  0;
det  1;
norm  pow(2*pi,DATA_DIM/2);

if xindex < N && yindex < M

for(j=0;j<DATA_DIM;j++)
   idx  j*N;
   idy  j*M;
   diff  Xg[xIndex+idx]-Mug[yIndex+idy];
   C  Cog[yIndex+idy];
   accum  accum + (diff*diff)/C;
   det  det*sqrt(C);

Result[outIndex] = (W[yIndex]/(norm*det))*exp(-0.5*acum);
```

Figure 9.8: CUDA kernel Implementation

9.3 Results and discussion

To compare the performance of different implementations of the GMM probability calculation on CPU and GPU, pre-trained GMM models with order M = 32, 64, 128, 256, 512, 1024, 2048 and 4096 are used to calculate the probabilities of a 100 second speech segment on CPU and GPU with different implementation methods. A front-end speech processor converted the acoustic waveform into a sequence of 66-dimensional acoustic feature vectors at 100 frames per second, resulting in 10,000 vectors. The CPU implementations ran on a 2.33GHz Intel Core 2 Quad Q8200 CPU with 4GB of memory. The GPU implementations ran on an Nvidia GeForce GTX 260 (Core 216) GPU installed in a

PC running 64-bit windows7, and an NVidia C1060 Tesla machine running 64-bit Linux. The NVidia GeForce GTX 260 was described previously. The C1060 Tesla machine comprises 30 multiprocessors (240 cores) and has 4GB of RAM. Microsoft visual studio 2008 and CUDA toolkit version 4.0 were used in compilation. All tests were repeated 10 times to obtain average CPU and GPU times.

GMM Order	LOC-$REPMAT$	LOD-$REPMAT$	LOC-$BSXFUN$	LOD-$BSXFUN$
32	0.4517	0.8534	**0.3064**	0.5073
64	0.9015	1.6591	**0.6013**	1.0014
128	1.8055	3.2963	**1.197**	1.98
256	3.5048	7.48	**2.371**	3.913
512	7.073	19.334	**4.836**	7.672
1024	13.97	39.73	**9.746**	15.35
2048	29.3	79.217	**19.345**	30.8618
4096	58.489	160.48	**40.398**	61.254

Table 9.2: Computation time (in seconds) for GMM Probability calculations for 10,000 66-dimensional acoustic frames running on CPU. LOC refers to 'loop on GMM components' and LOD to 'loop on dimension'.

Table 9.2 shows the computation times (in seconds) for four different implementations of GMM probability calculation running on CPU, namely looping over GMM components, i, using MATLAB function $REPMAT$ (column 2), looping over dimension, d, using MATLAB function $REPMAT$ (column 3) , looping over GMM components using optimized mex-function $BSXFUN$ (column 4) and looping over dimension using

GMM Order	NVidia GeForce GTX 260			NVidia C1060 Tesla		
	LOC-GPUmat	LOD-GPUmat	LOD CUDA	LOC-GPUmat	LOD-GPUmat	LOD CUDA
32	0.4336	0.4777	0.0202	0.2376	0.173	**0.0162**
64	0.8615	0.5182	0.023	0.4635	0.1747	**0.0219**
128	1.7204	0.6182	**0.0314**	0.9162	0.2524	0.0393
256	3.3612	1.2052	**0.0499**	1.8204	0.4585	0.0987
512	6.7686	1.4401	**0.0838**	3.6306	0.8734	0.1883
1024	13.6125	2.0032	**0.1432**	7.25	1.7013	0.2447
2048	26.7198	3.5713	**0.2838**	14.5014	3.3584	0.822
4096	53.56	7.16	**0.634**	28.953	6.673	1.6701

Table 9.3: Computation time (in seconds) for GMM Probability calculations, including data transfer time, for 10,000 66-dimensional acoustic frames on GTX260 GPU and C1060 Tesla, implemented with GPUmat and CUDA driver API. LOC refers to 'loop on GMM components' and LOD to 'loop on dimension'.

$BSXFUN$ (column 5).

The results in table 9.2 show that the implementations with a loop over the GMM components outperform that with a loop over dimension when using either $REPMAT$ or $BSXFUN$. The explanation for this is that $REPMAT$ and $BSXFUN$ are used twice in the loop over dimension implementations and only one time in the loop over components implementations (see algorithms in figure 9.5). It is also clear that using the $BSXFUN$ outperforms the $REPMAT$ function, because it is highly optimized for the C/C++ programming language. We also note that in the loop over components case, the computation time is proportional to the number of GMM components because the algorithm is optimized to calculate the probability of a single GMM component and executed according to GMM order.

Because the optimized $BSXFUN$ function is not available in the GPUmat toolbox, only the implementations with the $REPMAT$ function are ported to the GPU using GPUmat. Table 9.3 shows the computation times (in seconds), including data transfer time, for three different implementations running on both the GTX 260 GPU and C1060 Tesla machine. As shown from the results, we gain no improvement when porting the loop over GMM components' algorithm, with $REPMAT$, to the GTX 260 GPU using GPUmat (column 2), whereas, we get around 2X speed up when running the same algorithm on the Tesla machine (column 5) because of its superiority in hardware specifications. Note also that in both columns 2 and 5, computation time is proportional to the number of GMM components, because the algorithm is mapped onto the GPU for a single GMM component and executed according to the GMM order. Columns 3 and 6 of table 9.3 show the results for a GPUmat implementation of the algorithm on the GTX260 and Tesla machine respectively, in which the loop is over the data dimension. In this case the increase in computation time is no longer proportional to GMM order. For up to 128 GMM components, the increase in computational load is accommodated by the capacity of the GPU and there is little increase in computation time. However, for 256 components there are insufficient threads to execute the entire process in parallel and so the algorithm must be implemented in two stages, resulting in an approximate doubling of computing time. This pattern is repeated for the higher order GMMs. The computation time for the 'LOD-$REPMAT$' GPU implementation for a 4096 component

GMM running on GTX260 is only 4.5% of the computation time of the corresponding algorithm running on CPU and just 18% of the corresponding computing time for the 'LOC-$BSXFUN$' (column 4). Columns 4 and 7 of table 9.3 show the results of further optimization using the low-level CUDA driver API for both the GTX 260 (column 4) and Tesla machine (column 7). For a 4096 component GMM these implementations reduce computation time relative to the best CPU implementation, column 4, by factors of 64 and 24, respectively. The superior performance of the GTX 260 relative to the higher-specification Tesla hardware may be due to the Linux CUDA driver.

These significant reductions in processing time allow approaches to be considered which would otherwise be ruled out on computational grounds. For example, the improvement in language ID performance reported in [65] and in section 8.1.5, results from the application of a 'GMM tokenizer' based on a 24,576 component GMM (our MLM-*uni*-gram system in table 8.9). Without the increased computing power offered by a GPU this would not have been attempted.

In a previous study, Cardinal at al. [30] used the NVidia GeForce 8800GTX with CUDA to accelerate probability calculations in an Automatic Speech Recognition (ASR) system. The probability computation is 5 times faster with their GPU implementation than the optimized CPU implementation, leading to 26% computation time reduction in their ASR system. Gupa and Owens [32] achieved over 60% reduction in GMM computation time and 90% reduction in the memory bandwidth with an NVidia 8800GTX GPU. The specification of the NVidia 8800GTX, with 128 cores and 768MB of RAM, is less capable than our GPU's, and the CPUs are also different, so that direct comparison is difficult. However, our GMM computation speedup factor is more than nine times greater, leading to 97.92% reduction in GMM computation time (see table 9.4).

	GMM-UBM (2048) Training	**NIST 2003 Evaluation**	**Overall**
CPU	174.264	2.8811	177.145
GTX260	14.6227	0.1844	14.8071
Speedup	11.917	15.6	11.97

Table 9.4: Processing time (in hours) for training and evaluation GMM-UBM language recognition system with 2048 Gaussian components. This system is trained on train and development sets of CallFriends corpus (around 1200 hours) and evaluated on NIST LRE 2003 30-sec evaluation segments (1200 30-sec).

9.4 Optimizing ISC computation

We have shown experimentally earlier in section 8.1.4, that the ISC technique is very beneficial for language ID, particularly at the feature level. However, we have also shown that this technique requires a huge computation, which limits its usage.

The results of using a GPU to speedup the GMM probability computation motivates us to accelerate the ISC computation with the GPU. By referring to equation 5.30 in section 5.7, we find that the compensation offset term depends on three factors; the GMM posterior probabilities, the eigen-channel subspace (U matrix) and the channel factors (X_c). The computation of the first factor is already accelerated as explained in the previous section (section 9.2). Because of the high dimensionality of the eigen-channel subspace, a recursive algorithm is used to estimate a subset of its eigenvectors which correspond to the highest eigenvalues. Both the high dimensionality and the recursive algorithm make it difficult to speedup the estimation of U matrix with GPU. Fortunately, the estimation of U matrix depends only on the GMM model and training data, thus it can be estimated off line and only once.

The third factor in the compensation offset term is the vector of channel factors, X_c, which depends on the feature vectors with their GMM posterior probabilities and the pre-computed U matrix. The posterior probabilities,$\gamma(t)$, are computed with the same GPU algorithms described earlier in section 9.2. Returning to the channel factors equations (equations 5.28, and 5.29) we find that estimating X_c involves two terms. the first is the summation over feature vectors (T) in equation 5.28 ($\sum_{t=1}^{T} \gamma_i(t) \frac{x_t - \mu_i}{\sigma_i}$) which is very similar to the summation term in the GMM probability evaluation (equation 9.1), therefore, a similar algorithm, particularly looping on dimension, is used to optimize the computation of this term. The first summation (over GMM components,M) in equation 5.28 ($\sum_{i=1}^{M} U_i^T$) disappears by vectorizing the multiplication of the U^T with the weighted sum of the normalized feature vectors, $U^T \times V_s$, where V_s is vectorized sum of normalized feature vectors. The second term is the two summations in equation 5.29 which also can be easily converted to matrix multiplication by weighting the U matrix with the GMM posterior probabilities and then computing $U_w^T \times U_w$, where U_w is a weighted U matrix. Matrix multiplication can be easily parallelized and computed efficiently on both the CPU and GPU. The CUBLAS (4.0) library (part of CUDA toolkit 4.0) which is highly optimized for matrix operations on GPU, was used to compute matrix multiplication on the GPU.

To study the benefit of the GPU for accelerating the ISC computation and compare it with CPU, ISC (at feature level) was used to compensate the 66-dimensional feature vectors of 30-sec and 30-minute speech segments using the UBM with 2048 components and a pre-computed U matrix with 400 eigenvectors (i.e. dimension of U is 135168×400). The optimized algorithm for calculating X_c and feature domain compensation was run on the CPU and on our GTX 260 GPU. The computation times (in seconds) of compensating 3000 and 180000 feature vectors extracted from 30-sec and 30-minute speech utterances, respectively, are reported in table 9.5. The computation time (in hours) for compensating the whole training data (600 hours of speech) and NIST 2003 evaluation data (1200 30-sec segments) on both the CPU and GPU is also presented in table 9.5.

The results in table 9.5 show that with a GPU, the computation time required for compensating 30-sec and 30-min is reduced by a factor of 4.82 and 13.96, respectively. We have mentioned earlier in chapter 8 that compensating the whole training and NIST 2003 30-sec testing data with an un-optimized ISC algorithm running on CPU took around 21 days using UBM of 512 components. With the optimized ISC algorithm, compensating cepstral features for the whole data (training and testing), took around 163 hours when running on CPU and only 13.27 hours when running on our GTX260 GPU. This means that with the GPU, the ISC computation is accelerated by a factor of around 12, which makes it possible to use this powerful technique in all of our experiments.

Speech Data	**CPU**	**GPU**	**Speedup**
30-sec	17.52s	3.63s	4.82
30-min	884.56s	63.36s	13.96
ALL train and test data	163.22 hours	13.27 hours	12.3

Table 9.5: Computation time of ISC at feature level on CPU and GTX260 GPU

9.5 Speech feature extraction

In the previous sections we have demonstrated the importance of program structure in achieving optimal GPU performance for a specific algorithm. However in some applications there is a choice between different algorithms. In this section we consider algorithm choice for feature extraction in terms of throughput and language ID performance. Throughput is measured for 30 second speech samples (a typical test utterance length in language ID) and 30 minute samples (a typical training utterance length).

The process of extracting cepstral features from speech signal, with four alternative algorithms for spectral analysis, was described in detail in chapter 3. In the case where spectral analysis is performed using a FFT (see section 3.2.1.3), profiling shows that this is the most costly component, in terms of computation time, in the whole feature extraction process. The time taken to process a 30-min conversation, excluding spectral analysis, is approximately 5.07 seconds, while spectral analysis of the same conversation requires approximately 4.6 seconds. This motivated our investigation of GPU implementations of spectral analysis.

9.5.1 IIR filter bank implementation

The IIR filter bank is described in section 3.2.1.1. Our IIR filter bank consists of 18 IIR filters, each of order 5. Unfortunately the recursive nature of this algorithm causes difficulties when it is run on a GPU, since each calculation of the output cannot complete until all its predecessors have been calculated. This clearly implies that the computation must be arranged in a serial rather than a parallel manner. However, we have overcome this problem by dividing the data into overlapped short segments and applying IIR filters independently.

Assuming that the speech signal is stationary, this dependency can be reduced by dividing the input signal into overlapped segments of 20-30 ms. The starting sample of each segment is shifted by 80 samples with respect to the previous segment to give outputs every 10 ms. For example, a 30 minute utterance is divided into 180,000 segments, which permits a large degree of parallel processing on the GPU. Consequently, each core processor implements one IIR filter on one segment independently of the other segments. Doing this introduces a small amount of inaccuracy, since the state of a filter at the start of a segment is set to all zeros rather than the state of the filter at the termination of the previous segment. However, this approximation does not markedly affect language ID performance. The implementation comprises a fifth order IIR Filter Bank consisting of eighteen channels. In this implementation, filtering a single speech segment with a single filter channel is computed by one thread. Consequently, filtering 30 segments with seventeen filter channels is computed by one block of threads since the maximum size of a thread block is 512 threads. The maximum number of launched thread blocks is 65,535. For memory access efficiency, the input samples are re-organized in a way that begins with the first sample of all segments, then the second sample, then the third and so on. Moreover, the coefficients of the seventeen filters are copied to the shared memory in order to reduce the global memory access. Because of the limited size of the shared memory, the input samples cannot be copied.

Algorithm 1 (in appendix A) shows the kernel of the IIR Filter Bank implementation with a two-stage single-pole filter smoothing. The algorithm works as follows: two shared arrays are declared to store the coefficients of the seventeen filters. Each thread copies single coefficient from the global memory to the shared memory and waits until all threads in the same thread block complete their operation. After that, every thread applies the recursive IIR algorithm on its own input samples and filter coefficients and writes the result into the output array in the global memory.

9.5.2 FIR filter bank implementation

The FIR filter bank is described in section 3.2.1.2. Our FIR filter bank is implemented with 19 filter-pairs (I and Q), each with 80 coefficients. Since the I and Q FIR filters require the same computation, one kernel is used to implement them both. The input stream of samples is divided into an array of 10ms (80 samples) non-overlapped frames and the same input samples are shifted by 5 ms (40 samples) and divided into another array of 10 ms frames. This results in two arrays with the same number of frames where corresponding frames have a time shift of 5 ms. The frames of samples in the two arrays are organized in the same way described earlier for the IIR filter bank (section 9.5.1), for memory access efficiency.

Our FIR kernel implementation for both I and Q filters is shown in algorithm 2 (in appendix A). First, the 80 coefficients of all I and Q FIR filters are copied to the shared memory before starting the filter operation because access time of the on-chip shared memory is only around 1% of that for global memory. Similar to the IIR filter kernel, a single thread applies single filter-pair (I and Q) into one frame of the input arrays at a time and writes back the output into an output array allocated in the global memory. Therefore, every thread block applies the nineteen filter-pairs to 26 input frames as the maximum number of threads per thread block is 512. The number of launched thread blocks is set to its maximum, 65,535.

As shown in equations 3.4 and 3.5, the filter operation is simply matrix multiplication. Therefore, each thread does 160 MUL-ADD operations for both I and Q filters, giving two outputs. These are squared and added to get the total power of the I and Q filters for every input frame. By applying this kernel to the two input arrays, we get two output arrays of size $N \times 19$, where N is the number of input frames (in each array) and 19 is the number filter-pairs (I and Q). The two output arrays are then added together to get a frame rate of 100 frames per second. The implementation requires $6,213$ floating-point operations (FLOPs) for each input frame, or $N \times 6,213$ in total.

9.5.3 FFT filter bank implementation

The FFT filter bank is described in section 3.2.1.3. Our CPU implementation of the FFT is based on the FFTW library [112, 113] which decomposes the problem recursively using the Cooley-Tukey algorithm [49] until it can be solved with one of several other algorithms such as the prime factor algorithm [46] or a split-radix algorithm [114]. The recursive nature of these algorithms imposes some limitation on the GPU implementation. The GPU FFT implementation is based on the NVidia CUDA library, cuFFT (version 4.0) [115]. These algorithms were compared by computing FFTs of 65,536 256-dimensional vectors of single-precision reals (a total of 16M elements). Ignoring the transfer time, the GPU implementation achieved around 76 GFLOPS/s, about ten times faster than the CPU algorithm, which is comparable with the best published results [116, 117]. This supports our assumption that the FFTW and cuFFT libraries are among the best implementations of the FFT available for CPU and GPU.

However, computation of the FFT coefficients is part of the DFT filter bank in our language ID system front-end. Therefore, memory transfer time should be added to the FFT computation time when running on the GPU. After taking memory transfer time into account, the GPU performance drops to 14.5 GFLOPS/s, approximately twice as fast as the CPU.

In our implementation of DFT filter bank, the input samples are divided into 20 ms frames (160 samples) overlapped by 10 ms (80 samples) to give 100 frames every second. Prior to the 256 point transform each frame is windowed with a 20 ms Hamming window and 96 zero samples are appended. The absolute values of the first 128 coefficients give the power spectrum, which is then filtered with nineteen Mel-scaled triangular filters. The DCT of the log of the filter outputs gives the cepstral features.

Alternative window sizes were considered for both the FFT and FIR filter banks, but the configurations described in this and the previous section were found to give the best language recognition performance.

9.5.4 LP implementation

Matlab built-in functions running on CPU were used to estimate 16 LP coefficients which were then used to calculate 19 linear frequency cepstral coefficients. For more details refer to chapter 3.

A GPU implementation for estimating the LP coefficients was developed with GPUmat and the NVidia CUDA tools. The input speech signal is divided into overlapped 20ms (160 samples) segments. The starting sample of each segment is shifted by 80 samples with respect to the previous segment to give outputs every 10ms. By considering that

LP coefficients can be predicted from the previous input samples, every single core in the GPU calculates the autocorrelation matrix of one input segment independently. The Levinson-Durbin algorithm is then used to estimate 16 LP coefficients from the autocorrelation matrix and then writes them in the GPU global memory. The sizes of CUDA blocks and grids are set to their maximum 512 threads, and 65535 blocks, respectively. For memory access efficiency, the input samples are re-organized in the same way which described earlier for the IIR and FIR filter banks.

9.6 Results and discussion

9.6.1 Computational speed

In order to set up a comparison, in terms of computation time, between the four described spectral analysis algorithms when running on CPU and GPU, 30-second and 30-minute speech segments (typically testing and training utterances in our language ID system) were used to compute seventeen cepstral coefficients at frame rate of 100 frames per second with the four algorithms. Matlab, GPUmat and CUDA were used to implement the spectral analysis algorithms and optimize them for CPU and GPU. The process was repeated 20 times to ascertain average CPU and GPU times.

Table 9.6 shows CPU and GPU processing times for each of the speech analysis algorithm investigated. An analysis of the computing time on the CPU reveals that the FIR is the most efficient technique, followed by the FFT then LP and the IIR filter bank. The recursive nature of the IIR filters requires that every filter output sample must be computed, even though we only require an estimate of the envelope every 10 ms. This leads to a much larger number of computations compared to the other techniques.

For the short 30s segments of speech, the spectral analysis computation times for FIR, FFT, IIR and LP, with single precision arithmetic, are accelerated by factors of 1.43, 3.5, 2 and 4.5, respectively, when running on the GPU compared with the CPU. For the longer speech segments, 30 minutes, the corresponding speedup factors are 2.52, 6.92, 9.8 and 6.7, respectively. This confirms that the benefits of a GPU are best realized with larger amounts of data, because the parallelism in the GPU is used more effectively and memory transfer time is relatively less significant.

	CPU		**GTX260 GPU**	
	30-sec Speech	*30-min Speech*	*30-sec Speech*	*30-min Speech*
FIR	0.0120 (**0.0097**)	0.8500 (**0.600**)	0.0075 (**0.0068**)	0.3227 (**0.238**)
FFT	0.0622 (**0.028**)	4.6667 (**2.180**)	0.0118 (**0.008**)	0.7705 (**0.315**)
IIR	0.3498 (**0.3354**)	20.77 (**18.026**)	0.1701 (**0.168**)	2.1310 (**1.842**)
LP	0.2420 (**0.121**)	28.420 (**9.223**)	0.0301 (**0.0273**)	1.4306 (**1.397**)

Table 9.6: CPU and GPU processing times (in *seconds*) for FIR, FFT, IIR and LPC based speech analysis algorithms. The processing time includes data transfer time and it is measured for doubles (**singles**) precision floating-point operations.

As we have seen earlier in chapter 3, conventional IIR filters are unable to exploit the advantages of the GPU effectively because each new output sample cannot be calculated until the previous output sample is available. However, dividing the speech signal into overlapped short segments and mapping it onto the GPU, as described in section 9.5.1, overcomes this limitation and speeds up the IIR filter bank computation by factors of 2 and 9.8 (single precision) for the 30 second and 30 minute speech segments, respectively, without affecting language ID performance. By contrast, the FIR filter is well-suited to the GPU architecture as the kernels comprising the vector multiplication can be executed in any order. Therefore the scheduler is free to arrange the execution of these kernels in an order best suited to maintaining a high computational throughput. Extracting 180,000 cepstral feature vectors from the 30 minute speech using the FIR filter bank on the GPU is approximately 9 and 14 times faster than the conventional FFT-based filter bank on the CPU, for single and double precision arithmetic, respectively. Although the number of FLOPS required in the FIR filter bank algorithm is around 21% greater than those for the FFT filter bank, the FIR algorithm is faster. This is probably because the main computation in the FIR filter bank is matrix multiplication which can be de-composed into independent tasks that the GPU can process very efficiently.

9.6.2 Algorithmic performance

The generative GMM-UBM and the discriminative GMM-SVM language ID systems, with 2048 GMM components, are trained and evaluated on cepstral features extracted using FIR, FFT, IIR and LP spectral analysis algorithms. The percentage EERs for the language ID systems, with and without ISC technique, are shown in table 9.7. The performance of the GMM-UBM system with an FIR filter bank, 5.337% EER, is approximately 7% better than with an FFT filter bank, 17% and 32% better than with an

IIR filter bank and LP spectral analysis, respectively. The ISC technique improved the language ID performance of the four algorithms by around 38%.

	GMM-UBM		GMM-SVM	
	without *ISC*	with *ISC*	without *ISC*	with *ISC*
FIR	5.337	3.26	1.28	1.08
FFT	5.75	3.52	1.35	1.08
IIR	6.4	3.91	2.13	1.47
LP	7.86	4.82	2.93	2.11

Table 9.7: LID performance (EER[%]) for FIR, FFT, IIR and LPC based speech analysis algorithms.

When the languages are modeled discriminatively with the GMM-SVM system, the superior performance of the FIR filter bank based features relative to the FFT based features is reduced to just 5% and disappeared when applying ISC technique. In addition, the 90% confidence intervals [118] for the GMM-UBM and GMM-SVM results, without ISC, are approximately ± 0.65 and ± 0.34, respectively. This suggests that there are no significant differences between the language identification performances achieved with the FIR and FFT filter banks.

Finally, table 9.8 compares the results, in terms of computation time, for the complete NIST 2003 LRE task conducted using language ID systems with FIR-based spectral analysis on the GPU and FFT-based spectral analysis on the CPU. The results show that the utility of GPUs and FIR-based analysis demonstrated in the previous sections, transfers to a real, practical application. The whole 2003 NIST LRE is completed in sixteen hours with the GPU/FIR system, compared with 180 hours for the more conventional

CPU/FFT system, a speed-up factor of more than eleven, with no reduction in language ID accuracy.

	Frontend Processing(∼*1210 hrs of speech*)	**GMM-UBM Training**	**NIST 2003 Evaluation** (*1200 30-sec*)	**Overall**
CPU (FFT)	3.3554	174.264	2.8811	180.5
GTX260 (FIR)	1.308	14.6227	0.1844	16.115
Speedup	2.565	11.917	15.6	11.2

Table 9.8: Processing time in hours for feature extraction, training and evaluation GMM-UBM language ID with 2048 Gaussian components. This system is trained on train and development sets of CallFriends corpus and evaluated on NIST LRE 2003 30-sec evaluation segments.

9.7 Summary and conclusion

The use of a GPU can provide a high level of computation at very low cost. However in order to fully realise its potential, it is necessary to understand how to map an algorithm onto the GPU architecture so that the available processing power is used to best advantage. This not only includes the effective coding of predetermined algorithms, but also the choice of algorithm or technique for a specific function. The difference in architecture between a CPU and a GPU can lead to surprising results when an appropriate algorithm is selected, and improvements in performance can be achieved with little impact on the processing effort required. It also becomes possible to try new techniques which would otherwise not be possible due to their high computational load.

For example, a GPU based system using an FIR filter bank front-end can complete the NIST 2003 LRE language ID task in 16 hours, compared with 180 hours for a more conventional FFT-based system running on a standard CPU (a speed up factor of more than 11), with no reduction in language ID performance.

Chapter 10

Human and Computer Recognition of Regional Accents from British English Speech

In this chapter we apply the state-of-the-art language ID system developed in chapter 8 to extract paralinguistic information from English speech, namely the speaker's regional accent. As well as measuring the overall performance of our language ID system, we report the performance of its acoustic and phonotactic subsystems on this task. In addition, the effect of changing some parameters and techniques on accent recognition performance is also investigated. The performance of our language ID systems on the accent ID task is compared with human listeners' performance, and with two systems based on Huckvale's ACCDIST measure [18].

A justifiable objection to this approach is that the application of language ID techniques to this problem is counter-intuitive. This is because regional accent within a given language is an example of variability to which a language ID system should, by definition, be insensitive. However, we argue that this objection is not valid. As explained previously, mainstream language ID is based on generic statistical and pattern recognition methods, primarily GMMs, SVMs and n-gram language models, aided by various normalization techniques designed to remove irrelevant variations from the sequences of feature vectors that are to be classified. Our premise is that within a given language (in this case English) the distributions of acoustic feature vectors, or phone n-grams, corresponding to different regional accents are sufficiently distinct to enable the same methods to be applied successfully to regional accent recognition. The systematic differences between various regional accents of British English are well documented [1, 2]. For example, according to [1] the "two most important characteristics setting northern accents apart from southern ones are (i) the absence of the *foot* - *strut* split, i.e. the lack of a phonemic opposition between the vowels of *foot* and *strut*; and (ii) the absence of *bath* broadening, i.e. the use in *bath* words of the vowel of *trap*" (here 'northern accent' refers to an accent associated with the north or midlands of England). A quantitative analysis of the vowel systems for each of the fourteen accents in the ABI-1 corpus is presented in [119], which shows scatter plots of the first formant frequency F_1 against the difference between this and the second formant frequency, $F_2 - F_1$, for eleven monophthong vowels for each accent. These diagrams show both systematic inter-accent differences, and considerable intra-accent consistency. Accent differences in British English are not restricted to vowels [1], however quantitative data is less readily available.

The objective of the work described in this chapter, is to determine whether or not these inter-accent differences are sufficient to enable methods from language ID to be applied successfully to automatic accent recognition. The main results of this work also appear in [120, 121].

10.1 Automatic classification systems

It is convenient to divide the automatic systems used in this chapter into text-dependent systems, which require a word-level text transcription of the test utterance, and text-independent systems, which do not. The text-dependent systems are variants of Huckvale's ACCDIST approach [18]. The text-independent systems are the components of our language ID system, which was developed in chapter 8.

10.1.1 Text-independent automatic systems

As we have seen earlier in chapter 8, our language ID system (phonotactic and acoustic components) was tuned on the NIST 2003 evaluation data set and evaluated on the NIST 2005 evaluation set. This system is applied to accent recognition with the same parameters and configuration, except that our English phone recognizer (described below) is added to the phonotactic systems. This results in sixteen phonotactic systems (4 phone recognizers $\times$ 4 n-grams).

Since our target applications are for English, we built an English decision-tree triphone-based phone recognizer, using the HMM toolkit (HTK) [122]. We trained the acoustic models using training data from the ABI-1 corpus (see chapter 7). The system uses 39 dimensional PLP-based feature vectors (see section 3.2.1.4). All phone HMMs comprise 3 emitting states without state-skipping, with one 16 component GMM per state. The phone recognizer uses a bi-gram phone-level language model derived from the ABI-1 training set. The pronunciation dictionary was generated from the British English Example Pronunciation dictionary (BEEP)[1].

[1]"ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/beep.tar.gz", [cited April 30, 2011]

Phone Recognizer \n-gram	1-gram	2-gram	3-gram	4-gram
English	39	1521	41827	275175
Czech	42	1764	91125	235341
Hungarian	58	3364	39943	229587
Russian	52	2704	40830	246671

Table 10.1: Dimensions of n-gram vectors for each phonotactic accent recognizer

The dimensions of the n-gram (n = 1, 2, 3 and 4) vectors, after removing the zero occupancy components, for each phontactic system are shown in table 10.1.

In the accent experiments, there is no development set to train the fusing coefficients. Therefore we divided the test speakers (in each 'jackknife' round) into two sets. The accent and gender of speakers are distributed equally in both sets. One set is used to find the coefficients for fusing the systems on the second set, and vice versa. The fused scores are then combined together and the final performance is calculated.

10.1.2 Text-dependent automatic systems

10.1.2.1 ACCDIST-based systems

In [1], British English accents are characterized according to differences in the realization of vowels in specific 'key words'. Huckvale's ACCDIST measure [18] makes this notion computationally useful. He argues that a relative measure, based on differences between the realizations of vowels in different words, is not only a cue for accent recognition, but is also less sensitive to other speaker specific characteristics than measures that depend on absolute spectral properties. Similar approaches are advocated in [19, 123]. ACCDIST is text-dependent, since it requires a phone-level transcription of an utterance to which it is applied. In order to make our system comparable with Huckvale's systems, only the nineteen MFCCs plus energy are used for our ACCDIST system. Given a transcribed utterance, the start and end times of each realization of a vowel are identified. Each vowel segment is then split into two halves by time and the average feature vectors for each half are concatenated to create a single 40 dimensional vector. The distances between these vectors, corresponding to different vowels in different contexts, are calculated using an unweighted Euclidean distance and stored in a distance table. To ensure that distance tables are comparable, all utterances must share the same phone-level transcription. An accent is represented as the average distance table over all of the training utterances for that accent. A test utterance is classified according to the correlation distance between its distance table and those for each of the accents.

In our variants of ACCDIST, a phonemic transcription of each of the SPA recordings in the ABI-1 corpus was generated using standard pronunciations from the BEEP dictionary. This was force-aligned with the speech data using our English phone recognizer (section 10.1.1). Our ACCDIST-based system differs from that in [18] in two ways: First, we used the SPA data from ABI-1 rather than the "Short Sentence" files. The SPA recording was chosen because we believe it is more suitable for human perceptual experiments, and we wanted to use the same test material to test automatic and human recognition. Second, we used all of the SPA recordings in our experiment, whereas only those recordings which were completed without errors or repetitions were used by Huckvale. This is because our ACCDIST systems do not require each recording to correspond to exactly the same phone sequence. Alternatively, the speaker distance tables are built from vowel tri-phone segments (i.e. "phone-vowel-phone") rather than words. We also include vowel duration as an extra feature. For repeated tri-phones the mean feature vector was used. Hence, each SPA recording is represented as a sequence of pairs $(v_i\ ,p_i)$, where v_i is the 41-dimensional feature vector (two concatenated vectors comprising nineteen MFCCs plus energy, plus vowel duration) of the i^{th} tri-phone in the sequence and p_i is its label. The 105 most common tri-phones across all speakers in the training data were found and used in constructing the speaker distance tables. Our ACCDIST-based systems are described in the following sections.

10.1.3 ACCDIST-Corr.dist

A speaker distance table was calculated for each speaker by finding the distances between the feature vectors of every tri-phone pair in the common tri-phones list. Then, the mean of the resulting speakers' distance tables was calculated for each accent. Accent recognition was performed using the correlation distance between the test speaker distance table and the accent mean distance tables. To calculate the correlation distance, d, between the two vectors V_1 and V_2 of the length J, we consider the two vectors as two streams and we normalize them with their mean and standard deviation. Then, the correlation distance is calculated as follow (dot product between the two normalized vectors):

$$d(V_1, V_2) = \sum_{j=1}^{J} (\frac{V_1^j - \mu_1}{\sigma_1}) \times (\frac{V_2^j - \mu_2}{\sigma_2}) \tag{10.1}$$

where, μ_1, μ_2 are the means of vectors V_1 and V_2, and σ_1 and σ_2 are the standard deviation of the vectors V_1 and V_2, respectively. When the two vectors are independent, the correlation distance is zero, and when the vectors are identical, the correlation distance is one.

The correlation took into account only those tri-phone-pairs which occurred in the test utterance. An obvious shortcoming of the vowel tri-phone approach is the limited vowel context, compared with the whole-word contexts in Huckvale's system.

10.1.4 ACCDIST-SVM

The success achieved by applying SVMs to supervectors constructed from stacked MAP-adapted GMM means for language ID (see chapter 8) motivated us to apply SVMs to the speaker distance tables in our ACCDIST-based system. In our version of Huckvale's system above (ACCDIST-Corr.dist.), the average of the speaker distance tables for a given accent was used to represent that accent. By contrast, in our ACCDIST-SVM system, SVMs were applied to the 'vectorized' speaker distance tables of all accents. Due to symmetry, each 105 $\times$ 105 distance matrix has 5460 distinct entries, which are rearranged into a 5460 dimensional vector. By labeling the distance tables of one accent as a target class (+1) and the remaining distance tables as a background class (-1), this results in one SVM for each accent. A test speaker vectorized distance table is evaluated against every accent model. The correlation distance in equation 10.1 was used as a kernel for training and evaluating the SVM systems. In this case $J = 5460$, and V_1 and V_2 are the two (5460 dimensional) distance table vectors.

In the ACCDIST based systems, not all of the tri-phones pairs seen in the training data are necessarily found in the test utterance. Assuming there is enough training data, the set of tri-phones of the test utterance is a subset of the tri-phones learned from the training data. This results in distance tables of different sizes. As mentioned above, for the ACCDIST-Corr.dist systems, this problem is simply solved by calculating the correlation distance taking into account only those tri-phone-pairs which occurred in the test utterance. However, in the ACCDIST-SVM system the situation is more complicated because the accent-dependent SVMs are already trained on vectorized distance matrices for the complete set of tri-phone-pairs found in the training data of each accent. We solved this problem by training new SVMs during recognition using only the tri-phone-pairs of the test utterance. A shortcoming of this solution is the intensive computation of the SVM training in the recognition phase.

10.2 Human experiments

To provide baselines against which the automatic systems could be compared, a web-based human perceptual experiment was conducted. This experiment used exactly the same SPA test recordings as automatic classification. Twenty four native British English speaking subjects, aged between 21 and 78, took part in the experiment. Each subject completed a registration process in which he or she gave their gender and age and indicated which, if any, of the thirteen different ABI-1 locations they had ever lived in, and which of the fourteen regional accents they were familiar with. Each subject then listened to a different set of twenty SPA recordings, each varying in length between 30s and 40s, selected randomly from the test set. For each recording, subjects were asked to identify the accent of the speaker (out of the fourteen possible accents), the speaker's gender and age and to state their confidence in their decision. The listeners were naïve in that they had no formal training in phonetics or linguistics and no explicit training in regional accent recognition was given. Instead the listeners were required to accomplish the task using the knowledge of regional accents that they had acquired naturally through their experiences to date.

10.3 Results and discussion

For each component of our language ID system, the accent recognition performance is reported in terms of EER[%] (detection task) and accuracy [%] (identification task). As we have mentioned in section 7.2, each accent recognition system is evaluated on the 30s segments and the SPA recordings.

Table 10.2 shows the performances of the individual n-gram phonotactic systems ($n = 2, 3, 4$) for each phone recognizer (the 1-gram systems performed very poorly, and their inclusion did not improve the overall performance of the fused systems). Focusing on recognition accuracy and the SPA test data, although the fused result is best with the Hungarian phone recognizer (73%), no individual system outperforms all of the others consistently and the performance obtained with the English phone recognizer is relatively poor. Ultimately, it seems that what is important is consistency rather than phone recognition accuracy. Referring to the general structure of phonotactic system, figure 4.1 in chapter 4, in this application it may be better to regard these systems as abstract 'tokenizers' rather than explicitly as phone recognizers. The best result (EER=6.5%, Acc=82.1%) was obtained by fusing the twelve phonotactic systems together.

The results for all of the automatic systems are summarized in table 10.3. Columns two and three, and four and five of table 10.3 show accent recognition performance on the 30s cuts and the SPA recordings, respectively. We focus on the SPA results (columns four and five), as these are available for all of the systems. The performance of each of the acoustic systems is poor despite the facts that a GMM with a large number of

	English		Czech		Hungarian		Russian		Fused	
	30s	SPA	30s	SPA	30s	SPA	30s	SPA	30s	SPA
2-gram	25.2	22.7	26.8	26.4	23.4	23.4	28.6	22.6	17.7	16.8
	38	*38.6*	*39*	*37*	*46*	*43*	*32*	*48.2*	*52.5*	*55*
3-gram	19.7	18.3	20.2	14.7	17	1.74	24.7	14.9	11.4	8.5
	50	*53*	*50*	*62.7*	*60.3*	*65*	*40*	*62*	*70*	*68.4*
4-gram	18.4	14.8	17.7	9.6	14.2	10.8	23.9	14.6	9.8	6.7
	51	*58*	*55*	*68.7*	*62*	*71*	*42*	*63*	*73*	*79.6*
Fused	16.7	12.8	17.3	8.7	13.3	9.3	22.2	11.3	9.2	6.5
	56	*61*	*57*	*70.5*	*66*	*73*	*43*	*69*	*74*	*82.1*

Table 10.2: Performance EER[%] (*Accuracy[%]*) of phonotactic accent recognizers using English, Czech, Hungarian and Russian phone recognizers.

mixture components (4096) has been used and the recordings are good quality rather than telephone quality speech.

The phonotactic system (6.5% EER) outperforms all of the acoustic systems, despite the fact that phonotactic differences are apparently restricted because all of the recordings correspond to readings of the same text. The best accent recognition performance achieved with our language ID system is 5.16% EER (86.4% accent classification accuracy), which is obtained by fusing all of the acoustic and phonotactic sub-systems.

Both variants of the ACCDIST measure give better results than the acoustic-phonotactic language ID system. The ACCDIST system with correlation distance gives 2.66% EER (93.17% accent classification accuracy), and the ACCDIST-SVM system gives the overall best result of 1.87% EER (95.18% accuracy). This compares with 92.3% accent

System	Accent ID (30s)		Accent ID (SPA)	
	EER [%]	**Acc** [%]	**EER** [%]	**Acc** [%]
GMM-UBM (4096)	17.32	57.38	14.83	60.2
GMM-SVM (4096)	16.2	60.12	12.7	66.3
GMM-SVM-GMM (4096)	14.73	64.4	10.7	70.4
GMM-uni-gram	16.94	57.2	15.43	58.95
GMM-bi-gram	22.3	50.13	21.2	53.5
Acoustic-fused	12.82	72.3	9.3	75.6
Phonotactics	9.18	74.1	6.5	82.1
Phono-Acoustic-Fused	7.2	85.4	5.16	86.4
ACCDIST-Corr.dist.	-	-	2.66	93.17
ACCDIST-SVM	-	-	1.87	95.18
Human	-	-	-	58.24

Table 10.3: Results of all accent recognition experiments. The figures are percentage EER and percentage recognition accuracy (Acc)

recognition accuracy reported in [18]. We conclude that exploiting linguistic knowledge about how the realization of vowels in particular contexts is indicative of regional accents of British English, gives a significant advantage compared to the purely data-driven approach that is followed in contemporary language ID. There is also an interesting trade-off between the need for a textual transcription of the test material in the ACCDIST approach and its modest computational requirements, and the text-independence but much greater computational requirement of the full language ID system.

Human performance on the accent identification task (58.24% recognition accuracy) is significantly poorer than any of the automatic systems. However, this has also been observed in other studies [13]. Table 10.4 shows the confusion matrix for the human accent recognition experiment, which is largely as one would predict. For example, there is confusion between the two northern English accents, East Yorkshire (eyk) and Lancashire (lan), the two Scottish accents, Glasgow (gla) and Scottish Highlands (shl), and between the two Irish accents , Dublin (roi) and Ulster (uls). The consistent misclassification of the North Wales accent (nwa) as Liverpool (lvp) is explained by the close geographical proximity of Denbigh, the town where the North Wales recordings were made, and Liverpool.

As expected, subjects are better at classifying accents with which they are familiar [13, 21]. Human accent recognition accuracy is 76.2% for accents from regions where the listener has lived, compared with 51.7% for accents from regions where they have not lived, and 71.63% for 'familiar' accents and only 40.2% for 'unfamiliar' accents (according to the listeners' responses to the questionnaire). The good performance for the Birmingham accent, and the overall shape of the confusion matrix, may be influenced by the presence of a disproportionate number of subjects from the Birmingham area in the listener group.

For comparison, table 10.5 shows the corresponding confusion matrix for the acoustic GMM-UBM system, which achieves a similar accent recognition accuracy to the human listeners. The confusions are generally less intuitive. For example, as well as the expected confusion between Lancashire (lan) and East Yorkshire (eyk), there are many other instances of data being incorrectly classified as East Yorkshire or Inner-London, and examples where the Birmingham accent (brm) is incorrectly recognized as seven different accents.

	brm	crn	ean	eyk	gla	ilo	lan	lvp	ncl	nwa	roi	shl	sse	uls	Acc
brm	**30**	.	2	2	.	1	1	1	.	.	.	1	.	1	76.9
crn	.	**15**	6	.	.	1	1	.	1	3	1	.	2	1	48.4
ean	.	8	**12**	.	.	9	2	.	.	2	.	.	4	1	31.6
eyk	.	.	6	**19**	1	1	14	.	3	.	.	.	1	.	42.2
gla	.	.	.	2	**20**	.	.	.	1	.	1	5	2	1	62.5
ilo	2	2	1	1	.	**24**	.	1	.	.	.	.	3	.	70.6
lan	1	.	2	7	.	.	**22**	1	2	.	1	1	.	.	59.5
lvp	.	.	.	.	.	.	.	**28**	3	.	.	.	2	.	84.9
ncl	.	3	.	2	2	.	3	.	**21**	.	.	1	.	.	65.6
nwa	.	.	2	4	.	.	3	11	2	**10**	.	.	3	1	27.8
roi	.	.	.	.	.	.	.	.	.	1	**22**	.	1	6	73.3
shl	2	.	.	1	9	.	.	.	.	1	.	**19**	.	1	57.6
sse	.	.	4	.	.	3	1	1	.	1	.	.	**17**	1	60.7
uls	.	.	1	.	1	.	.	.	.	.	8	1	.	**20**	64.5

Table 10.4: Confusion matrix for human regional accent recognition experiment. see table 7.2 in chapter 7 for accents' abbreviations

The main results for all systems and tasks described in this thesis are also summarized in table A.1 in appendix A.

	brm	crn	ean	eyk	gla	ilo	lan	lvp	ncl	nwa	roi	shl	sse	uls	Acc
brm	**7**	2	1	1	1	2	.	.	.	3	.	.	1	.	38.9
crn	.	**7**	.	.	.	7	.	1	.	1	.	2	.	.	38.9
ean	.	.	**10**	2	.	3	.	.	.	.	.	.	3	.	55.6
eyk	.	1	.	**11**	.	.	3	.	1	2	.	.	.	.	61.1
gla	.	1	.	.	**12**	.	.	.	.	.	.	5	.	.	66.7
ilo	.	2	.	.	.	**12**	1	.	.	2	.	1	.	.	66.7
lan	.	.	.	3	.	1	**13**	.	.	1	.	.	.	.	72.2
lvp	.	.	.	2	.	2	.	**11**	.	3	.	.	.	.	61.1
ncl	.	1	.	2	1	2	.	.	**11**	.	.	1	.	.	61.1
nwa	.	.	.	1	.	3	1	1	.	**12**	.	.	.	.	66.7
roi	.	.	.	.	.	.	.	.	.	.	**14**	.	.	3	82.4
shl	.	.	.	.	.	1	.	.	.	.	.	**17**	.	.	94.4
sse	3	2	.	2	.	2	.	.	.	.	.	.	**6**	.	40.0
uls	.	.	.	1	.	2	.	.	.	1	4	1	.	**8**	47.1

Table 10.5: Confusion matrix for GMM-UBM regional accent recognition experiment. See table 7.2 in chapter 7 for accents' abbreviations

10.3.1 Vowel system for accent recognition

As noted at the beginning of this chapter, D'Arcy in her thesis [119] shows vowel plots (scatter plots of F1 against F2-F1) for all speakers for each ABI-1 region. These plots show that the vowel systems are different between accents and they show considerable consistency within an accent. This provides evidence that the vowel systems may be strong cues for accent recognition. Moreover, Huckvale's ACCDIST system focusses exclusively on vowels. This motivated us to investigate the effect of using only the vowels (and voiced sounds in general) on the performances of our language ID systems.

10.3.1.1 Pitch-based VAD

One way of focusing on voiced sounds is by estimating the pitch (fundamental frequency) for each speech frame and then using the VAD to remove the silence and unvoiced sounds, in the same way described in chapter 3. The pitch is estimated with the auto-correlation method using the standard 'praat' toolkit [2].

Using the pitch-based VAD, the performances of all of our acoustic and fused systems are reported in table 10.6. In spite of the fact that the pitch based VAD not only removes the silence but also the unvoiced sounds, the performances of all acoustic and fused systems are improved with the pitch-based VAD. The percentage EER of the GMM-UBM system is improved by around 11.8% and 12.5% when evaluated on the 30-second segments and SPA recordings, respectively. By fusing all of the acoustic systems (with pitch-based VAD), with the twelve phonotactic systems, the performance is improved to 4.52% EER (89.6% accuracy) compared with 1.87% EER (95.18% accuracy) for the ACCDIST-SVM system.

10.3.1.2 Accent ID using "Careful Words"

The "Careful Words" recordings in ABI-1 corpus (see section 7.2) are read speech of a list of syllables of the form hVd. In other words, the initial and end consonants are /h/ and /d/, respectively and 'V' is a variant of vowels. In order to study how well

[2]http://www.fon.hum.uva.nl/praat/

our language ID systems can distinguish between the regional accents using vowels, only the "Careful Words" recordings in the test sets are used to evaluate our acoustic and phonotactic systems. The performance (EER[%] and accuracy [%]) of all of our systems are presented in columns 6 and 7 of table 10.6. These results show that using only vowels, the performance of all of the acoustic systems is improved. As expected, the performance of the phonotact system is degraded because all of the careful words have exactly the same sequence of words.

In general, these results suggest that the voiced speech segments, and specifically the vowels, have the most discriminative information to distinguish between the British accents.

System	30-second		SPA recordings		Careful Words	
	EER [%]	**Acc** [%]	**EER** [%]	**Acc** [%]	**EER** [%]	**Acc** [%]
GMM-UBM	15.28	60.34	12.98	65.1	8.76	82.18
GMM-SVM	14.3	63.2	11.15	71.82	8.5	83.8
GMM-SVM-GMM	13.0	67.72	9.41	76.11	8.4	84.2
GMM-*uni*-gram	14.95	60.12	13.54	63.8	7.9	83.4
GMM-*bi*-gram	19.69	52.12	18.5	57.83	18.5	59.4
Acoustic-fused	12.33	73.6	8.3	77.32	6.44	88.3
Phonotactics	9.18	74.1	6.5	82.14	14.22	61.5
Acou-Phono-fused	6.4	88.8	4.52	89.6	5.5	88.5

Table 10.6: Performance of the acoustic and fused accent recognition systems using pitch-based VAD. The figures are percentage EER and percentage recognition accuracy (Acc)

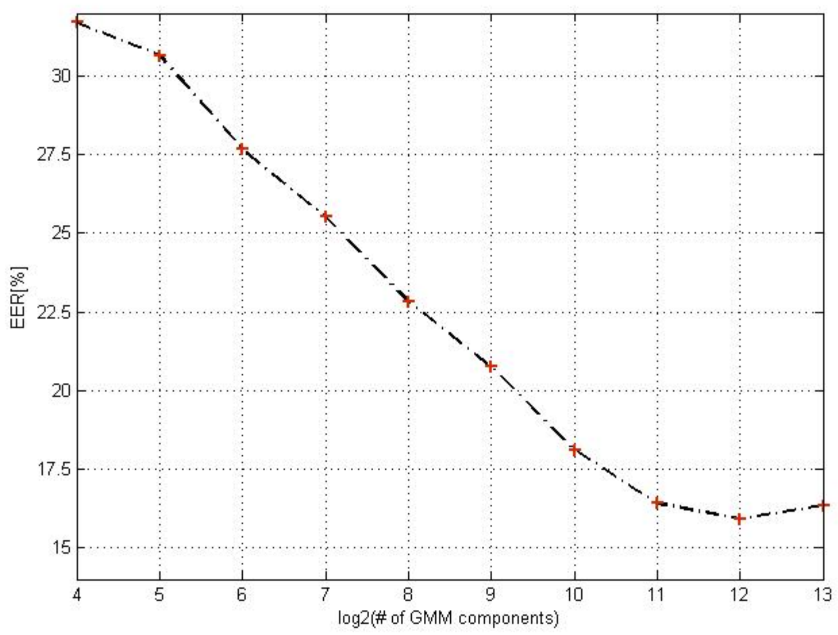


Figure 10.1: Accent recognition performance (EER[%]) for the GMM-UBM system with different number of GMM components.

10.3.2 Number of GMM components

As we have seen, the performance of acoustic accent recognition is significantly degraded by including non-vowel voiced sounds. In order to check that this is not due to the use of too many GMM components for the available training data, we repeated the GMM-UBM experiment using GMMs with 2^N components ($N = 4, ..., 12$). The pitch based VAD is used in all of these experiments.

Although the amount of available training data for accent ID is small compared with the language ID, the performance of the GMM-UBM system improves by increasing the number of GMM components up to 4096 components, as shown in figure 10.1. This confirms what we have already found for language ID.

10.3.3 The effect of speech bandwidth on accent recognition

As we have seen in the previous section, vowels appear to have the most discriminative information between the British regional accents. The fact that the formants of vowels occupy the low frequency region (less than 4 KHz) and the formants of the un-voiced sounds occupy the high frequency region (greater than 4 KHz) motivated us to study the effect of the bandwidth on the accent recognition performance.

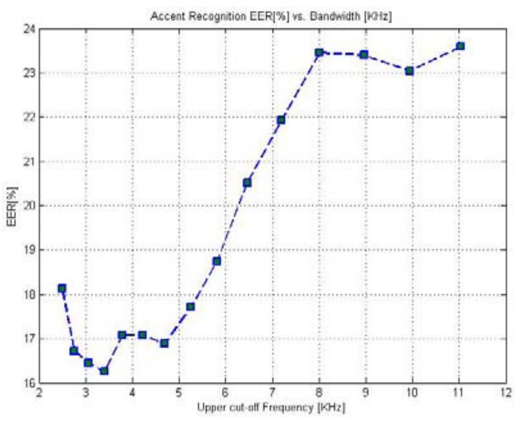
Lu and Dang in [124] investigated the dependency between frequency components and speaker information quantitatively using the F-ratio and mutual information measurements. They found that the most important speaker specific information exist in three different regions on the frequency axes. Specifically, the low frequency region (less than 500 Hz), the region between 4 KHz and 5.5 KHz and the high frequency region around 7.5 KHz. They also found that there is less speaker discriminative information in the region from 500 Hz to 3.5 KHz. Since our objective in accent recognition is to ignore the speaker specific information within a single accent, we have investigated the relationship between the frequency bandwidth and the accent recognition. The aim of this investigation is to find the bandwidth which minimizes the inter-speaker variation and maximizes the desired inter-accent variation.

For this purpose, we used the FFT based front end with 30 Mel frequency triangular filters with a total bandwidth of 11.025 KHz (full bandwidth of speech waveforms in ABI corpus) to extract 19 MFCC and 49 shifted-delta cepstra coefficients at frame rate of 100 frames per second. In order to study the effect of the high frequency region (greater than 3.5 KHz) on accent recognition, the same feature vectors were re-extracted but with the bandwidth reduced from the high frequency end. We reduced the bandwidth by discarding one filter at a time from the highest frequency region and fixing the low frequency cut-off at zero. This process was repeated 15 times, which corresponds to a high frequency cut-off range from 11.025 KHz down to 2.5 KHz. Our GMM-UBM with 2048 components was used to evaluate the accent performance in each case. Figure 10.2(a) shows the relationship between accent recognition performance (EER[%]) and the high frequency cut-offs. The best performance was obtained by setting the high frequency cut-off to around 3.5 KHz. This means that the high frequency region greater than 3.5 KHz contains information (probably speaker specific information) which is not desirable for accent recognition. This result is also consistent with findings of other studies such as [124, 125] that the high frequency region contains the most discriminative speaker information.

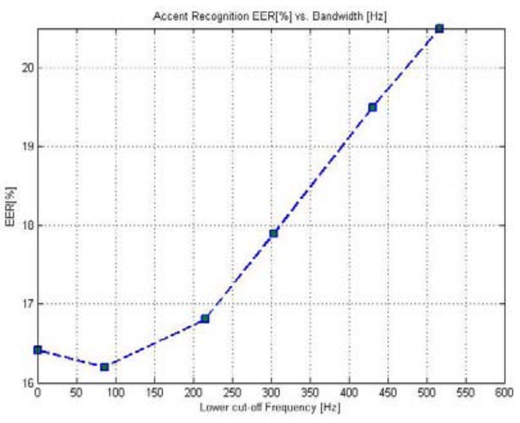
To study the effect of low frequency cut-off on accent recognition, the same procedure was used, but skipping one filter at a time from the lowest frequency region and fixing the high frequency cut-off at 3.5 KHz. This process was repeated six times which corresponds to low frequency cut-offs range from zero to around 516 Hz. The effect of the low frequency components on accent recognition performance is shown in figure 10.2(b). The best performance is obtained with low frequency cut-off of around 86 Hz. By excluding frequency components between 86 Hz and 516 Hz, the accent recognition performance degrades dramatically.

10.4 Summary and conclusion

The objective of this study on regional accents of the British Isles is to measure the ability of a state-of-the-art automatic language ID system to extract the regional accent



(a)



(b)

Figure 10.2: The effect of bandwidth on accent recognition. (a) effect of upper cut-off [KHz] and (b) effect of lower cut-off [Hz]

of a speaker from a short section of speech signal. The performance of our language ID system is compared with human performance, and with other automatic systems based on Huckvale's ACCDIST measure. The ABI-1 corpus of good quality recordings of read speech, representing fourteen different regional accents of spoken British English, is used for our experiments.

For accent recognition, automatic language ID outperforms human listeners. The classification error rate for human listeners is approximately four times greater than that for the language ID system (41.76% compared with 13.6%).

Regional accent recognition appears to be a challenging task for both automatic systems and human listeners. Even though the ABI-1 recordings are good quality read speech (rather than telephone conversational speech), the best accent recognition performance of our language ID system on 30s segments is 7.2% EER (85.4% accuracy) compared with 0.34% EER (99.1% accuracy) and 4.4%EER (93% accuracy) for language ID using the same amount of telephone conversational speech from the NIST 2003 and

2005 evaluations, respectively (see chapter 8). The best regional accent recognition performance is 1.87% EER (95.18% accuracy), which is achieved using the ACCDIST-SVM system and the SPA recordings. The superior performance of the ACCDIST-based systems relative to the language ID system is an interesting example where the explicit use of linguistic knowledge results in a method that outperforms a purely data-driven statistical approach, and with a much lower computational requirement. However, a clear disadvantage of the ACCDIST method is its text dependency, in that transcriptions of the training and test utterances are required. An obvious challenge is to exploit the ideas that motivate ACCDIST without relying on a such a transcription.

Regionally accented speech in the ABI-1 corpus is defined to be speech spoken by an individual who was born in that region and has lived there for all of his or her life. However, even with this residency constraint many subjects' accents exhibit non-regional influences. It seems that naïve native human listeners can correctly place characteristic examples of regionally accented speech but have difficulty in cases where, for example, as a consequence of social or educational factors a subject's accent exhibits strong traits of Standard English. However, the relatively good performances of the automatic systems indicate that correct classification of many of these more subtle instantiations of regional accent is possible. It would be interesting to know how well human listeners can perform given suitable explicit training.

Chapter 11

Human and Computer Recognition of Ethnic Groups From British English speech

As we have seen in chapter 10, spoken British English can be partitioned into a range of regional accents and dialects [1]. However, even within a particular accent region there is variation. For example, people born and raised in different neighborhoods or in different social groups in the same city can often be distinguished by their speech.

In chapter 10, we have successfully adopted the most common techniques used in language ID to regional accent recognition. The purpose of this chapter is to determine whether these same techniques can distinguish between different groups within the same accent. Our language ID system which is developed in chapter 8 and applied to the accent recognition task in chapter 10 is also applied to ethnic group identification. All of the parameters and normalization techniques of our system, which are tuned on the NIST 2003 evaluation data, are kept the same for the ethnic group ID, except the VAD. Because the pitch-based VAD has been found to improve the accent recognition performance significantly (see section 10.3.1), it is also used for our ethnic group ID. A further investigation about the best parameters and best techniques which may affect the performance of ethnic group ID is also conducted in this chapter.

For all of the experiments described in this chapter, we used the VaB corpus of telephone conversational speech (section 7.3). The two largest ethnic groups in Birmingham (UK) are the 'Asian' and 'White' communities, and the VaB corpus includes a substantial amount of data from each group. The task is to assign a subject to one of these two groups using a forty second sample of his or her telephone conversational speech. For the Asian group we only used data from second generation subjects. Although we refer to this as 'ethnic group' classification, and to the two groups as 'Asian' and 'White', it is clear that we are actually concerned with differences between the patterns of pronunciation and language usage between the two communities, and not explicitly with ethnicity. We compare the performance of our language ID system with that of human listeners on this task. Related human perceptual studies for varieties of American English are reported in [23] and [22].

The main results of this work also appears in [126] and [121].

11.1 Automatic classification systems

Because transcriptions are not available for the VaB corpus, it is not possible to apply the ACCDIST-based systems, which were described earlier in chapter 10, to the ethnic group ID. Therefore, only our language ID systems (phonotact and acoustic) are used for automatic ethnic group identification.

After removing zero occupancy components, the dimensions of the n-gram vectors for

Phone Recognizer \ n-gram	1-gram	2-gram	3-gram	4-gram
English	39	1521	43652	360360
Czech	42	1764	32464	275350
Hungarian	58	3364	31640	216413
Russian	52	2704	33294	213197

Table 11.1: Dimensions of n-gram vectors for each phonotactic system

each of the sixteen phonotactic systems are shown in table 11.1.

Because there is no development data to train the fusing coefficients, we divided the 315 speakers in the test set into two subsets; one with 157 speakers and the other with 158 speakers. The ethnic group and gender of speakers are distributed equally in both sets. One subset is used to find the coefficients to be used in fusing the systems on the second subset, and vice versa. The fused scores are then combined together and the final performance is estimated. This method was used to obtain all of the fusion results in this chapter.

11.2 Human performance

To provide a baseline against which the automatic ethnic group recognition systems could be compared, a web-based human perceptual experiment was conducted using exactly the same 315 test utterances that were used for automatic classification. Eight listeners who were familiar with the Birmingham accent took part in the experiment. As with the human accent recognition experiment, in chapter 10, the listeners had no formal background in phonetics or linguistics and no explicit training was given. Two subjects listened to all of the 315 test utterances, and six subjects listened to sets of 20 utterances. For each utterance, subjects were asked to identify the ethnic group (Asian or White), to indicate their confidence in their decision, to estimate the age of the speaker, and to indicate the factors (acoustic quality, use of particular words or phrases, intonation, grammar, or other factors) that influenced their decision. The human listeners scored an average error rate of 8.72% for the ethnic group identification task.

11.3 Results and discussion

The experimental results for the sixteen phonotactic systems are presented in table 11.2. The results are presented as percentage EER and accuracy on the 315 test utterances.

For each n-gram, the performances of the phonotactic systems with different phone recognizers are almost the same. However, since the VaB corpus is British English speech, the phonotactic systems with British English phone recognizers were expected to perform better than the other phone recognizers for ethnic group ID. A possible explanation for the fact that it does not, is that the Phone Error Rate (PER) of our phone recognizer is around 42% which is relatively high compared with 24%, 33% and 39% PER for the Czech, Hungarian and Russian phone recognizers [1], respectively. As with language and accent ID, combining phonotactic systems with different phone recognizers improves the performance for all n-gram systems. In addition, fusing the 2-gram, 3-gram and 4-gram systems for each single phone recognizer improves the performance. The best performance (13.0%EER, 87.2% accuracy) is obtained by fusing all of the sixteen phonotactic systems

[1]http://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context, November 2011

Phone Recognizer	English	Czech	Hungarian	Russian	Fused
1-gram	39.34 *56.5*	44.6 *55.3*	38.14 *61.9*	43 *56.8*	31.38 *68.5*
2-gram	18.8 *81.2*	23.65 *76.5*	20.68 *79.4*	17.96 *81.9*	16.08 *83.6*
3-gram	18.35 *82.4*	19.84 *80*	18.69 *81.3*	20.31 *79.6*	17.52 *84*
4-gram	19.67 *80.6*	21.36 *78.5*	24.61 *75.6*	26.37 *73.7*	18.11 *82.3*
2,3,4-fused	16.55 *83.1*	15.61 *83.5*	18.35 *81.9*	18.25 *81.6*	13.0 *87.2*

Table 11.2: Performance EER[%] (*Accuracy[%]*) of phonotactic ethnic groups recognizers using English, Czech, Hungarian and Russian phone recognizers.

together.

The performances of the acoustic and fused systems are presented in table 11.3. All of the acoustic systems achieve a similar level of performance, with the GMM-UBM giving the lowest EER at 15.1%. Fusion of the acoustic systems ('Acoustic-fused') results in an EER of 8.22%, a reduction of approximately 50% in EER relative to individual acoustic systems. This indicates that there is some orthogonality between the different acoustic systems for the ethnic group classification task. Despite the fact that the GMM-SVM system involves discriminative training it performs worse than the GMM-UBM system on this task. The final fused system ('Acoustic-phonotactic-fused') scores 4.0% EER (94.3% accuracy).

Acoustic system	EER [%]	Accuracy [%]
GMM-UBM	15.1	85
GMM-SVM	19.8	83.67
GMM-SVM-GMM	19	84.35
GMM-uni-gram	17	83
GMM-bi-gram	17.78	82.16
Acoustic-Fused	8.22	90.56
Phonotactic	13.0	87.2
Phono-Acoustic-Fused	4.0	94.3

Table 11.3: Performance (EER [%] and Accuracy [%]) of the acoustic based 'ethnic groups' ID systems and fused system

The main results for all systems and tasks described in this thesis are also summarized in table A.1 in appendix A.

11.4 The effect of using the SDC and pitch-based VAD on the ethnic group ID

We have seen earlier for language and accent recognition that there are many parameters and techniques that may improve or degrade recognition performance. The only way to find out what is the effect of changing these parameters and techniques on recognition performance for a specific recognition task, is by practical experiments. Therefore, we used the simple GMM-UBM system (with 4096 components) to answer the following questions for ethnic group ID:

- what number of MFCCs gives the best performance?
- which is better, in terms of performance, the traditional Δ parameters or the SDC coefficients?
- which of the feature normalization techniques gives the best performance?
- which of the score normalization techniques gives the best performance?
- what is the effect of using the pitch-based and the energy-based VAD on ethnic group ID performance?

After running a number of experiments, we ended up with the same configuration for language and accent recognitions (see chapters 8 and 10) , except for the SDC coefficients. Using the traditional Δ parameters rather than SDCs has been found to improve the ethnic group ID performance. In addition, as for accent recognition, the energy-based VAD has also been found to degrade ethnic group ID performance (table 11.4).

Table 11.4 shows the performances (EER[%]) of the GMM-UBM system when combining 19 MFCCs with 19 traditional Δ (over 5 frames) parameters (38 dimensions) and also when combining the 19 MFCCs with the 49 SDC coefficients using 7-3-1-7 configuration (68 dimensions) . For each case, the system is trained and evaluated when using the energy-based VAD and the pitch-based VAD. Recall that in the case of pitch VAD, both silence and unvoiced frames are removed.

As shown in the results in table 11.4, a significant improvement is obtained by using the pitch-based VAD, which keeps the voiced frames only. This is consistent with our result for the regional accent recognition that the voiced sounds, specifically vowels, have

Features	Energy-based VAD	Pitch-based VAD	GMM-UBM	
			EER[%]	Acc[%]
MFCC+Δ	$\surd$	–	18.24	81.7
	–	$\surd$	13.33	87
MFCC+SDC	$\surd$	–	20.8	79.5
	–	$\surd$	15.1	84.6

Table 11.4: Performance (EER[%] and accuracy[%]) comparison of using the Δ parameters and the SDCs, and for each case, using the energy-based and the pitch-based VAD with the GMM-UBM ethnic group ID recognizer.

the best information to discriminate between closely related variations of British English, including regional accents and ethnic groups within a single accent.

Unexpectedly, using the traditional Δ parameters outperformed the SDCs, even though the later have been shown to improve language ID performance. According to this result, and to investigate the effect of using traditional Δ parameters rather than SDCs on all of our acoustic systems, they are re-trained and re-evaluated using the Δ parameters. The performance of all of the acoustic systems is improved by an average of 13% (EER) and 2.8% (accuracy). Fusing the acoustic systems together improves the performance to 7.28% (EER) and 92.7% (accuracy). The best performance (EER =3.57% and accuracy =96.51%) is obtained when fusing all of the acoustic systems with the sixteen phonotactic systems. This compares with 9.76% error rate (90.24% accuracy) for the human listeners.

The fact that the phonotactic and acoustic components contribute approximately equally to automatic ethnic group identification performance is interesting. Subjectively, it is evident from listening to the recordings that the speech quality is different for the two groups, and one would expect this to be exploited by the acoustic systems. However it is also evident that the Asian recordings are characterized by the more frequent use of particular English words and the almost exclusive use of some non-English words (for example, people's names), which one would expect to be exploited by the phonotactic system. However, it seems that both of these phenomena contribute approximately equally to automatic classification performance. In contrast, the human listeners reported that in 75.5% of the tests the 'quality' of the speech contributed to their judgement, compared with 28% for the occurrence of specific words or phrases, 23.8% for intonation, 11.8% for grammar, and 0.6% for 'other factors'.

11.5 Summary and conclusion

In this chapter we investigated whether techniques used for language and accent recognition are able to distinguish between talkers from different ethnic (social) groups within a single regional accent. The 2001 census of England and Wales identifies two main ethnic groups in the city of Birmingham, UK, namely Asian and white. These groups are well represented in Voices across Birmingham, a corpus of recordings of telephone conversational speech between individuals in the city. In this study we only consider speech from those participants who were born in Birmingham. The results of applying various acoustic and phonotactic language ID systems to this problem are reported. The best phonotactic and acoustic systems score EERs of 13% and 8.22%, respectively. The overall best performance (3.57% EER) is achieved using a system which fuses the outputs of a combination of these phonotactic and acoustic systems that use the traditional Δ parameters and pitch-based VAD. This result is much better than anticipated and compares with an error rate of 8.72% for human listeners The fact that it is possible to decide automatically which ethnic group within a particular accent group an individual belongs to, and to achieve this using as little as 40s of data, has interesting implications for automatic speech recognition. First, it confirms that there are significant acoustic and phonotactic differences even within a homogeneous accent group. Second, it shows that these differences are sufficiently large to be detected automatically. Hence it may be possible to identify suitable acoustic, lexical and even grammatical models automatically for rapid adaptation.

Chapter 12

Conclusion and Future Work

This chapter summarizes the contributions of this thesis and draws some conclusions about this research. Finally, it outlines some possible directions for future work.

12.1 Conclusions

The summary and conclusion of each individual chapter are presented in its last section. This section gives an overview of the main conclusions of the whole thesis.

Back to the introduction in chapter 1, the main objective of this thesis is to measure the ability of a state-of-the-art automatic language ID system to extract two particular types of paralinguistic information from a speech signal, specifically the regional accent of the speaker, and the ethnic group to which he or she belongs. In both cases, the performance of the language ID system is compared with human performance, and for regional accent recognition, with other automatic systems based on Huckvale's ACCDIST measure. The ABI-1 corpus of good quality recordings of read speech, representing fourteen different regional accents of spoken British English, is used for our experiments in accent recognition. The VaB corpus of telephone conversational speech between subjects who were born and live in the city of Birmingham (UK) is used for ethnic group recognition.

For both regional accent and ethnic group recognition, automatic language ID outperforms human listeners. The recognition error rate for human listeners is approximately four times greater than that for the language ID system for regional accent recognition (41.76% compared with 13.6%), and two times greater for ethnic group recognition (9.76% compared with 5.7%).

Regional accent recognition appears to be a challenging task for both automatic systems and human listeners. Even though the ABI-1 recordings are good quality read speech (rather than telephone conversational speech), the best accent recognition performance of our language ID system on 30s segments is 7.2% EER (85.4% accuracy) compared with 0.34% EER (98.42% accuracy) for language ID using the same amount of telephone conversational speech from the NIST 2003 evaluation (see chapter 8). The best regional accent recognition performance is 1.87% EER (95.18% accuracy), which is achieved using the ACCDIST-SVM system and the SPA recordings. The superior performance of the ACCDIST-based systems relative to the language ID system is an interesting example where the explicit use of linguistic knowledge results in a method that outperforms a purely data-driven statistical approach, and with a much lower computational requirement. However, a clear disadvantage of the ACCDIST method is its text dependency, in that transcriptions of the training and test utterances are required. An obvious challenge is to exploit the ideas that motivate ACCDIST without relying on a such a transcription.

Regionally accented speech in the ABI-1 corpus is defined to be speech spoken by an individual who was born in that region and has lived there for all of his or her life. However, even with this residency constraint many subjects' accents exhibit non-regional influences. It seems that naïve native human listeners can correctly place characteristic examples of regionally accented speech but have difficulty in cases where, for example, as a consequence of social or educational factors a subject's accent exhibits strong traits of Standard English. However, the relatively good performances of the automatic systems indicate that correct classification of many of these more subtle instantiations of regional accent is possible. It would be interesting to know how well human listeners can perform given suitable explicit training.

Intuitively, the ethnic group classification task appears to be more difficult than accent recognition, even though it is a two-class problem, since the classes share some aspects of the same regional accent, and the data is telephone conversational speech. However, the acoustic and phonotactic components of our automatic language ID system score recognition accuracies of 92.7% and 87.2%, respectively, and the overall best performance is 94.3% accuracy, achieved by fusing all of the acoustic and phonotactic subsystems. This result is much better than expected and compares with an accuracy of 90.24% for human listeners. As in the case of regional accent recognition, it would be interesting to know how well human listeners would perform if they were given explicit training for this task.

We also investigated the effect of focusing our language ID system on vowel (or more generally 'voiced' phonemes) differences between accents and ethnic groups. This was achieved by using a pitch-based VAD and in case of accents, choosing the "careful words" recordings from ABI-1 as test data. The results show that using vowels, regional accent and ethnic group recognition performances (%EER) are improved by around 12.4% and 27.4%, respectively.

The fact that it is possible to access these types of paralinguistic information using as little as 30s of data has interesting implications for automatic speech recognition.

It confirms that there are significant acoustic and phonotactic differences between, and even within, regional accents, and it shows that these differences are sufficiently large be detected automatically. Hence it may be possible to use these technologies to identify suitable acoustic, lexical and even grammatical models automatically, as a first step towards rapid adaptation.

Although it is difficult to make direct comparisons, it seems that our language ID system performs better on the language recognition and the ethnic group recognition tasks than on the regional accent recognition task, even though the former are based on conversational speech recorded over a telephone channel while the latter involves good quality recordings of read speech. It seems likely that, unlike automatic speech recognition, the availability of natural conversational speech may be advantageous for these types of paralinguistic tasks.

Returning to our original premise, we conclude from the results presented in this thesis that the distributions of acoustic feature vectors, and phone n-grams, corresponding to different regional accents of English or different ethnic groups within an accent, are sufficiently distinct to enable pattern recognition methods from language ID to be applied successfully to automatic regional accent and ethnic group classification. From a broader perspective, this raises the possibility of applying similar techniques to the automatic classification of other paralinguistic phenomena.

In the other part of this thesis, we investigated the effectiveness of the GPUs to accelerate the intensive computation required in language ID. We used the GPU to accelerate the computation of the most consuming components of the standard language ID techniques, specifically conditional probability calculation and spectral analysis algorithms.

Our result shows that the GPUs seem to be very effective for speeding up the language ID computation. Using a GPU, the overall computation of our GMM-UBM language ID system was accelerated by a factor eleven, which makes it possible to try new techniques which would otherwise not be possible due to their high computational load. For example, our MLM-*uni*-gram system comprises 24,576 GMM components. The performance of this system is comparable with the state-of-the-art, but its evaluation would have been impossible without a GPU implementation.

Our results also indicate that an FIR-based front-end has substantial computational advantages in the case of a CPU implementation and major advantages when the implementation utilizes a GPU. For example, for 30 minutes of speech, FIR on GPU is nine times faster than FFT on CPU.

12.2 Future work

This thesis describes an investigation of modeling phonotactic and acoustic cues from a speech signal with standard language ID techniques to recognize the fourteen accents in the ABI-1 corpus and the main two ethnic groups within a single accent, specifically the Birmingham accent. However, these are not the only available cues from speech; and based on what is known about how humans recognize accents, a number of other cues might also be exploited. Their use in an automatic accent recognition framework offers a number of possible avenues for future work. It is also important to acknowledge some of the limitations of the studies reported in this thesis, and future work should also involve addressing some, if not all, of these limitations. Some avenues of work described in this thesis are listed below.

- All of the language ID experiments reported in this thesis were developed and tuned on NIST 2003 and NIST 1996 evaluation data sets and evaluated on the NIST 2005 evaluation data set. The NIST 2005 LRE is relatively old compared with NIST 2007, NIST 2009 and NIST 2011 LREs. This in turn limits the language ID work reported in this thesis to an exploratory investigation that unearthed interesting hypotheses that need to be validated on recent NIST evaluations. Therefore, evaluating our language ID system on the NIST 2007, 2009 and 2011 LREs is a natural extension for this research.

- In addition to phonotactic and acoustic features for language ID, prosodic features such as pitch and energy contours are known to be good paralinguistic cues in speech. Fusing the phonotactic and acoustic systems with a prosodic based system may improve the overall performance, because these three different features are considered to be mutually complementary.

- Some other techniques for language ID, such as removing some gender and inter-speaker variation by normalizing the length of the vocal tract using the VTLN technique should also be interested. De-correlating and reducing the dimensionality of the acoustic features, especially the SDC features, with Heteroscedastic Linear Discriminant Analysis (HLDA) is also interesting for future work.

- All of the accent recognition experiments reported in this thesis were performed on the ABI-1 corpus. This corpus is limited in size, contained recorded speech for only fourteen accents and consisting of read speech. Applying the same approaches on a larger database and natural speech collected through the telephone network is interesting future work. It is not clear whether telephone conversational speech would make the problem harder or easier.

- For accent recognition, and when the transcription of the speech is available, Huckvale's ACCDIST measure was proven to outperform traditional text-independent methods which depend on absolute spectral properties or sequence analysis. Extending this method to other tasks such as language ID and ethnic group ID is another interesting challenge for the future. Removing the text-dependency limitation in the ACCDIST based system by relying on a high accuracy phone recognizer to do phonetic segmentation is a possible way forward that should be investigated. In addition, weighting the speaker distance tables with a discriminative weighting technique similar to the LLR weighting which was successfully used in the n-gram systems also has potential, as does using the concept of inter-session compensation to compensate inter-speaker variability in the ACCDIST speaker distance tables.

- Human accent recognition performance was found to be higher for the accents of regions where listeners have lived and also for accents that they consider themselves familiar with. This suggests that training human listeners on the target accents training data before they do the classification test will improve their classification rate.

- For accent recognition, projecting speakers' speech into high-dimensional space prior to classification performs well. Therefore a more interesting challenge for the future is to develop continuous space representation of speakers and accent, such that subjects who are close in this space speak in a similar manner and, from the perspective of automatic speech recognition, can be characterized by similar sets of model parameters.

Bibliography

[1] J.C. Wells. *Accents of English, volumes 1, 2 and 3.* Cambridge University Press, 1982.

[2] A. Hughes, P. Trudgill, and D. Watt. *English accents and dialects: an introduction to the social and regional varieties of English in the British Isles.* Hodder Arnold, fourth edition, 2005.

[3] S. Elmes. *Talking for Britain: A Journey through the nation's dialects.* Penguin Books, 2005.

[4] O. Koller, A. Abad, I. Trancoso, and C. Viana. Exploiting variety-dependent phones in Portuguese variety identification applied to broadcast news transcription. In *Eleventh Annual Conference of the International Speech Communication Association, InterSpeech'10*, pages 749–752, 2010.

[5] J.J. Humphries and P.C. Woodland. Using accent-specific pronunciation modelling for improved large vocabulary continuous speech recognition. In *Fifth European Conference on Speech Communication and Technology, EuroSpeech'97*, pages 2367–2370, 1997.

[6] M. Tjalve and M. Huckvale. Pronunciation variation modelling using accent fea-

tures. In *9th European Conference on Speech Communication and Technology, InterSpeech'05*, pages 1341–1344, Lisbon, Portugal, 2005.

[7] F. Biadsy, J. Hirschberg, and M. Collins. Dialect recognition using a phone-GMM-supervector-based SVM kernel. In *Eleventh Annual Conference of the International Speech Communication Association, InterSpeech'10*, pages 753–756, 2010.

[8] J. Yamagishi, B. Usabaev, S. King, O. Watts, J. Dines, J. Tian, Y. Guan, R. Hu, K. Oura, Y.J. Wu, et al. Thousands of voices for HMM-based speech synthesis–analysis and application of TTS systems built on various ASR corpora. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):984–1004, 2010.

[9] M.A. Zissman. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Speech and Audio Processing*, 4(1):31–44, 1996.

[10] M.A. Zissman, T.P. Gleason, D.M. Rekart, and B.L. Losiewicz. Automatic dialect identification of extemporaneous conversational, latin American Spanish speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'96*, volume 2, pages 777–780, 1996.

[11] F.S. Richardson, WM Campbell, and P.A. Torres-Carrasquillo. Discriminative n-gram selection for dialect recognition. In *Tenth Annual Conference of the International Speech Communication Association, InterSpeech'09*, pages 192–195, 2009.

[12] D.R. Miller and J. Trischitta. Statistical dialect classification based on mean phonetic features. In *Fourth International Conference on Spoken Language, ICSLP'96.*, volume 4, pages 2025–2027, 1996.

[13] L.M. Arslan and J.H.L. Hansen. Language accent classification in american English. *Speech Communication*, 18(4):353–367, 1996.

[14] C. Teixeira, I. Trancoso, and A. Serralheiro. Recognition of non-native accents. In *Fifth European Conference on Speech Communication and Technology, EuroSpeech'97*, pages 2375–2378, 1997.

[15] M. Lincoln, S. Cox, and S. Ringland. A comparison of two unsupervised approaches to accent identification. In *Fifth Int. Conf. on Spoken Language Processing, ICSLP'98*, pages 109–112. Citeseer, 1998.

[16] R. Huang, J.H.L. Hansen, and P. Angkititrakul. Dialect/accent classification using unrestricted audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(2):453–464, 2007.

[17] P. Angkititrakul and J.H.L. Hansen. Advances in phone-based modeling for automatic accent classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):634–646, 2006.

[18] M. Huckvale. Accdist: an accent similarity metric for accent recognition and diagnosis. *Muller, C., (ed.) Speaker Classification II. Springer Verlag: Berlin/ Heidelberg, Germany*, pages 258–275, 2007.

[19] W.J. Barry, C.E. Hoequist, and F.J. Nolan. An approach to the problem of regional accent in automatic speech recognition. *Computer Speech and Language*, 3(4):355–366, 1989.

[20] C. Woehrling and P.B. Mareuil. Identification of regional accents in French per-

ception and categorization. In *Ninth International Conference on Spoken Language Processing, InterSpeech'06*, pages 1511–1514, 2006.

[21] A. Ikeno and J.H.L. Hansen. Perceptual recognition cues in native English accent variation: Listener accent, perceived accent, and comprehension. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'06.*, volume 1, pages 401–404, 2006.

[22] T. Purnell, W. Idsardi, and J. Baugh. Perceptual and phonetic experiments on American English dialect identification. *Journal of Language and Social Psychology*, 18(1):10–30, 1999.

[23] J.H. Walton and R.F. Orlikoff. Speaker race identification from acoustic cues in the vocal signal. *Journal of Speech and Hearing Research*, 37(4):738–745, 1994.

[24] S.M. D'Arcy, M.J. Russell, S.R. Browning, and M.J. Tomlinson. The accents of the British Isles (ABI) corpus. In *Modélisations pour l'Identification des Langues, MIDL Paris*, pages 115–119, 2005.

[25] M. Schroeder and BS Atal. Code-excited linear prediction (CELP): High-quality speech at very low bit rates. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'85*, volume 10, pages 937–940, 1985.

[26] http://en.wikipedia.org/wiki/flops/, accessed may, 2011.

[27] J. Fung and S. Mann. Computer vision signal processing on graphics processing units. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'04*, volume 5, pages V–93, 2004.

[28] U. Erra. Toward real time fractal image compression using graphics hardware. *Advances in Visual Computing*, pages 723–728, 2005.

[29] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A.E. Lefohn, and T.J. Purcell. A survey of general-purpose computation on graphics hardware. In *Computer graphics forum*, volume 26, pages 80–113, 2007.

[30] P. Cardinal, P. Dumouchel, G. Boulianne, and M. Comeau. GPU accelerated acoustic likelihood computations. In *Ninth Annual Conference of the International Speech Communication Association, InterSpeech'08*, pages 964–967, 2008.

[31] P.R. Dixon, T. Oonishi, and S. Furui. Fast acoustic computations using graphics processors. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'09.*, pages 4321–4324, 2009.

[32] K. Gupta and J.D. Owens. Three-layer optimizations for fast GMM computations on GPU-like parallel processors. In *IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU'09*, pages 146–151, 2009.

[33] N.S.L.P. Kumar, S. Satoor, and I. Buck. Fast parallel expectation maximization for Gaussian mixture models on GPUs using CUDA. In *1th IEEE International Conference on High Performance Computing and Communications*, pages 103–109, 2009.

[34] E. Singer, P.A. Torres-Carrasquillo, T.P. Gleason, WM Campbell, and D.A. Reynolds. Acoustic, phonetic, and discriminative approaches to automatic language identification. In *Eighth European Conference on Speech Communication and Technology, EuroSpeech'03*, pages 1345–1348, 2003.

[35] L. Burget, P. Matejka, and J. Cernocky. Discriminative training techniques for acoustic language identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'06*, pages 209–212, 2006.

[36] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *EuroSpeech'97*, pages 1895–1898, 1997.

[37] James P. Egan. *Signal detection theory and ROC analysis.* Academic Press, 1975.

[38] Y.K. Muthusamy, N. Jain, and R.A. Cole. Perceptual benchmarks for automatic language identification. In *IEEE International conference on Acoustics, Speech, and Signal Processing, ICASSP'94*, pages 333–336, 1994.

[39] D.A. van Leeuwen, M. de Boer, and R. Orr. A human benchmark for the NIST language recognition evaluation 2005. *IEEE Odyssey'08: The Speaker and Language Recognition Workshop*, 2008.

[40] A.F. Martin and M.A. Przybocki. NIST 2003 language recognition evaluation. In *Eighth European Conference on Speech Communication and Technology, EuroSpeech'03*, pages 1341–1344, 2003.

[41] National institute of standard and technology, http://www.nist.gov.

[42] J.S. Garofolo, T. Robinson, and J.G. Fiscus. The development of file formats for very large speech corpora: Sphere and shorten. In *IEEE International conference on Acoustics, Speech, and Signal Processing, ICASSP'94*, pages 113–116, 1994.

[43] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.

[44] L.R. Rabiner and R.W. Schafer. *Digital processing of speech signals.* Prentice-hall Englewood Cliffs, NJ, 1978.

[45] J. McClellan and T. Parks. A united approach to the design of optimum FIR linear-phase digital filters. *IEEE Transactions on Circuit Theory*, 20(6):697–701, 1973.

[46] A.V. Oppenheim and R.W. Schafer. *Discrete-time signal processing.* Prentice hall Englewood Cliffs, NJ:, 1989.

[47] F.J. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[48] JF Kaiser. Nonrecursive digital filter design using the I0-sinh window function. In *Proc. IEEE Int. Symp. Circuits Syst*, volume 3, pages 20–23, 1974.

[49] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput*, 19(90):297–301, 1965.

[50] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition.* Prentice-Hall, 1993.

[51] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975.

[52] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[53] Y. Yan and E. Barnard. Experiments for an approach to language identification with conversational telephone speech. In *IEEE International conference on Acoustics, Speech, and Signal Processing, ICASSP'96*, pages 789–792, 1996.

[54] R. Chengalvarayan. Robust energy normalization using speech/nonspeech discriminator for German connected digit recognition. In *Sixth European Conference on Speech Communication and Technology, EuroSpeech'99*, pages 61–64, 1999.

[55] A. Benyassine, E. Shlomot, H.Y. Su, D. Massaloux, C. Lamblin, and J.P. Petit. ITU-T Recommendation G. 729 Annex B: a silence compression scheme for use with G. 729 optimized for V. 70 digital simultaneous voice and data applications. *Communications Magazine, IEEE*, 35(9):64–73, 1997.

[56] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(1):52–59, 1986.

[57] P.A. Torres-Carrasquillo, E. Singer, M.A. Kohler, R.J. Greene, D.A. Reynolds, and JR Deller. Approaches to language identification using Gaussian mixture models and shifted delta cepstral features. In *Int. Conf. Spoken Language Processing, ICSLP'02*, volume 2, pages 89–92, Denver, USA, 2002. Citeseer.

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

[59] B.S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America*, 55(6):1304–1312, 1974.

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

[61] O. Viikki and K. Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1-3):133–147, 1998.

[62] J. Pelecanos and S. Sridharan. Feature warping for robust speaker verification. In *Odyssey'01: The Speaker and Language Recognition Workshop, Crete, Greece*, pages 213–218. Citeseer, 2001.

[63] W.M. Campbell, J.P. Campbell, T.P. Gleason, D.A. Reynolds, and W. Shen. Speaker verification using support vector machines and high-level features. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2085–2094, 2007.

[64] W.M. Campbell, J.P. Campbell, D.A. Reynolds, D.A. Jones, and T.R. Leek. Phonetic speaker recognition with support vector machines. *Advances in Neural Information Processing Systems*, 16:1377–1384, 2004.

[65] A. Hanani, M. Carey, and M.J. Russell. Improved language recognition using mixture components statistics. In *Eleventh Annual Conference of the International Speech Communication Association, InterSpeech'10, Makuhari, Chiba, Japan*, pages 741–744, 2010.

[66] J.L. Gauvain, A. Messaoudi, and H. Schwenk. Language recognition using phone

latices. In *Eighth International Conference on Spoken Language Processing, InterSpeech'04*, pages 25–28, 2004.

[67] L.F. Zhai, M. Siu, X. Yang, and H. Gish. Discriminatively trained language models using support vector machines for language identification. In *IEEE Odyssey'06: The Speaker and Language Recognition Workshop*, pages 1–6, 2006.

[68] S. Nakagawa, Y. Ueda, and T. Seino. Speaker-independent, text-independent language identification by HMM. In *Second International Conference on Spoken Language Processing, ICSLP'92*, pages 1011–1014, 1992.

[69] M.A. Zissman. Automatic language identification using Gaussian mixture and hidden Markov models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'93*, pages 399–402, 1993.

[70] W. Campbell, T. Gleason, J. Navratil, D. Reynolds, W. Shen, E. Singer, and P. Torres-Carrasquillo. Advanced language recognition using cepstra and phonotactics: MITLL system performance on the NIST 2005 language recognition evaluation. In *IEEE Odyssey'06: The Speaker and Language Recognition Workshop*, pages 1–8, 2006.

[71] W. Zhang, B. Li, D. Qu, and B. Wang. Automatic language identification using support vector machines. In *8th IEEE International Conference on Signal Processing, ICSP'06*, volume 1, pages 16–20, 2006.

[72] F. Castaldo, D. Colibro, E. Dalmasso, P. Laface, and C. Vair. Acoustic language identification using fast discriminative training. In *Eighth Annual Conference of the*

International Speech Communication Association, InterSpeech'07, pages 346–349, 2007.

[73] J.R. Deller Jr., P. A. Torres-Carrasquillo, and D.A. Reynolds. Language identification using Gaussian mixture model tokenization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'02*, volume 1, pages 757–760, 2002.

[74] X. Yang and M. Siu. N-best tokenization in a GMM-SVM language identification system. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'07*, volume 4, pages IV–1005.

[75] G.J. McLachlan and D. Peel. *Finite mixture models.* Wiley-Interscience, 2000.

[76] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[77] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 881–892, 2002.

[78] C.M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[79] D. Qu and B. Wang. Discriminative training of GMM for language identification. In *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[80] J.L. Gauvain and C.H. Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, 1994.

[81] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn. Speaker verification using adapted Gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[82] V.N. Vapnik. Support vector learning machines–tutorial at nips97. *Denver (CO), December*, 1997.

[83] W.M. Campbell, E. Singer, P.A. Torres-Carrasquillo, and D.A. Reynolds. Language recognition with support vector machines. In *The Speaker and Language Recognition Workshop, ODYS'04*, pages 285–288, 2004.

[84] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and kernel methods MATLAB toolbox, perception systèmes et information, insa de rouen, rouen, france, (http://asi.insa-rouen.fr/enseignants/ arakotom/toolbox/), 2005.

[85] W.M. Campbell. Generalized linear discriminant sequence kernels for speaker recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'02*, volume 1, pages I–161, 2002.

[86] P.J. Moreno and P.P. Ho. A new SVM approach to speaker identification and verification using probabilistic distance kernels. In *Eighth European Conference on Speech Communication and Technology, EuroSpeech'03*, volume 3, pages 2965–2968, 2003.

[87] W.M. Campbell, DE Sturim, and DA Reynolds. Support vector machines us-

ing GMM supervectors for speaker verification. *IEEE Signal Processing Letters*, 13(5):308–311, 2006.

[88] R. Kondor and T. Jebara. A kernel between sets of vectors. In *International Conference on Machine Learning, ICML'03*, volume 20, page 361, 2003.

[89] WM Campbell. A covariance kernel for SVM language recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'08*, pages 4141–4144, 2008.

[90] WM Campbell and Z.N. Karam. A framework for discriminative SVM/GMM systems for language recognition. In *Tenth Annual Conference of the International Speech Communication Association, InterSpeech'09*, pages 2195–2198, 2009.

[91] D.A. Reynolds. Channel robust speaker verification via feature mapping. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'03*, volume 2, pages II–53, 2003.

[92] Puming Zhan and Alex Waibel. Vocal tract length normalization for large vocabulary continuous speech recognition. Technical report, CMU COMPUTER SCIENCE TECHNICAL REPORTS, 1997.

[93] D. Klatt. A digital filter bank for spectral matching. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'76*, volume 1, pages 573–576, 1976.

[94] MJF Gales and S. Young. An improved approach to the hidden Markov model

decomposition of speech and noise. In *EEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'92*, volume 1, pages 233–236, 1992.

[95] C. Vair, D. Colibro, F. Castaldo, E. Dalmasso, and P. Laface. Channel factors compensation in model and feature domain for speaker recognition. In *IEEE Odyssey'06: The Speaker and Language Recognition Workshop*, pages 1–6, 2006.

[96] P. Kenny and P. Dumouchel. Disentangling speaker and channel effects in speaker verification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings, ICASSP'04*, volume 1, pages I–37, 2004.

[97] N. Brummer. Spescom datavoice NIST 2004 system description. *NIST Speaker Recognition Evaluation 2004*, 2004.

[98] M.W. Mason, R.J. Vogt, B.J. Baker, and S. Sridharan. Data-driven clustering for blind feature mapping in speaker verification. In *Sixth Annual Conference of the International Speech Communication Association, InterSpeech'05*, pages 3109–3112. International Speech Communication Association (ISCA), 2005.

[99] V. Wan and W.M. Campbell. Support vector machines for speaker verification and identification. In *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 2, pages 775–784, 2000.

[100] E. Wong and S. Sridharan. Fusion of output scores on language identification system. *Workshop on Multilingual Speech and Language Processing, Aalborg Denmark*, 2001.

[101] N. Brummer and D.A. van Leeuwen. On calibration of language recognition scores.

In *IEEE Odyssey'06: The Speaker and Language Recognition Workshop*, pages 1–8, 2006.

[102] D.A. van Leeuwen and N. Brummer. Channel-dependent GMM and multi-class logistic regression models for language recognition. In *IEEE Odyssey'06: The Speaker and Language Recognition Workshop*, pages 1–8, 2006.

[103] S. Pigeon, P. Druyts, and P. Verlinde. Applying logistic regression to the fusion of the NIST'99 1-speaker submissions. *Digital Signal Processing*, 10(1-3):237–248, 2000.

[104] S. Hongbin, L. Ming, L. Ping, and Y. Yonghong. Using SVM as back-end classifier for language identification. *EURASIP Journal on Audio, Speech, and Music Processing*, pages 1–6, 2008.

[105] The scribe manual" http://www.phon.ucl.ac.uk/resource/scribe/scribe-manual.htm, 1998.

[106] P. Matějka, P. Schwarz, J. Cernockỳ, and P. Chytil. Phonotactic language identification using high quality phoneme recognition. In *Ninth European Conference on Speech Communication and Technology, InterSpeech'05*, pages 2237–2240, 2005.

[107] Ogi - oregon graduate institute, http://www.cslu.ogi.edu/.

[108] P. Matejka, L. Burget, P. Sckwarz, and J. Cernocky. Brno university of technology system for NIST 2005 language recognition evaluation. In *IEEE Odyssey'06: theSpeaker and Language Recognition Workshop*, pages 1–7, 2006.

[109] A. Hanani, M.J. Carey, and M.J. Russell. Language recognition using multi-core processors. *Computer Speech & Language journal*, March 2012.

[110] Corp NVIDIA. http://www.nvidia.com/cuda/, June 2010.

[111] C. NVIDIA. NVIDIA Compute Unified Device Architecture: Programming Guide. *NVIDIA: Santa Clara, CA*, 2007.

[112] http://www.fftw.org/, June 2010.

[113] M. Frigo and S.G. Johnson. FFTW: An adaptive software architecture for the FFT. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98*, volume 3, pages 1381–1384, 1998.

[114] P. Duhamel and M. Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal Processing*, 19(4):259–299, 1990.

[115] C. NVIDIA. Cufft library, 2011.

[116] NVidia. CUDA toolkit 4.0 performance report, nvidia website, june 2011.

[117] Y. Dotsenko, S.S. Baghsorkhi, B. Lloyd, and N.K. Govindaraju. Auto-tuning of fast Fourier transform on graphics processors. In *Proceedings of the 16th ACM symposium on Principles and practice of parallel programming*, pages 257–266, 2011.

[118] R.M. Bolle, S. Pankanti, and N.K. Ratha. Evaluation techniques for biometrics-based authentication systems (FRR),. In *15th Int. Conf. on Pattern Recognition, ICPR'00*, pages 28–31. IEEE Computer Society, 2000.

[119] S. M. D'Arcy. *The effect of age and accent on automatic speech recognition performance*. PhD thesis, The University of Birmingham, 2007.

[120] A. Hanani, M. Russell, and M.J. Carey. Computer and human recognition of regional accents of British English. In *Twelveth Annual Conference of the International Speech Communication Association, InterSpeech'11, Florence, Italy*, 2011.

[121] A. Hanani, M.J. Russell, and M.J. Carey. Human and computer recognition of regional accents and ethnic groups from British English speech. *Computer Speech & Language journal*, January 2012.

[122] S.J. Young, G. Evermann, MJF Gales, D. Kershaw, G. Moore, JJ Odell, DG Ollason, D. Povey, V. Valtchev, and PC Woodland. The HTK book version 3.4. 2006.

[123] N. Minematsu. Mathematical evidence of the acoustic universal structure in speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'05*, pages 889–892, 2005.

[124] Xugang Lu and Jianwu Dang. An investigation of dependencies between frequency components and speaker characteristics for text-independent speaker identification. *Speech Communication*, 50(4):312 – 322, 2008.

[125] L. Besacier, J.F. Bonastre, and C. Fredouille. Localization and selection of speaker specific information with statistical modeling. *Speech Communications*, 31:106, 1999.

[126] A. Hanani, M. Russell, and M.J. Carey. Speech-based identification of social groups

in a single accent of British English by humans and computers. In *IEEE International conference on Acoustics, Speech, and Signal Processing, ICASSP'11*, pages 4876–4879, 2011.

Appendix A

System	Language ID				Accent ID				Ethnic	
	(NIST2003)		(NIST2005)		(ABI-1)				Group ID	
	(30s)		(30s)		(30s)		(SPA)		(VaB)	
	EER	Acc	EER	Acc	EER	Acc	EER	Acc	EER	Acc
GMM-UBM	1.33	95.9	13.1	74.5	17.32	57.38	14.83	60.2	15.1	85
GMM-SVM	1.2	96.5	11.75	75.3	16.2	60.12	12.7	66.3	19.8	83.7
GMM-SVM-GMM	0.5	98.4	9.14	81.6	14.73	64.4	10.7	70.4	19	84.4
MLM-uni-gram	1.34	96.6	11.9	76	16.9	57.2	15.4	59	17	83
UBM-bi-gram	3.18	91.8	18.2	66	22.3	50.13	21.2	53.5	8.2	90.6
Acoustic-fused	0.41	98.7	8.4	83.4	12.82	72.3	9.3	75.6	7.3	92.7
Phonotactics	1.48	95.8	5.2	92.7	9.18	74.1	6.5	82.1	13.0	87.2
Phono-Acou-Fused	0.34	99.1	4.4	93	7.2	85.4	5.2	86.4	4.0	94.3
ACCDIST-Corr.dist.	-	-	-	-	-	-	2.66	93.17	-	-
ACCDIST-SVM	-	-	-	-	-	-	1.87	95.18	-	-
Human	-	-	-	-	-	-	-	58.24	-	90.2

Table A.1: Summary of results for all systems and tasks. The figures are percentage EER and percentage recognition accuracy (Acc)

Algorithm 1 IIR Filter Bank kernel

```
Inputs:: SEG, N, nF, Order, sm, indata, a ,b
Outputs: outdata
$xIndex \leftarrow blockIdx.x \times blockDim.x + threadIdx.x$;
$yIndex \leftarrow blockIdx.y \times blockDim.y + threadIdx.y$;
$oIndex \leftarrow yIndex \times N + xIndex$;
—shared— float tempA[4][17], tempB[5][17] ;
float shifta[4], shiftb[5], Am[4], Bm[5], tmp = 0 ;
int dd1, dd2, j;
float tmp_sm1, prev_tmp_sm1 = 0, tmp_sm2, prev_tmp_sm2 = 0 ;
if$(xIndex < N$and$yIndex < nF)$ {
if$(threadIdx.x < 4)$
$tempA[threadIdx.x][threadIdx.y] \leftarrow a[yIndex + (int)threadIdx.x \times nF]$
if$(threadIdx.x < 9$ AND $threadIdx.x > 3)$
$tempB[threadIdx.x - 4][threadIdx.y] \leftarrow b[yIndex + (int)threadIdx.x \times nF]$
SyncThreads() ;
for$(j = 0; j < Order; j++)$ {
if$(j < Order - 1)${
$Am[j] \leftarrow tempA[j][yIndex]$
$shifta[j] \leftarrow 0$
}
$Bm[j] \leftarrow tempB[j][yIndex]$
$shiftb[j] \leftarrow 0$
}
$\textbf{for}(int j = 0; j < SEG; j++)$ {
$\textbf{for}(j = 0; j < Order - 1; j++)$
$shiftb[j] \leftarrow shiftb[j + 1]$
$shiftb[Order - 1] \leftarrow idata1[xIndex + j \times N]$
$tmp \leftarrow 0$
$\textbf{for}(j = 0; j < Order - 1; j++)$ {
$tmp \leftarrow tmp + Bm[j] \times shiftb[j] - Am[j] \times shifta[j]$
}
$tmp \leftarrow tmp + Bm[Order - 1] \times shiftb[Order - 1]$
tmp_sm1 $\leftarrow$ abs(tmp) - sm$\times$prev_tmp_sm1
$tmp\_sm2 \leftarrow tmp\_sm1 - sm \times prev\_tmp\_sm2$
$\textbf{for}(j = 0; j < Order - 2; j++)$
$shifta[j] \leftarrow shifta[j + 1]$
$shifta[Order - 2] \leftarrow tmp$
$prev\_tmp\_sm1 \leftarrow tmp\_sm1$
$prev\_tmp\_sm2 \leftarrow tmp\_sm2$
}
$odata[oIndex] \leftarrow tmp\_sm2$
}
```

Algorithm 2 FIR Filter Bank kernel

```
Inputs:: DATA_DIM, N , M, G_Ceof, indata
Outputs: outdata
$xIndex \leftarrow blockIdx.x \times blockDim.x + threadIdx.x;$
$yIndex \leftarrow blockIdx.y \times blockDim.y + threadIdx.y;$
$oIndex \leftarrow yIndex \times N + xIndex;$

—shared— float S_Coef[17][80]
float $acum = 0.0$

if$(xIndex < N \textbf{and} yIndex < M)$
{
if$(threadIdx.x < M$ AND $threadIdx.y < DATA\_DIM)$
$S\_Coef[threadIdx.x][threadIdx.y] \leftarrow G\_Coef[yIndex + threadIdx.y \times M]$

SyncThreads()

for$(int j = 0; j < DATA\_DIM; j++)$
$acum \leftarrow acum + indata[xIndex + j \times N] \times S\_Coef[threadIdx.x][j]$

$outdata[oIndex] \leftarrow acum$

}
```