Mehdi Soufifar

# Subspace Modeling of Discrete Features for Language Recognition

Thesis for the degree of Philosophiae Doctor

Trondheim, November 2014

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

**NTNU – Trondheim**
Norwegian University of
Science and Technology

**NTNU**
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

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



ISBN 978-82-326-0496-8 (printed ver.)
ISBN 978-82-326-0497-5 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2014:292

Printed by NTNU-trykk

# Abstract

This thesis addresses the language recognition problem with a special focus on phonotactic language recognition. A full description of different steps in a language recognition system is provided. We study state-of-the-art speech modeling techniques in language recognition that comprise phonotactic, acoustic and prosodic language modeling. A brief understanding of the state-of-the-art subspace modeling technique known as the iVector model for continuous features is given. Using recent proposals on training the iVector model for continuous features, we explain our recipe for extracting iVectors for acoustic and prosodic features that results in similar language recognition performance as the state-of-the-art results reported in the recent literature. In the next step, inspired by the intuition behind the iVector model for continuous features, we propose our iVector model for discrete features. After a general explanation of the model, adaption of the proposed model to the n-gram model that is used to extract iVectors representing the language phonotactics is given. Finally a regularized iVector extraction model for discrete features that is robust to model overfitting is proposed. The full theoretical derivation of the proposed iVector model for discrete features is given. We also explain use of discriminative and generative classifiers for training language models based on the different extracted iVectors. Effects of the iVector normalizations for binary and multi-class formulation of the used classifiers is also studied.

We report performances of our iVector model on NIST language recognition evaluation LRE2009, LRE2011 and RATS language recognition as the most recent and challenging language recognition task. Using our phonotactic iVector model, we obtain a significant improvement over our phonotactic baseline system which was a state-of-the-art system at the time of starting this thesis. Our results on NIST LRE09, NIST LRE2011 and RATS confirms superior advantage of our iVector model for discrete features compared to the other state-of-the-art phonotactic system.

# Acknowledgements

May individuals have influenced me during the compilation of this thesis. First, I would like to thank my supervisor Torbjørn Svendsen who gave me the opportunity to do this PhD and all his support during this thesis. Specially for supporting my collaboration with other speech groups and in particular the collaboration with Speech@FIT. Takk Torbjørn!

During the last five years, I had the pleasure of visiting other speech groups and share moments of hard work, deadline nights and cheer. My very special thanks to Honza Černocký and Lukáš Burget at Speech@FIT who kindly accepted me as a member of Speech@FIT and helped me to pursue my research on language and speaker recognition. Even though only Lukáš Burget is the co-supervisor on this thesis, I should thank them both equally, Lukáš for his splendid technical supervisions and all the midnight discussions, and Honza for organizing everything and keeping the group running on such a high level. Moje veliké děkuji patří oběma!

I would like to thanks all my colleagues at Speech@FIT particularly Pavel Matějka, Ondrej Glembek, Olda Plchot, Karel Veselý, Sandro Cumani, Marcel Kockmann, Martin Karafiát and Petr Schwarz for their support and for making my stay in Brno such a memorable chapter in my life. I was privileged to participate in BOSARIS, SRE and LRE workshops during my stay in Speech@FIT and enjoy my time with all the great participants, in particular, Nikko Brümmer and Najim Dehak. Thank you all!

Even though my visit from speech group at university of eastern Finland was fairly short, I learned a lot from working with great people there in particular Rahim Saeidi, Tomy Kinnunen and Pasi Fränti.

I feel grateful of my family to whom I owe everything in my life. Their patience and support have always delighted every single moment of my life. It was far from reality to accomplish this PhD without their inspiration.

Last but not least, I would like to thank all my Friends at NTNU and Trondheim who supported me during this thesis. Thank you Timo, Marco, Babak, Hessam, John, Line, Alfonso, Arild and Jarle.

# Contents

# List of Figures

# List of Tables

# Nomenclature

BLR Binary Logistic Regression

BNBS Broadcast Narrow Band Speech

BSVM Binary Support Vector Machines

BUT Brno University of Technology

CTS Conversational Telephony Speech

DCT Discrete Cosine Transform

EM Expectation Maximization

GD Gradient Descent

GLC Generative Linear Classifier

GLDS Generalized Linear Discriminant Sequence

GMM Gaussian Mixture Model

IPA International Phonetic Association

LDC Linguistic Data Consortium

LR Logistic Regression

LVCSR Large Vocabulary Continuous Speech Recognition

MFCC Mel-Frequency Cepstral Coefficients

ML Maximum Likelihood

MLR Multi-class Logistic Regression

MMI Maximum Mutual Information

MSVM Multi-class Support Vector Machine

NIST National Institute of Standards and Technology

PER Pair Error Rate

PPRLM Parallel Phoneme Recognizer Language Model

SID Speaker Identification

SMM Subspace Multinomial Model

SnGM Subspace n-gram Model

SVD Singular Value Decomposition

UBM Universal Background Model

VAD Voice Activity Detection

VOA Voice Of America

WCCN Within Class Covariance Normalization

# Chapter 1

# Introduction

Languages are the main means of communication nowadays. There are more than 7 billion people and more than 6912 known languages in the world [30]. Recent communication advances, and in particular Internet, has made multimedia information from all around the world available for everyone. This has removed geographical borders for communication and there is an increasing demand for communication among people with different languages. Nevertheless, there is a barrier for those who would like to use and process multimedia and in particular speech data; *language diversity.* Almost the whole automatic speech processing technology requires prior information about the embedded language in the data to have a reasonable performance. In this thesis, we address the problem of the automatic language recognition that can serve as a front-end to many other speech technologies (e.g automatic speech recognition, automatic translation, speech synthesis etc.). Having an accurate prior on the data language makes a huge improvement in the performance of these systems.

## 1.1 Automatic language recognition

The task of automatically determining the language of a spoken utterance is called Language recognition. Being able to recognize a language requires prior knowledge of the target languages. This leads us to a better definition of the task that we are going to address in this thesis as:

*Having some prior knowledge from a target list of languages, we are asked to automatically determine the language identity of a trial speech utterance.*

Similar to many other technologies, language recognition and more generally automatic speech processing technology is inspired by the human capability of understanding speech. In the early years of language recognition, a comparative study showed that even by providing the machine with reasonable amount of prior knowledge about a language, humans seem to outperform machines in this task [53]. A native speaker of a language can detect a person speaking the same language in a speaking crowd almost immediately. A multilingual person can easily recognize the language of a speech sample and in case the language is not among the languages of his expertise, he can make a reasonable guess. However, as the trials are taken from linguistically closer languages, or as the length of trials become shorter, recognizing the target languages becomes more challenging. A recent human benchmark for National Institute of Standards and Technology (NIST) proposed by [16] language recognition tasks (see Chapter 6) showed that in many adverse conditions, the current state-of-the-art language recognition systems outperform human beings in language recognition [83].

Let us have a closer look at the world's languages. Even though there are more than 7 billion people and 6912 known languages in the world, 5% of the world's languages are spoken by 94% of the world's population [30]. In fact, there are lots of similarities among some of the spoken languages and in many cases, it is not really easy to decide whether two spoken languages are dialects of the same language or they should be considered as separate languages. Languages are partly characterized by their phoneme sets. In linguistics a *phoneme* is a basic element of a given language or dialect, from which, words in that language or dialect are built. From a linguistic point of view, a language can be characterized by the following features:

I ) **Phonetics**: The articulatory system of the human being is capable of producing many phonemes. However, number of the phonemes in each language ranges from 15 to 50 with the peak at 30 [87]. Many of these phonemes are common in different languages. Some are phonetically the same but produced slightly different in different languages. Some phonemes are more frequent in some languages than others. In general, the phoneme set of a language and frequency of each phoneme in the corresponding language can lead us to the language identity.

II ) **Phonotactics**: Phonotactics is dealing with the combination of the phonemes. Two languages may share almost the same phoneme set. However, the combination of sounds may differ. E.g. in Slavic languages, it is possible to have multiple consonants in a row while that is not common in some other languages such as Farsi. As we will see, the

repetition pattern of phonemes is a very useful source of information to discriminate among languages.

III ) **Prosody**: Prosody is a perceptual music of speech. It is strongly correlated with fundamental frequency (F0) [77]. Prosody is generally suprasegmental, i.e. that it is not common to speak of the prosody of a sound. Rather, prosody usually refers to the syllabic, word or phrase level (or even longer segments). Features like loudness and stress are also affecting the prosody. English is an exemplar language for the prosodic features. The three widely spoken variations of English: American, British and Indian are the same language. However, the prosody is very different.

IV ) **Lexicon & Morphology**: In linguistic, a morpheme is roughly defined as the smallest linguistic unit that has semantic meaning and morphology is the identification, analysis and description of the structure of words. Two languages may have very similar sets of sounds. However, the lexica, containing the words of the two languages may be different, and the frequency of words may be different. For example, Farsi and Dari have very similar phoneme sets and lots of words in common. However, some words are rarely used in one compared to the other. Two languages might even have similar words with different morphologies as is the case with Farsi and Arabic.

V ) **Grammar**: This is probably the highest level of abstraction that may carry useful information to discriminate among languages. However, we are not aware of any reported language recognition system (LID) system that takes advantage of that.

Such a language characterization does not provide a quantitative measure on language similarity and just shows the broad level of information that can be exploited for the language recognition task. In the case of dialect recognition, more specific techniques might be more practical. Normally, we regard variations of a language as different dialects of the language if the difference is not only different accents but also different lexicons.

In this thesis, We are addressing the language recognition problem in general and we do not get into dialect recognition. Nevertheless, some of the techniques might be useful for the dialect recognition as well.

So far, we have talked about sources of information that can lead us to discriminate among languages from a linguistic point of view. Almost all of these information sources require a certain level of abstraction for speech modeling. The lowest level of speech representation is the signal itself.

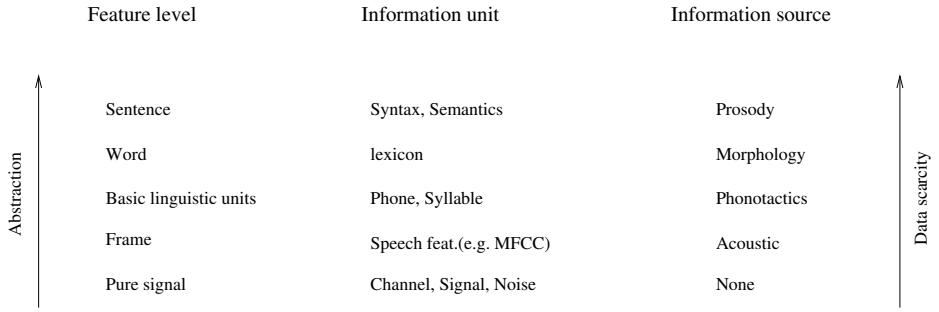



Figure 1.1: Language discrimination in different levels

Statistical modeling of the speech cepstrum has been shown to be a very effective way of speech modeling. Figure 1.1 depicts levels of abstraction and the corresponding sources of information in the speech signal. In fact, the more abstract we get, the more uncertain the information becomes. This is mainly because the information sources in each level depend on the output of the lower level and any error in the lower level would be propagated bottom-up. For example to have a relatively good stream of spoken phonemes in an utterance, a proper phoneme recognizer trained on the same kind of signal is necessary. Furthermore, the more abstract we get, the more expensive it is to have training data and we have a sparser source of information.

## 1.2 Automatic language recognition solutions

A general block diagram of a typical language recognition system is depicted in Figure 1.2. As any other statistical modeling based system, it is based on collecting statistics for the problem and modeling the space and finally deciding on a newly observed statistics based on the trained model. The whole process can be split into two main sub–systems: front-end and back-end. In the field of language recognition, we regard any effort with the purpose of providing an information to the statistical model as front-end and the statistical modeling and interpretation of the input data is regarded as back-end.

Solutions for the language recognition problem mainly vary in the front-end of the system and as long as the model assumption holds, similar back-ends can be used for different front-ends. More specifically, the differences are mainly in the steps up to *language modeling* in Figure 1.2. Generally, solu-

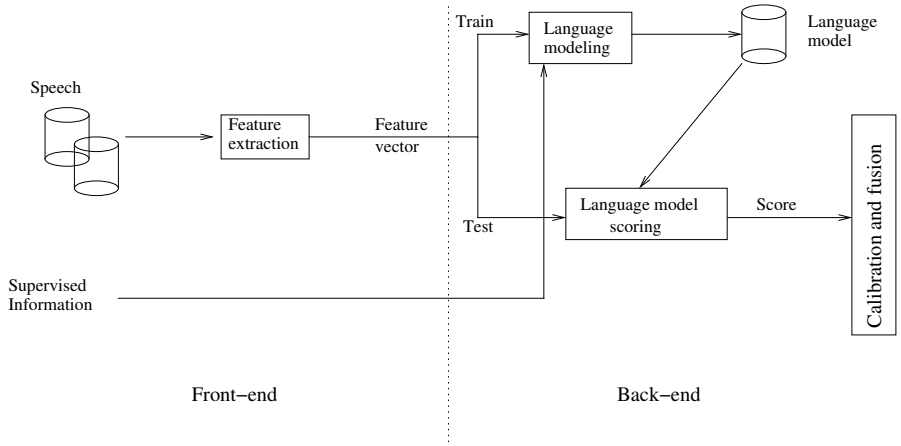



Figure 1.2: A general language recognition block diagram.

tions to the language recognition task are classified into two main categories: phonotactic and acoustic that mainly differ in front-end speech modeling. Let us first briefly talk about phonotactic and acoustic modeling of speech.

### 1.2.1 Phonotactic solutions

The main idea behind phonotactic language recognition is that languages can be discriminated according to their phonotactics. To get statistics about language phonotactics, we normally use a linguistic unit recognizer (i.e. phoneme in this work) to tokenize a speech signal to a stream of linguistic unit symbols based on which n-gram statistics that represent language phonotactics are extracted. Lots of effort has been put into training proper phoneme recognizers for the language recognition tasks in the system front-end. Even though it has been shown that high precision phoneme recognizers can effectively improve the language recognition system performances [49], it should be noted that phoneme recognition accuracy is not the only criterion for the effectiveness of the phoneme recognizer. Robustness against the channel variability and acoustic coverage of the phoneme recognizer should be always taken into consideration while training a phoneme recognizer for the language recognition tasks. To have a better acoustic coverage of languages, different configurations for multilingual phoneme recognizers have been studied [75],[42]. Use of other acoustic sub-word units without corresponding linguistic definition as an alternative to phoneme recognizer was also proposed in the literature [42]. The

phoneme recognizer (either monolingual or multilingual) is then used to extract n-gram statistics for all utterance from any language of interest. The generated n-gram statistics corresponding to language specific training utterances are then used to train statistical language models for the languages of the interest. The n-gram statistics can be used to train language models (e.g. n-gram language models) directly or they can be transformed to any other form that may be effective for modeling languages. This basically means that the feature extraction block in the system front-end can include subsystems.

A traditional approach in the language recognition is to use n-gram statistics and train generative n-gram language models as in *phoneme recognition followed by language modeling* (PRLM) [88]. In the PRLM, we run the phoneme recognizer on training data from each target language and then we train a separate n-gram language model for the corresponding language. This way, we will end up with $N$ language models where $N$ is number of the target languages in our language recognition task. During the test phase, we can calculate the likelihood of each test speech sample with respect to each of the target languages by running the phoneme recognizer on the test speech sample and calculating the likelihood of the produced phoneme sequence with respect to the corresponding language model. This is done in the *Language model scoring* block in Figure 1.2. In case of using monolingual phoneme recognizer, we are simulating a situation in the real word where a monolingual person is exposed to samples from list of target languages and he is asked to decide on the language identity of a test speech sample. There might be many phonemes in the target languages that are unfamiliar for him. However, he tries to project the acoustic space of the target languages to his own acoustic space and decide on the language identity of the test speech sample based on the language models he has already made for each of the target languages. A step up to this solution is to use multiple phoneme recognizers. This is called *parallel phoneme recognition followed by language modeling* (PPRLM) configuration and is inspired by the behavior of multilingual people [88][85]. This way, every utterance is decoded with multiple phoneme recognizers that gives us different n-gram statistics. For each phone recognizers, n-gram statistics corresponding to the language specific utterances are used to train a separate n-gram language model. More info on PRLM system configuration is given in Section 4.1.

In the current state-of-the-art phonotactic language recognition systems, the n-gram statistics are still the main tool for describing the language phonotactics in the front-end. However, recent proposals for the back-end shows that it is more reasonable to separate within-class and between-class

variability while training language models. In [16], the author showed that we can represent an utterance by putting all the n-gram statistics of a certain order in a fixed length vector that is considered as an multidimensional observation of the corresponding language. This way we can model the within-class variability as well as the between-class variability. In [16] SVM classifiers are used to discriminate between observations from different languages. As we will see later in this thesis, different classifiers (e.g. logistic regression or Gaussian classifiers) are other alternatives. Later on, other researchers proposed more effective features based on transformed n-gram statistics. In [42] a vector space modeling (VSM) formulation for the language recognition is proposed and [50] proposed a PCA-based transformation of the n-gram statistics. These methods use a combination of linear and nonlinear transformations to represent the vector of n-gram statistics with a low-dimensional representation of the corresponding speech utterance. These low-dimensional representations of each language are then used to train classifiers that are used as our language models. The phonotactic solutions and the corresponding speech modeling techniques are described in detail in Chapter 4.

### 1.2.2 Acoustic solutions

Acoustic solutions are using solely information contained in the signal spectrum without using any higher level information such as linguistic units. As a result, they have a much simpler front-end. Normally, conventional speech feature vectors, e.g. Mel-frequency cepstral coefficients (see Section2.1.1), are extracted from the signal. The main idea is to estimate the distribution of extracted feature vectors for each language of the interest. This is, mainly, accomplished through Gaussian mixture model (GMM) modeling of the feature vectors. Many other features for the front-end of the language recognition system has been proposed among which, shifted delta cepstral (SDC) [81] features are widely used for the language recognition problem. The extracted features in the front-end can be used to train language models. In other words, we will end up with multidimensional GMMs as our language models. Generalized linear discriminant sequence (GLDS) proposed by [16] and discriminative training of language specific GMMs based on maximum mutual information (MMI) criterion proposed by [15] were successful proposals in this category.

More recently, researchers tried to use utterance-specific GMMs instead of language-specific GMMs. Success of subspace modeling and eigenchannel compensation which was originally proposed for speaker identification (SID) tasks, has revolutionized acoustic language recognition [35][36] [10].

The most recent front-end subspace modeling technique known as iVector, which is a feature extraction model in the front-end of the language recognition system, has become the state-of-the-art technique in SID [21] and was successfully adapted for the language recognition [46]. The main idea of the iVector model in acoustic language recognition is to represent each utterance dependent GMM with a low-dimensional latent variable (see Section 2.4) and use the low-dimensional representation of the utterance as a feature vector to the following language classifier. The acoustics solutions and the corresponding speech modeling techniques are described in detail in Chapter 2.

### 1.2.3 Language recognition back-end

So far we briefly spoke about main approaches for modeling the speech using phonotactic or acoustic features. Note that, as Figure 1.2 shows, the front-end processing is the same for all training and test speech utterances. Once we have the representation of a speech utterance in terms of feature vectors, we can train language models on top of that. The language model can be a traditional generative n-gram language model, a Gaussian linear classifier or discriminative classifiers. In this thesis we use SVM and logistic regression as different flavors of discriminative classifiers and a Gaussian linear classifier as generative classifier. These classifiers are used to generate language and utterance specific scores. These scores can be probabilistic (e.g log likelihoods or log likelihood ratios) or other different scores that do not have a probabilistic interpretation (i.e output of SVM). Training of the language models is discussed in Chapter 5. The language scores have to be converted to comparable class conditional likelihoods by means of a multi-class classifier (e.g. Gaussian linear classifier or multi-class logistic regression). We also should calibrate the class likelihoods to get reasonable performance on the test data. The calibration is normally done on a development set that has a closer distribution to the test data. In the next step, we may also have multiple language recognition front-ends and we would like to used them to define the language label of an utterance. This is referred to as *fusion* in the language recognition. The calibration and fusion are normally done by means of training another layer of a multi-class classifier on development data and can be done jointly or separately. This is shown as *calibration and fusion* in Figure 1.2. The output of the *calibration and fusion* is a vector of class conditional likelihoods corresponding to the target language set. In other words, each dimension in the output vector of the *calibration and fusion* is $P(\mathbf{x}|l_k)$ where $\mathbf{x}$ stands for a test utterance and $l_k$ is the $k^{th}$ language in the target language set.

Once we have the calibrated and fused class likelihoods, we can make a decision on the language label of a speech utterance. When making the decision, we can consider a particular application (or operating point) defined in terms of prior probabilities over languages. The language priors are used to produce language posterior probabilities using Bayes rule:

$$p(l_n|\mathbf{x}) = \frac{p(\mathbf{x}|l_n)p(l_n)}{\sum_k p(\mathbf{x}|l_k)p(l_k)}. \tag{1.1}$$

Labels are typically assigned to utterances based on maximum a-posteriori (MAP) rule (i.e. language with highest posterior probability is chosen). This procedure is discussed in detail in Chapter 5.

The performances of different language recognition systems should be evaluated on standard widely used evaluation sets. NIST arranges language recognition evaluations (LRE) that are normally held every other year. The history and definition of the language recognition task in NIST LREs are given in Chapter 6. The last two NIST language recognition evaluations (i.e. NIST LRE11 and NIST LRE2009) are used in this thesis.

## 1.3 Claims and contributions of this thesis

The state-of-the-art solutions in both of the language recognition approaches are focusing on similar problem: distinguishing between sources of the within-class variability that are imposed by language variability with those that are imposed by other factors (e.g. transmission channel, speaker, etc.). Benefiting from SID, acoustic language recognition based on iVector model for continuous features has a more mature mathematical background and also better performance compared to the phonotactic state-of-the-art systems. In this thesis, we show how we can represent an utterance-specific n-gram language model with a low-dimensional vector that we refer to as phonotactic iVector based on iVector model for discrete features. We propose a framework for extraction of iVectors from vector of the n-gram statistics.

The main objective of this thesis is to provide a framework for phonotactic iVector extraction and show that we can obtain better language recognition performance by extracting phonotactic iVectors from the output of the commonly used phoneme recognizers. In the next step, we show that, not only do the phonotactic iVectors outperform the state-of-the-art phonotactic language recognition as a stand-alone system, it provides more complementary information, compared to previous state-of-the-art system, to the language recognition system fusion. To achieve this goal, we need to develop other state-of-the-art language recognition systems (e.g. acoustic

iVectors) to show how the complementary systems can improve the overall performance of a language recognition system.

The main contributions of this thesis are as follows:

I ) **Acoustic iVector extraction**: we present our understanding of the iVector model for continuous features based on which, we build the acoustic language recognition system that is comparable with the reported state-of-the-art acoustic iVector language recognition systems [46].

II ) **Prosodic iVectors**: prosodic language recognition yields much lower performance than phonotactic and acoustic language recognition systems. However, since it is exploiting different sources of information, it provides complementary information to language recognition system fusion [69]. We develop a prosodic language recognition system based on the iVector model for continuous features by adapting the prosodic feature extraction that was proposed for SID [39]. We explain our prosodic feature extraction and modeling and we show that prosodic language recognition system can improve the language recognition performance along with other more efficient acoustic and phonotactic language recognition systems.

III ) **Phonotactics iVector**: we propose our iVector model for discrete features and explain how to extract phonotactic iVectors. We show that the phonotactic language recognition based on the proposed phonotactic iVectors performs better than current state-of-the-art phonotactic language recognition systems. In the next step, we propose an enhanced regularized phonotactic iVector extraction model that is theoretically consistent with the n-gram model assumption, robust to model overfitting and significantly outperforms the current state-of-the-art phonotactic language recognition systems.

IV ) **Comparative study**: we compare performances of the proposed phonotactic, acoustic and prosodic language recognition systems. Different fusion schemes are studied and reported. A descriptive comparison on phonotactic and acoustic language recognition systems for the most recent NIST LRE (i.e. LRE11, see Chapter 6) is given and reasonable intuitions for inspiring researchers to work on both categories of solutions are given. Furthermore, we study the performance of our phonotactic iVectors along with acoustic iVectors over the most recent and challenging language recognition task called RATS. This gives us

a valuable system performance comparison under noisy channel conditions.

### 1.3.1 Publications within the scope of this thesis

List of my publication related to the scope of this thesis are as follows:

- **Mehdi Soufifar**, Marcel Kockmann, Lukáš Burget, Oldřich Plchot and Torbjørn Svendsen. *iVector Approach to Phonotactic Language recognition.* In Proceeding of Interspeech 2011 Florence, Italy.

  This paper includes the subspace multinomial model (SMM) for discrete features (see Section 4.5). It also studies performances of binary SVM and logistics regression for the language recognition.
- **Mehdi Soufifar**, Sandro Cumani, Lukáš Burget and Jan Černocký. *Discriminative Classifiers for Phonotactic Language Recognition with iVectors.* In proceeding of International Conference on Acoustic, Speech and Signal Processing (ICASSP) 2012, Kyoto, Japan.

  This paper studies performances of binary and multi-class formulations of SVM and logistic regression used for training language models. Effect of different normalization techniques on the performance of the language recognition is also studied (see Section 7.5).
- **Mehdi Soufifar**, Lukáš Burget, Oldřich Plchot, Sandro Cumani and Jan Černocký. *Regularized Subspace n-gram Model for Phonotactic iVector Extraction.* In proceeding of Interspeech 2013 Lyon, France.

  This paper proposed our latest iVector model for discrete features (see Section 4.5.6 and Section 4.5.7). An extension to the previous iVector model for discrete features is given. To avoid model overfitting, a regularized iVector model parameter estimation is also proposed.
- Niko Brümmer and Sandro Cumani and Ondřej Glembek and Martin Karafiát and Pavel Matějka and Jan Pešán and Oldřich Plchot and **Mehdi Soufifar** and Edward Villiers de and Jan Černocký. *Description and analysis of the Brno276 system for LRE2011.* In proceedings of Odyssey: The Speaker and Language Recognition Workshop 2012 Singapor, Singapor.

  This paper provides description of the systems that were submitted to NIST language recognition evaluation 2011 under the name

Brno276. Part of the phonotactic systems were developed by the author of this thesis and are reported in Section 7.12.

- Pavel Matějka and Oldřich Plchot and **Mehdi Soufifar** and Ondřej Glembek and Fernando Luis D'Haro and Karel Veselý and František Grézl and Jeff Ma and Spyros Matsoukas and Najim Dehak. *Patrol Team Language Identification System for DARPA RATS P1 Evaluation.* In proceedings of Interspeech 2012, Portland, USA.

  This paper provides description of the systems that were submitted to RATS language recognition evaluation under the name Patrol. The phonotactic systems were developed by the author of this thesis and are reported in Section 7.13.

- Fernando Luis D'Haro and Ondřej Glembek and Oldřich Plchot and Pavel Matějka and **Mehdi Soufifar** and Ricardo Cordoba and Jan Černocký. *Phonotactic Language Recognition using i-vectors and Phoneme Posteriogram Counts.* In proceedings of Interspeech 2012, Portland, USA

  This paper shows that the iVector model based on discrete features can be used for other discrete features than n-gram statistics. This paper uses phonotactic features called Posteriogram as an input to the iVector model for discrete features.

## 1.4 Clarification on the terminologies

The language recognition problem can be treated as either a verification task or an identification task. Verification is about accepting or refusing that a trial utterance belongs to a certain language while identification is about defining a language label for a trial utterance. We will explain the difference in Chapter 6. Nevertheless, we shall mention in the beginning that we address the language recognition as defined by NIST. Since it is common to use the term *LID* as a reference to the NIST language recognition task, we use the same terminology. All the *LID* terms in this thesis are referring to language recognition as defined in NIST language recognition task.

The iVectors based on iVector model for continuous cepstral features is commonly referred to as acoustic iVector in the LID community. Other continuous features (e.g prosodic features as explained in Chapter 3) can also be modeled with iVector model for continuous features. In this thesis acoustic iVectors and prosodic iVectors are referring to the iVectors that are extracted using iVector model for continuous features using cepstral

(e.g. SDC) and prosodic features, respectively.

The main contribution of this thesis is the proposal of iVector model for discrete features. We use phonotactic n-gram statistics as an input to this model and therefore we refer to the iVectors extracted using the iVector model for discrete features shortly as *phonotactic iVectors*.

## 1.5 Structure of this thesis

The rest of this thesis is organized as follows:

- **Chapter 2** explains language modeling with acoustic features. The subspace modeling of GMM means is described and steps towards making state-of-the-art iVector extraction for continuous features are explained.

- **Chapter 3** explains language modeling with prosodic features. Different methods for segmentation of speech in syllable-like segments are explained. Approximation of prosodic features (pitch and energy) contours in each segment with Legendre polynomials is explained and steps toward building fixed-length feature vectors suitable for iVector model for continuous features are explained.

- **Chapter 4** we describe the current state-of-the-art phonotactic LID systems. Next, we propose our subspace multinomial model to model n-gram probabilities and our iVector extraction scheme for discrete features. We also explain how it differs from the iVector model for continuous features. The enhanced phonotactic iVector model, called regularized subspace n-gram model is the state-of-the-art solution for the LID problem that is proposed in this thesis.

- **Chapter 5** different techniques for training language models are explained. The necessity of a calibration and fusion module in LID system is discussed and different calibration and fusion plans used in this thesis are explained

- **Chapter 6** we briefly describe NIST language recognition evaluation history and evaluation metrics used by NIST language recognition evaluations. Next, different data sets used for the experiments in this thesis for each NIST language recognition evaluation is described. Sources of the collected data, distribution of the data in different train, development and evaluation sets are given.

- **Chapter 7** all the experiments regarding the development and tuning of each system are explained. The comparison of the baseline with the proposed techniques, performance of each of the explained systems and different fusion plans of the explained systems are given. Next, a comparison of the phonotactic and acoustic iVectors on the NIST LRE11 task is given. In the end, performance of the acoustic and phonotactic iVectors over RATS language recognition is compared.

- **Chapter 8** we conclude this thesis with a summary of our approaches and findings, followed by an outline of the potential extension of this work.

# Chapter 2

# Acoustic speech modeling

For many speech applications, the speech waveform is not directly usable. In the case of SID and LID, we normally segment the speech signal into overlapping frames and extract a feature vector for each frame. There have been many different feature extraction schemes proposed by researchers for different applications among which, MFCC and SDC [81] are widely used in SID and LID. The conventional approach to modeling speech for acoustic LID using cepstral features (e.g. MFCC, SDC and etc) involves training of a GMM (see Section 2.2) for each language of interest. A successful alternative to this approach was to train a GMM model referred to as universal background model (UBM) by pooling data from all languages and then adapt the UBM to a language of the interest using language specific data [62]. This adaptation was done using maximum a posteriori (MAP) adaptation and was originally proposed for SID. The MAP adaptation can give us a fairly robust language model estimation. Later, discriminative training between language specific GMMs that focuses on the boundaries of language distributions rather than estimating language distributions, was proposed in [15]. After that, works on channel compensation methods showed notable effect on SID system performances [13].

A breakthrough in LID modeling was the proposal of the subspace modeling approaches which are based on the assumption that each utterance (even those corresponding to a same language) has potentially different (channel specific) distribution of features. It is further assumed that such distribution can be modeled by an utterance specific GMM. The parameters of such GMM are, however, constrained to live in a low-dimensional subspace and are represented by low-dimensional latent vector. A successful subspace modeling approach that was called joint factor analysis (JFA) model and was originally proposed for SID [37], was successfully transformed

to LID [18]. In JFA, language specific GMM parameters were allowed to move in the subspace corresponding to channel variability. This was used to adapt language models to the channel of each test utterance. Recently, a successful variation of subspace modeling that serves as a feature extraction model, known as iVectors, was proposed in [21] and soon became a permanent part of all state-of-the-art SID and LID systems. The iVector is a low-dimensional representation of an utterance that contains information about all important sources of data variability. More specifically, the iVector is the value of the latent vector corresponding to the GMM adapted to a given utterance.

We shall mention that subspace modeling of model parameters has been used for many applications in speech technology. In [41] it was proposed for speaker adaptation in large vocabulary continuous speech recognition (LVCSR), [25] proposed it for cluster adaptive training of hidden Markov models and [60] proposed it for subspace modeling of the GMMs in LVCSR.

In the rest of this chapter, we first explain acoustic features extraction and then training of a GMM on top of the acoustic features. To achieve the objective of this thesis, we developed an acoustic LID system according to the recipe given in [29] using iVector model. This way, we showed that, one can obtain state-of-the-art acoustic LID using a published recipe and furthermore, we use the acoustic iVector system for the purpose of comparison and fusion with the other systems. In Section 2.4, we briefly explain the iVector model as used in this thesis. Extensive study of the recent subspace modeling techniques are available in [29] and [26].

## 2.1 Feature extraction

The MFCC[19] features are widely used as acoustic features in LID systems. However, authors in [81] showed that it is useful to use broader temporal information for LID and proposed use of shifted delta cepstra (SDC) for LID. Let us first briefly explain the extraction of the MFCC features and then extraction of SDC based on MFCC features.

### 2.1.1 Mel-frequencies cepstral coefficients (MFCC)

To get the Mel filter bank energies, the speech signal is first filtered by pre-emphasis linear filter to amplify higher frequencies and is then divided into overlapping frames with a typical length of 25ms and 10ms shift between each frame. A Hamming window is applied in the next step and the Fourier power spectrum is calculated for each frame. Finally a Mel filter bank is

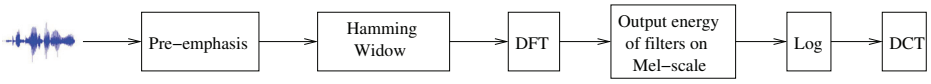



Figure 2.1: Steps toward getting Mel filter bank cepstral coefficients.

applied to smooth the spectrum. To mimic the human's nonlinear frequency resolution of loudness, a logarithm is applied to the output of the Mel filter bank energies and finally the discrete cosine transform is applied on the logarithm of the Mel filter bank energies to decorrelate the features and to reduce the dimensionality. Typical dimensionality for the final feature vector is 13. Steps toward getting MFCC features are shown in Figure 2.1.

The output of the Mel filter banks are used as input features for training the phoneme recognizers as is explained in Section 4.2.

### 2.1.2 Shifted delta cepstra (SDC) features

1st and 2nd order temporal derivatives of the MFCCs (delta- and delta-delta-cepstra) have typically been used to provide information about temporal evolution. In LID, the SDC features [81] comprise delta cepstra computed across multiple frames. The SDC is specified by four parameters: $N$, $d$, $P$ and $k$ where $N$ is the number of cepstral coefficients computed at each frame, $d$ defines the time advance for delta computation, $k$ stands for number of the blocks whose data coefficients are stacked to form the final feature vector and $P$ is the time shift between consecutive blocks. Calculation of SDC features in 7-1-3-7 configuration and for one of the MFCC coefficients is shown in Figure 2.2. Note that in 7-1-3-7 configuration, each SDC feature vector comprises 49 coefficients calculated fro MFCC $C0$ to $C6$ according to Figure 2.2. In this thesis, experiments on acoustic LID use SDC features.

## 2.2 Gaussian mixture model (GMM)

The Gaussian mixture model (GMM) is a weighted sum of C component Gaussian densities and the probability of a D-dimensional observation vector is calculated as:

$$p(\mathbf{o}_i|\text{GMM}) = \sum_{c=1}^{C} \eta_c \boldsymbol{\mathcal{N}}(\mathbf{o}_i; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)), \tag{2.1}$$

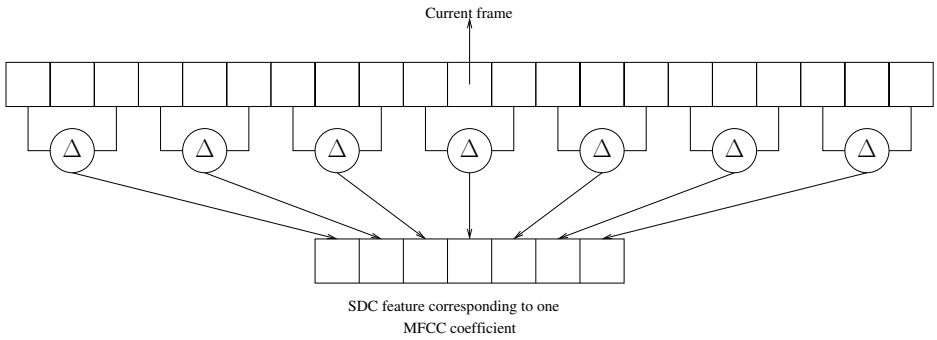



Figure 2.2: Extraction of SDC features in 7-1-3-7 configuration for one MFCC coefficient.

where $\boldsymbol{\mu}_c$, $c = 1 \ldots C$ denotes a D-dimensional Gaussian mean for component $c$. $\boldsymbol{\Sigma}_c$ and $\eta_c$ are covariance matrix and weight for the corresponding Gaussian component $c$, respectively. Assuming that the observations in an utterance are statistically independent, the likelihood of all the observations in an utterance w.r.t. a GMM model is:

$$p(\mathbf{O}|GMM(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\eta})) = \prod_{i=1}^{N} \sum_{c=1}^{C} \eta_c p(\mathbf{o}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c). \tag{2.2}$$

In order to have a fine resolution in acoustic space modeling, we need to train a fairly big GMM (normally 1024 or 2048 Gaussian components)

For the GMM training, we assume that lots of training utterances that represent the whole acoustic space are available. To keep the notations simple, we pool all the training data together and consider it as $N$ training frames denoted as $\mathbf{O}$. All of $C$ GMM components in the GMM are multivariate $D$ dimensional Gaussians, where $D$ is the dimensionality of the feature vector extracted from the speech signal. The parameters of the GMM comprise weights $\eta_c$, means $\boldsymbol{\mu}_c$ and covariance matrices $\boldsymbol{\Sigma}_c$. The parameters are trained in a maximum likelihood (ML) manner using the iterative expectation maximization (EM) algorithm [22]. To avoid an over-fitting of the Gaussian components to the training data, we used covariance flooring [86]. This means that if a GMM variance gets too small, we force it to stay above a predefined variance floor. The covariance matrix can be full or diagonal. In this thesis, diagonal covariance matrices are used for all experiments unless stated otherwise. Each EM iteration has two steps:

I ) *E* step: Given each frame, we calculate the posterior probability of

each Gaussian component $\gamma_c$.

II ) *M* step: Re-estimation of the current model parameters based on the soft assignment of frames to the Gaussian components of the GMM given by $\gamma_c$ from *E* step.

The *E* step involves calculation of the posterior probability of each Gaussian component for a given observation. The posterior probability of a Gaussian component $c$ for observation $\mathbf{o}_i$ is:

$$\gamma_c(i) = \frac{\eta_c \mathcal{N}(\mathbf{o}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}{\Sigma_{j=1}^{C} \eta_c \mathcal{N}(\boldsymbol{o}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{2.3}$$

Posterior probabilities are assigning Gaussian component to each frame and this is called *alignment of the data to the Gaussian components*. To update the model parameters in the *M* step, we have to first accumulate sufficient statistics over all training frames. The sufficient statistics can be used to estimate any parameters of the GMM model [8]. We can calculate the sufficient statistics as:

$$\gamma_c = \sum_i \gamma_c(i) \tag{2.4}$$

$$\boldsymbol{\theta}_c = \sum_i \gamma_c(i)\mathbf{o}_i \tag{2.5}$$

$$\boldsymbol{\Theta}_c = \sum_i \gamma_c(i)\mathbf{o}_i\mathbf{o}_i^T \tag{2.6}$$

The $\gamma_c$ is called zero-order statistics and can be seen as the occupation count of the Gaussian component $c$ over all the data. The $\boldsymbol{\theta}_c$ is called first-order statistics and $\boldsymbol{\Theta}_c$ is called the second–order statistics. Having the accumulated sufficient statistics, the mean of each Gaussian component can be updated as:

$$\boldsymbol{\mu}_c^{new} = \frac{1}{\gamma_c}\boldsymbol{\theta}_c. \tag{2.7}$$

The covariance matrix can be updated as:

$$\boldsymbol{\Sigma}_c^{new} = \frac{1}{\gamma_c}\boldsymbol{\Theta}_c - \boldsymbol{\mu}_c^{new}\boldsymbol{\mu}_c^{new\,T}. \tag{2.8}$$

The mixture component weights can be updated as:

$$\eta_c = \frac{\gamma_c}{N}. \tag{2.9}$$

**Algorithm 1:** maximum likelihood training of UBM-GMM

**Data**: Pool data from the UBM training set
**Output**: UBM-GMMs
$C \leftarrow 1$ ;
$\boldsymbol{\mu}_c \leftarrow$ mean of the training set ;
$\boldsymbol{\Sigma}_c \leftarrow$ covariance of the training set;
**while** *not enough mixture components* **do**
    split each mixture component in two ;
    find the largest eigenvalue of the covariance matrix $\boldsymbol{\Sigma}_c$ and its eigenvector $\mathbf{r}$ $\boldsymbol{\mu}_c^{p1} \leftarrow \boldsymbol{\mu}_c^p + \sqrt{2}\mathbf{r}$ ;
    $\boldsymbol{\mu}_c^{p2} \leftarrow \boldsymbol{\mu}_c^p - \sqrt{2}\mathbf{r}$ ;
    **while** *log-likelihood changes significantly* **do**
        do $E$ step ;
        do $M$ step ;

The iterative training of the GMMs starts with a single multivariate components that represents the whole training data. Until reaching a predefined number of Gaussian components, we keep splitting every Gaussian component in two and run the iterative EM algorithm. In each iteration, the likelihood of the data with respect to the GMM is calculated and the iteration continues until the change in the likelihood value between two consecutive iterations becomes negligible. At this point we split each Gaussian component and start the iteration all over again. To initialize the Gaussian components after the split, component weights are set to half of the old Gaussian component weight, means are moved in the direction of the largest variability with a small step of $\pm\sqrt{2}\mathbf{r}$ and we keep the covariance as before. The $\mathbf{r}$ is the eigenvector corresponding to the largest eigenvalue of the covariance matrix. The algorithm we used to train the UBM is shown in Algorithm 1

### 2.2.1 The likelihood function

We need to calculate the likelihood of data with respect to the Gaussian components in many places in this thesis ( e.g in (2.4)). Since we describe the data in terms of sufficient statistics, we need to express the likelihood function in terms of sufficient statistics. We start with a simple case where we know the model parameters, $\boldsymbol{\Omega}$. The likelihood of the whole observation, $\mathbf{O}$, with $N$ frames for a component $c$ of the GMMs:

$$p(\mathbf{O}|\mathbf{\Omega}_c) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{o}_i|\boldsymbol{\mu}^c, \boldsymbol{\Sigma}_c) \tag{2.10}$$

$$\boldsymbol{\mathcal{N}}(\mathbf{o}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_c|}} \exp\left(-\frac{1}{2}(\mathbf{o}_i - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\mathbf{o}_i - \boldsymbol{\mu}_c)\right), \tag{2.11}$$

where $D$ is the dimensionality of the feature vector. Normally, we prefer the logarithm of the likelihood function mainly because it is easier to work with. The log-likelihood of a GMM with $C$ components is

$$\log p(\mathbf{O}|\mathbf{\Omega}) = \sum_{i=1}^{N} \log \sum_{c=1}^{C} \eta_c \, \boldsymbol{\mathcal{N}}(\mathbf{o}_i|\boldsymbol{\mu}^c, \boldsymbol{\Sigma}_c^{-1}). \tag{2.12}$$

The sum inside the logarithm term in (2.12) is difficult to deal with and as a consequence there is no closed form solution for the maximization of the likelihood function in (2.12). To get the summation outside of the logarithm, we can expand (2.12) as:

$$\begin{aligned}
\log p(\mathbf{O}|\mathbf{\Omega}) &= \sum_i \log p(\mathbf{o}_i) \\
&= \sum_i \underbrace{\sum_c p(\mathbf{c}|\mathbf{o}_i)}_{=1} log\left(\frac{p(\mathbf{o}_i|\mathbf{c})p(\mathbf{c})}{p(\mathbf{c}|\mathbf{o}_i)}\right) \\
&= \sum_i \sum_c \underbrace{p(\mathbf{c}|\mathbf{o}_i)}_{\gamma_i^c} \log \underbrace{p(\mathbf{o}_i|\mathbf{c})}_{\mathcal{N}(\mathbf{o}_i;\boldsymbol{\mu}^c,\boldsymbol{\Sigma}_c)} - \sum_i \sum_c p(\mathbf{c}|\mathbf{o}_i) \log \frac{p(\mathbf{c}|\mathbf{o}_i)}{\underbrace{p(\mathbf{c})}_{\eta^c}} \\
&= \sum_i \sum_c \gamma_i^c \log \mathcal{N}(\mathbf{o}_i; \boldsymbol{\mu}^c, \boldsymbol{\Sigma}_c) - \sum_n \sum_c \gamma^c \log \frac{\gamma_i^c}{\eta^c}.
\end{aligned} \tag{2.13}$$

Expanding Equation 2.13 and using diagonal covariance matrices, we can efficiently express the likelihood in terms of sufficient statistics as:

$$\begin{aligned}\log p(\mathbf{O}|\mathbf{\Omega}) &= \sum_{i=1}^{N}\sum_{c=1}^{C}\gamma_i^c \log \mathcal{N}(\mathbf{o}_i;\boldsymbol{\mu}^c,\mathbf{\Sigma}_c) - \sum_{i=1}^{N}\sum_{c=1}^{C}\gamma_i^c \log\frac{\gamma_i^c}{\eta^c} \\ &= \sum_{i=1}^{N}\sum_{c=1}^{C}\gamma_i^c \log\frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}_c|^{1/2}} - \sum_{i=1}^{N}\sum_{c=1}^{C}\gamma_i^c \log\frac{\gamma_i^c}{\eta^c} \\ &- \sum_{i=1}^{N}\sum_{c=1}^{C}\gamma_i^c\frac{1}{2}[(\mathbf{o}_i-\boldsymbol{\mu}^c)^T\mathbf{\Sigma}_c^{-1}(\mathbf{o}_i-\boldsymbol{\mu}^c)] \\ &= \mathcal{D} + \sum_{c=1}^{C}[-\frac{1}{2}tr(\mathbf{\Sigma}_c^{-1}\mathbf{\Theta}^c) + \boldsymbol{\mu}^{cT}\mathbf{\Sigma}_c^{-1}\boldsymbol{\theta}^c - \frac{1}{2}\boldsymbol{\mu}^{cT}\gamma^c\mathbf{\Sigma}_c^{-1}\boldsymbol{\mu}^c],\end{aligned} \tag{2.14}$$

where $\mathcal{D}$ is a constant which does not depend on $\boldsymbol{\mu}^c$

$$\mathcal{D} = \sum_{n=1}^{N}\sum_{c=1}^{C}\gamma_n^c \log\frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}_c|^{1/2}} - \sum_{n=1}^{N}\sum_{c=1}^{C}\gamma_n^c \log\frac{\gamma_n^c}{\eta^c}.$$

## 2.3 GMM as universal background model

So far we have explained how to train a GMM model. In the traditional LID, cesptral acoustic observations of each language are assumed to be drawn from a language-specific GMM as:

$$\mathbf{o}_i \sim \text{GMM}(\boldsymbol{\mu}^L, \mathbf{\Sigma}^L, \boldsymbol{\eta}^L), \tag{2.15}$$

where $\mathbf{o}_i$ stands for an observation from language $L$, $\boldsymbol{\mu}^L$ are language specific means, $\mathbf{\Sigma}^L$ are language specific covariance matrices and $\boldsymbol{\eta}^L$ are language specific Gaussian component weights. Even though such a language specific GMM can be estimated using Maximum likelihood (ML) estimation, it yields a poor language model since there is no prior knowledge of the variability source in the observations. A successful alternative to this approach was to train a universal background model (UBM) by pooling data from all languages and then adapt the UBM to a target language using language specific data [62]. This adaptation was done using maximum a posteriori (MAP) adaptation and was originally proposed for SID. The MAP adaptation can give us a fairly robust language model estimation.

In the subspace modeling of the GMM mean parameters, the UBM-GMM is used to generate Baum-Welch sufficient statistics (see (2.4) and (2.5) ) that describe utterances in the acoustic space.

## 2.4 iVector Model

In the traditional GMM–based LID system, the assumption is that observations ($\mathbf{o}_{in}$) are drawn from language specific GMMs

$$\mathbf{o}_{in} \sim \text{GMM}(\boldsymbol{\mu}_L, \boldsymbol{\Sigma}_L, \boldsymbol{\eta}_L), \tag{2.16}$$

where $\boldsymbol{\mu}_L$ denotes a vector of Gaussian component means for language $L$, $\boldsymbol{\Sigma}_L$ defines the corresponding covariance matrices for the language $L$ and $\boldsymbol{\eta}_L$ is a vector holding the weights of Gaussian components in the GMM model. In other words, we assume that the parameters of the language model live in the original space of GMM parameters and we model within-class and between-class variability in the space of GMM parameters. However, in the subspace model and in particular, the iVector model, the key assumption is that observations in each utterance ($\mathbf{o}_{in}$) are drawn from a utterance specific GMM

$$\mathbf{o}_{in} \sim \text{GMM}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n, \boldsymbol{\eta}_n). \tag{2.17}$$

We also assume that all the utterance specific GMMs have corresponding Gaussian components that share the same covariance matrices and Gaussian component weights, $\boldsymbol{\eta}$, that are the same as those in the UBM. By this parameter sharing, the problem reduces to estimation of the utterance specific GMM means, $\boldsymbol{\mu}_n$. By stacking vectors of Gaussian component means to form a single vector, we form a supervector of GMM means ($\boldsymbol{\mu}_n$). Next, we assume that the utterance specific mean supervector lives in a subspace that has much lower dimensionality compared to the dimensionality of the mean supervector and can be expressed as:

$$\boldsymbol{\mu}_n = \mathbf{m} + \mathbf{T}\boldsymbol{\phi}_n, \tag{2.18}$$

where $\boldsymbol{\phi}_n$ defines the coordinates the utterance specific model in the low-dimensional subspace. This low-dimensional subspace is represented by a low rank matrix $\mathbf{T}$. The $\mathbf{m}$ defines the origin of the GMM mean space and is set to the GMM-UBM mean. We assume that $\boldsymbol{\phi}_n$ is a latent variable with standard normal prior

$$p(\boldsymbol{\phi}_n) = \mathcal{N}(\boldsymbol{\phi}_n; \mathbf{0}, \mathbf{I}). \tag{2.19}$$

This way, the prior distribution of $\boldsymbol{\mu}_n$ is given as:

$$p(\boldsymbol{\mu}_n) = \mathcal{N}(\boldsymbol{\mu}_n; \mathbf{m}, \mathbf{T}\mathbf{T}^T). \tag{2.20}$$

Each utterance is associated with a low-dimensional latent variable, $\boldsymbol{\phi}_n$. We can use the EM algorithm to estimate the model parameter $\mathbf{T}$ and the posterior distribution of the utterance–dependent latent variables $\boldsymbol{\phi}_n$. The model parameter $\mathbf{T}$ is also referred to as the subspace matrix in this thesis since it defines the expansion from the low-dimensional latent variable space to the space of GMM means and is fixed for all utterances. First, we need to express the data likelihood in terms of posteriors of the hidden variables. To avoid confusion, we shall mention that, unlike in GMM training, we do not pool all the data together. In the discussion about subspace, $n$ refers to a single utterance.

As we mentioned before, in the iVector model, we use a UBM to generate sufficient statistics for utterances. For each utterance, zero, first and second–order statistics denoted as $\boldsymbol{\gamma}_n, \boldsymbol{\theta}_n$ and $\boldsymbol{\Theta}_n$, respectively, are extracted and used to represent the corresponding utterance.

To simplify the notations in the following equations, we use transformed sufficient statistics. First, we center the sufficient statistics of each utterance with respect to the UBM

$$\tilde{\boldsymbol{\theta}}_n^c = \boldsymbol{\theta}_n^c - \boldsymbol{\gamma}_n^c \mathbf{m}^c. \tag{2.21}$$

where $c$ stands for a Gaussian component in the GMM. This removes the role of $\mathbf{m}$ in (2.18). In the next step we normalize first order sufficient statistics and the $\mathbf{T}$ by means of UBM covariance as:

$$\begin{aligned} \hat{\boldsymbol{\theta}}_n^c &= \boldsymbol{\Sigma}^{(c)-1/2} \tilde{\boldsymbol{\theta}}_n^c, \\ \hat{\mathbf{T}}^c &= \boldsymbol{\Sigma}^{(c)-1/2} \mathbf{T}^c, \end{aligned} \tag{2.22}$$

where $\boldsymbol{\Sigma}^{(c)-1/2}$ is a Cholesky decomposition of the inverse of $\boldsymbol{\Sigma}^{(c)}$. The $\mathbf{T}^c$ is a D×r dimensional sub-matrix of $\mathbf{T}$ corresponding to the mixture component $c$. $r$ is the dimensionality of the subspace. Note that after normalizations in (2.22), $\boldsymbol{\Sigma}$ becomes the identity matrix.

### 2.4.1 Likelihood function

So far, we showed how to represent utterance specific GMM means in terms of the iVector model parameter $\mathbf{T}$ and an utterance dependent latent variable $\phi_n$. The log likelihood of a data set, $\mathbf{O}$, given corresponding utterance dependent GMM models, $\boldsymbol{\Omega}$ is:

$$\log p(\mathbf{O}|\boldsymbol{\Omega}) = \sum_n \sum_i \log \mathcal{N}(\mathbf{o}_{ni}; \boldsymbol{\Omega}_n) \tag{2.23}$$

The $\mathbf{\Omega}_n$ stands for utterance specific model parameters that comprise the utterance specific mean, $\boldsymbol{\mu}_n$ and shared covariance matrix and weights. By expanding the likelihood function in (2.14) in iVector form and using transformed sufficient statistics in (2.21) and (2.22), the likelihood function in the iVector form becomes

$$\log p(\mathbf{O}_n|\boldsymbol{\phi}_n, \mathbf{T}) = \boldsymbol{\phi}_n^T \hat{\mathbf{T}}^T \hat{\boldsymbol{\theta}}_n - \frac{1}{2}\boldsymbol{\phi}_n^T \hat{\mathbf{T}}^T \boldsymbol{\gamma}_n \hat{\mathbf{T}} \boldsymbol{\phi}_n + const. \tag{2.24}$$

where all terms independent of $\hat{\mathbf{T}}$ or $\boldsymbol{\phi}_n$ are included in the constant.

### 2.4.2 Posterior distribution of hidden variables

To estimate the model parameter $\mathbf{T}$, we need to have the utterance dependent latent variable $\boldsymbol{\phi}_n$ defined. In our calculation we use $\hat{\boldsymbol{\phi}}_n$ as a MAP point estimate of $\boldsymbol{\phi}_n$. To do so, we first need to get the posterior distribution of $\boldsymbol{\phi}_n$. We know that

$$p(\boldsymbol{\phi}_n|\mathbf{O}_n, \mathbf{T}) \propto p(\mathbf{O}_n|\boldsymbol{\phi}_n, \mathbf{T}) p(\boldsymbol{\phi}_n). \tag{2.25}$$

Since both $\log p(\boldsymbol{\phi}_n, \mathbf{o}_n|\mathbf{T})$ and $\log p(\boldsymbol{\phi}_n)$ are quadratic functions of $\boldsymbol{\phi}_n$, it follows that $\log p(\boldsymbol{\phi}_n|\mathbf{o}_n, \mathbf{T})$ is quadratic function of $\boldsymbol{\phi}_n$ and therefore $p(\boldsymbol{\phi}_n|\mathbf{o}_n, \mathbf{T})$ is Gaussian distributed.

$$p(\boldsymbol{\phi}_n|\mathbf{O}_n, \mathbf{T}) = \boldsymbol{\mathcal{N}}(\boldsymbol{\phi}_n; \hat{\boldsymbol{\phi}}_n, \mathbf{L}_n^{-1}), \tag{2.26}$$

where the mean $\hat{\boldsymbol{\phi}}_n$ can be shown to be:

$$\hat{\boldsymbol{\phi}}_n = \mathbf{L}_n^{-1} \hat{\mathbf{T}}^T \hat{\boldsymbol{\theta}}_n \tag{2.27}$$

and $L_n$ is the precision matrix:

$$\mathbf{L}_n = \mathbf{I} + \sum_{c=1}^{C} \gamma_n^c (\hat{\mathbf{T}}^c)^T \hat{\mathbf{T}}^c \tag{2.28}$$

### 2.4.3 Estimation of the hyper parameter T

Our objective for training the model parameter $\mathbf{T}$ is to maximize the data likelihood where likelihood for each utterance is marginalized over the latent variable:

$$p(\mathbf{O}_n|\mathbf{T}) = \int p(\mathbf{O}_n|\boldsymbol{\phi}_n, \mathbf{T}) p(\boldsymbol{\phi}_n) \mathrm{d}\boldsymbol{\phi}_n \tag{2.29}$$

However, we can also conventionally express the likelihood in terms of $p(\mathbf{O}_n|\bar{\boldsymbol{\phi}}_n, \mathbf{T})$ from (2.24) and $p(\bar{\boldsymbol{\phi}}_n|\mathbf{O}_n, \mathbf{T})$ from (2.26) using any $\bar{\boldsymbol{\phi}}_n$ as follows: using Bayes rule we can write:

$$p(\mathbf{O}_n|\mathbf{T}) = \frac{p(\mathbf{O}_n|\bar{\boldsymbol{\phi}}_n, \mathbf{T}) p(\bar{\boldsymbol{\phi}}_n)}{p(\bar{\boldsymbol{\phi}}_n|\mathbf{O}_n, \mathbf{T})}. \tag{2.30}$$

Notice that (2.30) holds for any value of $\bar{\boldsymbol{\phi}}_n$. Using (2.24), (2.26) and choosing $\bar{\boldsymbol{\phi}}_n = \mathbf{0}$[1], we can write the logarithm of (2.30) as:

$$\begin{aligned} \log p(\mathbf{O}_n|\mathbf{T}) &= \log p(\mathbf{O}_n|\bar{\boldsymbol{\phi}}_n, \mathbf{T}) - \log p(\bar{\boldsymbol{\phi}}_n|\mathbf{O}_n, \mathbf{T}) + const. \\ &= -\log|\mathbf{L}_n| \ + \frac{1}{2}\hat{\boldsymbol{\phi}}_n{}^T \mathbf{L}_n \hat{\boldsymbol{\phi}}_n + const. \end{aligned} \tag{2.31}$$

Leaving out the terms that are constant w.r.t. $\mathbf{T}$, we can write our ML objective function for the $n^{th}$ utterance and for the whole train set as:

$$\begin{aligned} \mathcal{L}_n(\mathbf{T}) &= -\log|\mathbf{L}_n| \ + \frac{1}{2}\hat{\boldsymbol{\phi}}_n{}^T \mathbf{L}_n \hat{\boldsymbol{\phi}}_n, \\ \mathcal{L}(\mathbf{T}) &= \sum_n \mathcal{L}_n(\mathbf{T}). \end{aligned} \tag{2.32}$$

We shall mention that so far in this section we described the likelihood function in (2.32) that we calculate after each update of the $\mathbf{T}$ to make sure that the corresponding update results in a higher likelihood. Estimation of the $\mathbf{T}$ matrix is described in the following:

**Expectation Maximization**

For maximum likelihood estimation of the $\mathbf{T}$ matrix, we use the EM algorithm as described in [9]. Given an initial value for $\mathbf{T}$ as $\mathbf{T}_0$, an auxiliary function $\mathcal{Q}$, is defined as:

$$\mathcal{Q}(\mathbf{T}|\mathbf{T}_0) = \sum_n E[\log p(\mathbf{O}_n, \boldsymbol{\phi}_n|\mathbf{T})]. \tag{2.33}$$

[1] $\bar{\boldsymbol{\phi}}_n$ refers to a certain value for the random variable $\boldsymbol{\phi}_n$

Based on (2.19) and (2.26), we can rewrite (2.33) as:

$$\begin{aligned}\mathcal{Q}(\mathbf{T}|\mathbf{T}_0) &= \sum_n E[\log p(\mathbf{O}_n|\boldsymbol{\phi}_n, \mathbf{T}) + const] \\ &= \sum_n E[\boldsymbol{\phi}_n^T \hat{\mathbf{T}}^T \hat{\boldsymbol{\theta}}_n - \frac{1}{2}\boldsymbol{\phi}_n^T \hat{\mathbf{T}}^T \boldsymbol{\gamma}_n \hat{\mathbf{T}} \boldsymbol{\phi}_n + const], \\ &= \sum_n tr(\mathbf{C}_n \hat{\mathbf{T}}^T) - \frac{1}{2} tr(\mathbf{A}_n \hat{\mathbf{T}}^T \boldsymbol{\gamma}_n \hat{\mathbf{T}}) + const\end{aligned} \tag{2.34}$$

where

$$\begin{aligned}\mathbf{C}_n &= \hat{\boldsymbol{\theta}}_n \hat{\boldsymbol{\phi}}_n \\ \mathbf{A}_n &= \hat{\boldsymbol{\phi}}_n \hat{\boldsymbol{\phi}}_n^T + \mathbf{L}_n\end{aligned} \tag{2.35}$$

More detail on defining the auxiliary function is given in [29]. Taking the derivative of (2.34) with respect to each sub-matrix corresponding to the GMM component $c$, gives us:

$$\frac{\partial \mathcal{Q}(\mathbf{T}, \mathbf{T}_0)}{\partial \hat{\mathbf{T}}^c} = \sum_n (\mathbf{C}_n^c)^T - \left( \sum_n \gamma_n^c \mathbf{A}_n^c \right) (\hat{\mathbf{T}}^c)^T \tag{2.36}$$

Setting the derivatives to zero, it gives us a closed form solution for the hyper parameter matrix $\mathbf{T}$ as:

$$\hat{\mathbf{T}}^c = \mathbf{C}^c (\mathbf{A}^c)^{-1}, \tag{2.37}$$

where $\mathbf{A}^c$ and $\mathbf{C}^c$ are the accumulators collected in the E-Step using (2.35) as:

$$\begin{aligned}\mathbf{C}^c &= \sum_n \hat{\boldsymbol{\theta}}_n \hat{\boldsymbol{\phi}}_n^T \\ \mathbf{A}^c &= \sum_n \gamma_n^{(c)} (\mathbf{L}_n^{-1} + \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T)\end{aligned} \tag{2.38}$$

**Minimum Divergence**

In the original recipe for the EM estimation of the hyper parameter $\mathbf{T}$ in [35], there are two different updates. The first one is what we have just

explained and the second one is called *Minimum Divergence*. In fact the latter is just a modification to the first estimation so that the assumption of a standard normal prior on $\boldsymbol{\phi}$ holds during the hyper parameter estimation. That helps us to get a faster convergence. The *minimum divergence* step can be done separately or jointly with the update of $\hat{\mathbf{T}}^c$. For the sake of computational overhead, we do it jointly with the $\hat{\mathbf{T}}^c$ update. This requires collecting the following statistics along with $\mathbf{C}$ and $\mathbf{A}^c$ as:

$$\begin{aligned} \overline{\mathbf{y}} &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\phi}_n \\ \mathbf{P}^{-1} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{L}_n + (\boldsymbol{\phi}_n - \overline{\mathbf{y}})(\boldsymbol{\phi}_n - \overline{\mathbf{y}})^T, \end{aligned} \tag{2.39}$$

The joint update of $\hat{\mathbf{T}}^c$ is given as:

$$\hat{\mathbf{T}}_{em}^{(c)} = \mathbf{C}\mathbf{A}^{(c)^{-1}}\hat{\mathbf{T}}_{md} = \mathbf{T}_{em}\mathbf{J}, \tag{2.40}$$

where $\mathbf{J}$ is the symmetrical Cholesky decomposition of $\mathbf{P}^{-1}$ as $\mathbf{J}\mathbf{J}^T = \mathbf{P}^{-1}$.

### 2.4.4 iVector versus joint factor analysis

Joint factor analysis (JFA) was a successful proposal of the subspace modeling that was originally proposed for SID [37]. It was successfully adapted for LID [18] later on. The iVector model for continuous features as the most recent and the most successful proposal of the subspace modeling in SID and LID was inspired by JFA model. However, we shall mention that there is an important difference between these two models that gives the iVector model a separate characteristic. In adapted JFA model for LID, *language specific* GMM parameters are allowed to move in the subspace corresponding to channel variability. This was used to adapt language models to the channel of each utterance as:

$$\boldsymbol{\mu}^{n,k} = \boldsymbol{\mu}^k + \mathbf{U}\mathbf{x}^{n,k}, \tag{2.41}$$

where $\boldsymbol{\mu}^{n,k}$ and $\boldsymbol{\mu}^k$ are the adapted and the original supervector of GMM $k$ parameters, respectively. $\mathbf{U}$ is a low rank matrix projecting the channel factors subspace, in the supervector domain [18]. In other words, calculating the likelihood of a given utterance for each of the target languages

is still done in the space of the GMM parameters. On the other hand, the iVector model constrains the *utterance specific* GMM parameters to live in a low-dimensional subspace and uses a MAP point estimate of this low-dimensional subspace to represent the corresponding utterance [21]. Calculating the likelihood of an utterance specific iVector with respect to a target language is done in the iVector space by means of a statistical classifier trained on the iVectors corresponding to labeled utterances [46]. The successful application of the iVector model for continuous features in the LID and SID inspired us to base this thesis on the iVector model idea and propose our iVector model for discrete features.

# Chapter 3

# Prosodic speech modeling

Processing of speech prosody is another approach to extract information for language discrimination. It has shown success in SID [23][20][39] and LID [45][56] tasks. The change in the speech loudness and the fundamental frequency (also referred to as pitch) are important characteristics of the speech prosody. As a measure of loudness, the short-term energy of the speech signal can be used. In this work we refer to pitch and energy values of each speech frame as prosodic features. Many algorithms have been proposed for pitch extraction. A common solution is to estimate the fundamental frequency using cross-correlation of the time domain signal in the voiced part of speech. The pitch and energy trajectories of a signal are depicted for a sample speech signal in Figure 3.1 and we refer to them as prosodic feature contours. We shall mention that a typical pitch value is in the range of $80 - 250$Hz and since the pitch and the energy contours are depicted in the same figure, variations of the pitch contour is smoothed out. However, human ears are very sensitive to small variations in pitch contour.

Pitch and energy contours in Figure 3.1 show lots of variation within a sentence. To get a better representation of speech prosody, it is better to analyze pitch and energy values within shorter segments. The shorter the segment is, the less abruption is observed in the values of prosodic features. Analyzing pitch and energy values in syllable-sized segment has been shown to be an effective tool to discriminate among different speakers or languages [61]. For many non-tonal languages, the pitch values stays fairly smooth in syllable-sized segments. However, the pitch is only defined for the voiced frames. On the other hand, energy values may change a lot in syllable-sized windows. Fitting curves to the values of the pitch in syllable-sized segments by means of polynomial basis functions was proposed in [61]. In this framework, within each defined segment and using basis

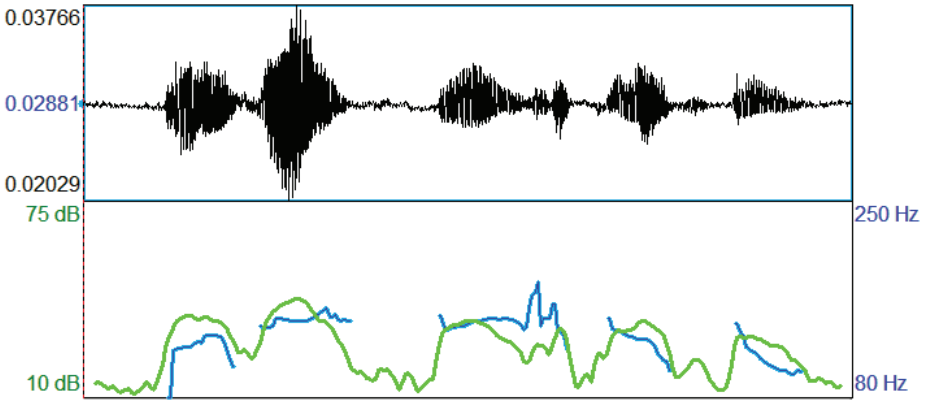



Figure 3.1: F0 (blue) and energy (green) contours extracted for a speech sentence.

functions, a curve is fit to pitch values. The coefficients of the basis functions (referred to as regression coefficients) are then used as a feature vector representing the corresponding segment of speech. A similar feature vector can be extracted for the energy values within each segment of speech. This gives us a fixed length feature vector for each segment that can be modeled with the GMM-UBM paradigm [62] similar to standard continuous features explained in Section 2. In [20], a JFA style subspace modeling approach was proposed to model such feature vectors. After the proposal of the iVectors in [21], a subspace modeling approach based on iVectors was studied in [39]. A similar approach was used for the LID problem in [45].

In this work, I develop a prosodic LID system by adapting the recipe proposed in [39] and [45]. The main goal of developing a prosodic LID system was to compare performance of the different approaches and their contribution to a final system fusion.

The prosodic front-end for the LID system in our implementation involves the following steps:

1. Pre-processing of the speech signal (e.g. frequency filtering, normalizing etc. see Section3.1).
2. Extracting prosodic features (e.g. pitch and energy).
3. Segmenting the speech signal to syllable-sized segments.
4. Fitting curves to the prosodic feature values using Legendre polynomials and representing the contours in terms of regression coefficients.

5. Subspace modeling of regression coefficients with iVector model.

## 3.1 Speech preprocessing

The goal of the pre-processing is mainly to remove interfering effects from the speech signal caused by external factors. This mainly comprises band pass filtering and voice activity detection (VAD). The speech signal is normally collected from different and occasionally diverse channels. In the case of telephony speech, it has already gone through a bandpass filter since a common frequency band in telephony is 300-3400Hz. However, a close look at the signal spectrum shows some harmonics in the lower frequencies below 300Hz. That is normally caused by channel noise. To be safe, we send every speech signal through a bandpass filter of 300-3400Hz to remove any noisy channel effect. Typical pitch values are in the range of 85-180Hz for male speakers and 165-255Hz for female speakers [80]. Nevertheless, the fundamental frequency can be still estimated from harmonics of the signal.

Another useful preprocessing is applying the VAD. It helps us to focus on the informative part of the signal by excluding non-informative part of the signal. Normally a phoneme recognizer that is trained on similar kinds of data is used to classify speech signal into speech and non-speech segments.

## 3.2 Basic prosodic features

**Pitch**

Pitch is a perceptual property that allows the ordering of sounds on a frequency-related scale [38]. It can not be represented in a quantitative way. However, the fundamental frequency, F0, seems to be strongly correlated with the perceived pitch. In this work, we may interchangeably use the value of F0, referring to the pitch. However, we shall mention that they are not identical. There are many different algorithms for estimating the F0 value. One popular algorithm that we use in this work is based on a robust algorithm for pitch tracking (RAPT) proposed in [77]. It uses normalized cross correlation function (NCCF) over the time signal. The output of the algorithm is the estimated F0 and probability of voiced frames. If a frame is unvoiced it outputs zero as the probability of being voiced frame and if the pitch tracker algorithm can not find any F0 value, it outputs 0 as the pitch value for the corresponding frame. We use a frame rate of the 10ms throughout this work. The blue line in Figure 3.1[1] shows the F0 contour

[1]Figure is produced using wavesurfer that is publicly available at

estimated using the RAPT algorithm.

**Energy**
Another important prosodic feature is the speech loudness. It is normally measured in terms of the signal energy [5] and it can be either directly estimated from the speech signal or from the squared magnitude of the signal short term spectrum. We use the same 10ms frame rate to extract the short term signal energy. The Green line in Figure 3.1 shows the short term signal energy contours for the same utterance.

## 3.3 Segmentation

As we mentioned before, we are trying to analyze prosodic features in shorter segments where the prosodic feature values are smoother. For this purpose, syllable-sized segments are popular units since in many languages prosodic characteristics of the speech (like stress, rhythm and loudness) are affected by the syllable. In some other languages (e.g Mandarin), syllables are the basic blocks of the language. However, there is no unique definition of syllable structure. To see what are the possible solutions, let us first explain the structure of a syllable.

The syllable is a linguistic unit that consists of three parts:

- onset: A group of one or more consonants.
- nucleus: A vowel or, more specifically, a voiced sound.
- coda: A group of one or more consonants.

Depending on the language or linguistic rules, the nucleus can be merged with onset in left-branching or the coda in right-branching segmentation. What remains fixed is that in each syllable there should be at least one vowel (or a syllabic consonant). A common solution for the syllable segmentation is the use of phoneme recognizer [39]. This way, we can obtain a typical syllable segmentation by applying the following heuristic rules:

- Run a phoneme recognizer (of possibly mismatched language)
- Map the output to one of the three classes: silence, vowel and consonant
- For two consecutive vowels, do

https://www.speech.kth.se/wavesurfer/


Figure 3.2: Defining syllable segmentation based on phoneme recognizer output.

1. If there is less than two consonants in between, draw syllable boundary before the second vowel
2. otherwise, draw the syllable boundary before the last consonant.

This algorithm is shown in Figure 3.2. Note that the phoneme recognizer is normally trained on a different language than the processing data and the phoneme boundaries are not that accurate. To alleviate this problem one can use a multilingual phoneme recognizer with many phonemes in its acoustic model set.

As we mentioned before, loudness is one of the prosodic features that is affected by the syllable and normally boundaries of the syllables lay in areas with lower energy values. In [20] segmentation of speech based on local minima in the energy contour was proposed for defining the syllable boundaries. This approach was first used by [43] for the LID problem. Another solution proposed in [23] is based on the output of a Large Vocabulary Continuous Speech Recognizer (LVCSR).

**Fixed-length segmentation**
Use of a phoneme recognizer or an LVCSR system to define the syllable boundaries rely on the output of the third party systems and this makes the segmentation more complicated and computationally expensive. An interesting alternative was proposed in [40] that is inspired by the MFCC feature extraction. In this approach, the speech signal is segmented into a fixed typical length syllable units with a few frames shift. This produces highly overlapped fixed length segments that will cause lots of redundancy in the produced features. However, similar to MFCC feature extraction, the GMM model will learn the useful information in the feature vectors.

## 3.4 Fitting curves to feature contours

Let us first explain why we want to fit a curve to prosodic feature contours. The first question is why do we not use the absolute values of prosodic features as our feature vector? There are two main reasons: firstly, the absolute

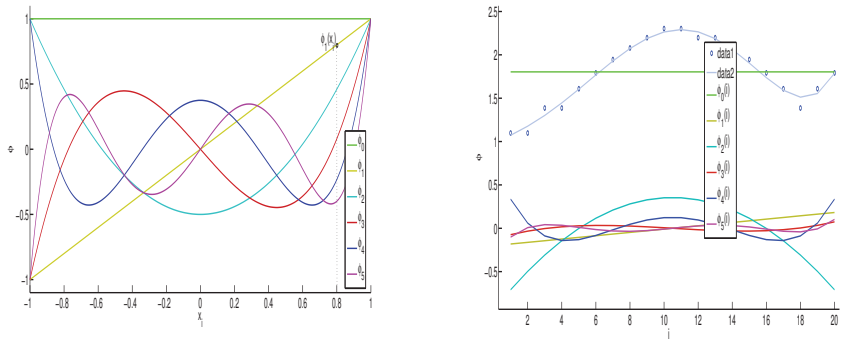
values of prosodic features are very noisy and second and most importantly, not all the frames have corresponding prosodic features. For example the pitch estimation value may fail to produce pitch value for a frame or pitch may not be defined for some frames (e.g unvoiced frames). This means that we can not represent all speech segments with fixed length feature vectors. By regression of prosodic features within a segment, we obtain a smoothed curve that represents the change in the prosodic features in the corresponding segment. For the regression, we are interested in expressing values of the prosodic features within each segment by a linear combination of basis functions. Two popular classes of basis functions used in SID and LID communities are:

- Legendre polynomials (LP) [1]
- Discrete cosine transform (DCT) basis functions

We can achieve a perfect curve fit by using high order of the basis functions. However, this would result in curve over-fitting [8, Chapter 1]. Experiments showed that, the first 6 basis function in both LP and DCT basis functions, give us reasonable curves [40]. This, however, requires at least 6 contiguous values in the prosodic contour of each segment. It may happen that there are missing values in the prosodic features that is caused by silence or unvoiced phonemes. In [40] a median filter is used to fix a discontinuity of 1 frame. However, in case of longer discontinuities the whole segment is discarded and no feature vector is extracted. In this work, we use LP as our basis functions with a different algorithm for the curve fitting that is based on a regression recipe proposed in [8, chapter 4]. The Legendre polynomials are defined in the domain of $-1$ to 1 as shown in Figure 3.3(a). Given that our regression problem is a curve fitting for $N$ data points, we need to get the value of LPs for $N$ points corresponding to the $N$ frames in the fixed length segmentation of the pitch and energy values as explained in Section 3.3. These points are referred to as $x_i$ then obtained as

$$\begin{aligned} x_i &= \frac{2 * i}{N - 2} - 1, \\ i &= 0 \ldots N - 1 \end{aligned} \tag{3.1}$$

Then assume that values of the first $M + 1$ LPs in each of $N$ frames are stored in a matrix as:



(a) First six LP basis function.

(b) Regression of LP basis to fit curves to a prosodic feature contour.

Figure 3.3: Curve fitting to F0 contours using the first six Legendre polynomial basis.

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_M(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_M(x_N) \end{bmatrix} \tag{3.2}$$

The $\phi_m(x_i)$ denotes the value of the LP of order $m$ for the point $x_i$. By assigning different weights to each of the LP functions, we can fit a curve to the pitch and energy contours as:

$$\mathbf{y}^T = \mathbf{w}^T \mathbf{\Phi}, \tag{3.3}$$

where $\mathbf{y}^T$ is a vector holding values of the resulting curve in each of the frame indices and $\mathbf{w}$ is a weight vector. We choose the curve that results in the least squared error between the existing prosodic feature values with those of $\mathbf{y}^T$. The $\mathbf{w}$ that results in such a curve is our solution and is given as [8, chapter 3]:

$$\mathbf{w}_{ML} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{t}. \tag{3.4}$$

where $\mathbf{t}^T = [t_1 ... t_N]$ is a row vector of prosodic feature values. For missing prosodic values, the corresponding rows in (3.2) will be omitted. This is depicted for an example prosodic feature vector in Figure 3.3. The first six LPs are drawn in Figure 3.3(a). In Figure 3.3(b), blue circles are

prosodic feature values. The light blue curve is the regression result of using the first six LPs. The corresponding first six weighted LPs are also shown.

We use the basis function weight vector $\mathbf{w}$ as our feature vector that is going to be modeled in the iVector model. This way, we can also look at the given solution as a feature reduction method that represents prosodic features of $N$ frames with a $(M+1)$–dimensional vector. This way, we also remove the limitation of having defined prosodic feature values for every frame. All we need is to have defined prosodic feature values for $M+1$ frames in each segment.

**Fixed length feature vector**

As we mentioned before our goal of approximating feature contours with LP is to represent speech prosody in segments with a fixed length feature vector so that it can be modeled in iVector model for continuous features. It is important to mention that once we get the segmentation, it applies to all different prosodic features (i.e energy and pitch in this work). The regression coefficients ($\mathbf{w}$) for the different prosodic features of each segment can be stacked together and modeled in the same iVector model as we explained in Section 2.4.

# Chapter 4

# Phonotactic speech modeling

Phonetics and Phonotactics are rich sources of discrimination among languages and are used by human beings to discriminate languages. In phonology, the smallest discrete segment of sound in a stream of speech that carries language-specific information is called a *phone*. A phone can be a voiced or unvoiced sound. It can also be a vowel or a consonant. A voiced sound is produced by vibration of the vocal folds while there is no vibration of the vocal folds during pronunciation of unvoiced sound. On the other hand, if there is no constriction or closure in the vocal tract while pronouncing a phone, it is regarded as vowel. Otherwise it is a consonant. These categories may have overlap as well. All the vowels are voiced but consonants can be voiced or unvoiced. Table 4.1 shows some example of voiced and unvoiced consonants.

From a linguistic point of view, a *phoneme* is a basic element of a given language or dialect, from which, words in that language or dialect are built. The international phonetic association (IPA) defines the phoneme as the smallest segmental unit of sound employed to form meaningful contrasts between utterances [3]. For most of the languages, phonemes are the basic elements of the languages. However, there are syllabic and tonal languages such as Japanese and Vietnamese.

Table 4.1: Sample voiced and unvoiced consonants in English.

| Articulation | Voiceless | Voiced |
|---|---|---|
| Pronounced with the lower lip against the teeth | [$f$] (**f**an) | [$v$] (**v**an) |
| Pronounced with the tongue near the gums | [$s$] (**s**ip) | [$z$] (**z**ip) |

Phonotactics is dealing with combinations of the sounds. Different languages have distinct phonotactics. For example in Czech, a word can start with three consonants without any vowel in between (as in "Brno" ) while in Farsi, a consonant in the beginning of the word must be followed by a vowel. As we move up toward more abstract level in Figure 1.1, there are more structured rules in case of the phonology to discriminate among languages.

## 4.1 Introduction

Language identification performed by humans, similar to other astonishing capabilities of the brain, is a complicated process, that uses all various sources of information in the speech signal in parallel. Among all the knowledge sources, distribution of the sound patterns seems to be one of the main tools of analyzing the speech signal by humans. A typical human being is capable of speaking a native language and the acoustic space that his ears can cover tends to be limited to phones available in his native language. It is a question whether humans are using similar phone units while analyzing acoustic characteristics of an unknown language or not. During the past few decades researchers have been exploring an effective solution for the LID problem, known as phonotactic LID, which is based on tokenizing speech signal to some elementary units and performing statistical analysis on the repetition patterns of the elementary elements. Different elementary units have been explored during these years. In [88] and [82] phonemes were used as the elementary unit. [64] proposed use of techniques in LVCSR for the LID task. [54] used syllable-like units in a parallel tokenizer architecture. More recently, the discriminative power of articulatory features was also explored for the LID task in [71].

Among all different configurations proposed by researchers for phonotactic LID, having single or multiple phoneme recognizers for tokenizing speech is widely used in the community [85][52]. A simple configuration is to let a phoneme recognizer tokenize a speech signal into a stream of phoneme labels. During the training phase, language-specific data is passed to the phoneme recognizer and n-gram statistics are calculated from the generated stream of phonemes based on which, an n-gram language model for the corresponding language is trained. LID is then performed based on calculating the likelihood of the phoneme stream with respect to the language-specific n-gram language models. The n-gram language model represents probability of a phoneme in a context of $n-1$ previous phonemes and is explained in detail in Section 4.3. This configuration is called *phoneme recognizer fol-*

*lowed by language model* (PRLM). We shall mention that the language of the phoneme recognizer and list of the target languages in LID task does not necessarily have to have any overlap. For example, Mandarin training data may be tokenized by a Hungarian phoneme recognizer. This resembles the real world condition where a Hungarian speaker is exposed to the unknown language Mandarin and is trying to model the acoustics of the Mandarin language with his own Hungarian acoustic space.

Researchers have studied the effect of the higher order n-gram language models as well. [55] tried to capture discriminative power of higher order n-grams in a binary tree structure where nodes split and distribution of leaves are estimated in a ML way. Even though an effective order of the n-gram model may depend on the amount of the training data, it is not common to use higher order than 4. It was also shown in [59] that it is rare to find useful information with orders higher than five. A common choice of order for the language model used for the LID task is 3-gram.

An step up to the PRLM system is to have multiple phoneme recognizers in the system front-end. The idea was inspired by the fact that multilingual people can discriminate languages better. This is partly due to the fact that their perception can cover a wider acoustic space and more sounds. With respect to the statistical modeling, having multiple tokenizers in parallel can provide more evidence for the classifier to discriminate among languages. Figure 4.1 shows a parallel tokenizer in a typical *Parallel Phoneme Recognizers followed by Language Models* (PPRLM) configuration. In the conventional PPRLM system, there are multiple phoneme recognizers in the front-end. The language specific training data is decoded by each of these phoneme recognizers and serves as a resource to train an n-gram language model. For the $M$ target languages and $P$ phoneme recognizers, we will have $M \times P$ language models. During the test phase, each test utterance is sent through all the phoneme recognizers. The likelihood of the output stream of each phoneme recognizer is calculated using all the n-gram language models trained for this phoneme recognizer. The produced likelihoods from different language models are, however, not comparable directly since each of them may have used different training data and have different dynamic ranges. To solve this, another module is normally included in a LID system, that calibrates and fuses input scores from heterogeneous sources. It is shown as the *score calibration and fusion* in Figure 4.1.

Even though, having parallel phoneme recognizers is an effective way of providing statistics for language discrimination, training a separate language model for every language in the list of the target languages, based on the output of each of the phoneme recognizers, is complicated. Further-

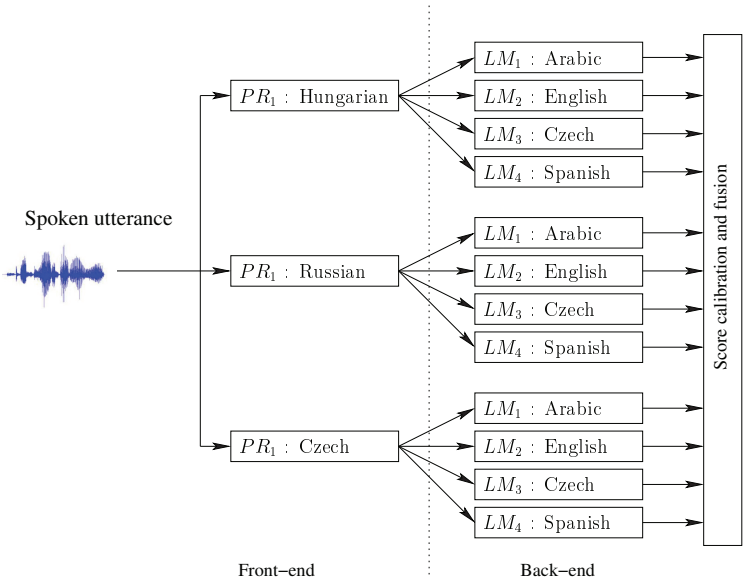



Figure 4.1: General LID system diagram in PPRLM configuration.

more, as the LID task evolved in time since its introduction, dealing with low-resources languages has become an interest in LID tasks and training of an n-gram language model for a low-resource language requires careful smoothing and tuning.

Having a single phoneme recognizer with finer acoustic models covering multiple languages can reduce the number of the required n-gram language models. [32] used 87 phonemes extracted from the multilingual OGI-TS[1] corpus. A set of phonemes with higher discriminative power was proposed in [7]. Even though having a single multilingual phoneme recognizer removes the overhead of training and running multiple phoneme recognizers, it does not outperform the parallel phoneme recognizer architecture. Furthermore, using finer acoustic models in a multilingual phoneme recognizer with a bigger phoneme set would require more training data to estimate a robust n-gram language model.

Most of the phonotactic LID solutions proposed before 2004 are based on extracting n-gram statistics from the single best output stream of the phoneme recognizer. In [28], a new phone lattice based method for automatic language recognition was proposed. They demonstrated that the

[1] http://cslu.cse.ogi.edu/corpora/corpCurrent.html

use of phoneme posteriors from phoneme lattices improves the accuracy of phonotactic LID systems significantly. The phoneme lattices contain alternative hypotheses of phoneme sequences as suggested by the phoneme recognizer. The posterior probabilities of phoneme arcs in the phoneme lattices gives us better estimates of how likely it is to observe those phonemes rather than using hard counts from a single best output. These phoneme posterior probabilities as referred to as the soft counts in this thesis.

Recently, the use of discriminative classifiers has gained lots of attraction. The key is to focus on the class boundaries rather than the distribution of the language classes. In this approach, after decoding the speech signal to a stream of labels, n-gram counts extracted for each utterance are stacked in a fixed length vector. These fixed length vectors (or a transformation of them) are then processed by discriminative classifiers. Use of n-gram soft counts along with support vector machines was proposed in [17] as a successful solution for the LID task. Their proposal simplified the problem of dealing with low-frequent n-gram counts and obviated n-gram language model training for LID task. However, training language models using a huge vector of n-gram statistics is an issue of concern in this approach.

### 4.1.1 Thesis intuition

Let us first bring a short summary of the recent evolution in speech modeling with continuous features (acoustic and prosodic features in this thesis) that has inspired the work on this thesis. For a long time before the proposal of subspace modeling, and in particular the iVector model, training a language specific GMM model or adapting a UBM-GMM to language specific data, was the dominant approach for speech modeling with continuous features. This way, it was assumed that observations $\mathbf{o}_i$ in utterance $n$ are drawn from language specific GMM models:

$$\mathbf{o}_{in} \sim \text{GMM}(\boldsymbol{\mu}_L, \boldsymbol{\Sigma}_L, \boldsymbol{\eta}_L)$$

where $\boldsymbol{\mu}_L, \boldsymbol{\Sigma}_L, \boldsymbol{\eta}_L$ are standing for supervector of language specific means, covariances and Gaussian component weights, respectively. In the training of the GMM model, the assumption is that frames are independent identically distributed (i.i.d assumption). Nevertheless, latent channel factors (e.g. channel, speaker and etc) that are constant in each utterance affect the distribution of the observations and consequently make the i.i.d assumption invalid.

In the iVector model, we assume that an observation $\mathbf{o}_{in}$ in utterance $n$ is drawn from utterance specific GMM that is adapted from the UBM while

constraining the model parameter to live in a low-dimensional subspace as:

$$\mathbf{o}_{in} \sim \text{GMM}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}, \boldsymbol{\eta}),$$
$$\boldsymbol{\mu}_n = \mathbf{m} + \mathbf{T}\boldsymbol{\phi}_n,$$

where $\boldsymbol{\phi}_n$ defines coordinates of the utterance dependent model parameters (the mean supervector) in the low-dimensional subspace. $\boldsymbol{\phi}_n$ is used as a feature vector representing the utterance $n$. This way, the Gaussian component variances in the utterance specific GMM accounts for intra-session variability and the latent variable representing utterance specific GMM accounts for inter-session (e.g. channel, speaker and etc.) variability. The intersession variability modeling is postponed to the system back-end and is done in the low-dimensional subspace. In LID tasks, iVectors are modeled using discriminative classifiers like SVM or logistic regression. In fact, since iVectors are extracted under the assumption of being Gaussian distributed, a simple linear generative classifier may perform even better than SVM [46]. We believe that there are two main conclusions that we can draw from the evolution of language modeling with continuous features in the last decade: first, to postpone modeling of intersession variability to the system back-end in low-dimensional subspace and second, to model language classes discriminatively and to focus on the class borders rather than the language distribution.

In the traditional PRLM approach, we use a phoneme recognizer to generate n-gram statistics. The generated n-gram statistics carry information about both language specific and intersession variability. By training a generative n-gram language model, we treat all this information as language specific information and we assume phoneme $l_i$ to be generated from a language specific n-gram language model $\boldsymbol{\phi}_L$ as:

$$\mathbf{l}_i \sim \text{nGram}(\boldsymbol{\phi}_L)$$

This approach has a similar problem as in training language specific GMMs in acoustic LID: the latent channel factor leads to "corruption" of n-gram statistics of a given utterance, which makes n-gram model assumption (each phone depends only on $n-1$ previous phones) unrealistic. It has been shown that instead of pooling n-gram statistics from language specific utterances together and training a generative n-gram language model for each language, we can put the n-gram statistics extracted from each utterance in a separate fixed length vector and model these language specific vectors of n-gram statistics with SVM [17]. This approach showed better

results than the traditional PRLM approach. However, SVM has to deal with the huge dimensionality in the vectors of n-gram statistics.

At the time of starting this thesis, reducing dimensionality of n-gram statistics with PCA and modeling PCA-transformed n-gram statistics with SVM was the state of art technique used in LID community for NIST LRE09. Even though it gives us a good LID performance, there are issues behind it! The n-gram statistics are counts of discrete events and are not Gaussian distributed. However, since PCA relies on the covariance matrix of the input data, Gaussian distributed data would be more suitable. In fact, n-gram statistics are more multinomial distributed. In Figure 4.2, we show the distribution of PCA-transformed 3-grams taken from the output of the Hungarian phoneme recognizer for the first 4 languages of NIST LRE09 target language list. For the data points depicted in Figure 4.2, the dimensionality of the PCA-transformed n-gram statistics is reduced to 2 using LDA. It is clear from the picture that samples from each class are not drawn from a Gaussian distribution since they are not normally distributed. In other words, modeling languages with SVM using PCA-transformed n-grams is an ad hoc phonotactic LID solution. In fact, for getting a good performance using this solution, lots of SVM parameter tuning is required and as you will see in Section 7.2, the Gaussian linear classifier (as a linear generative classifier) fails to model PCA-transformed n-grams compared to logistic regression and we believe it is due to the distribution of PCA-transformed n-grams .

The goal of this thesis is to apply the same idea as in subspace modeling of continuous features to the discrete n-gram features. As we mentioned in the iVector model for continuous features, utterance specific GMMs are adapted from the UBM while constraining the model parameters to live in a low-dimensional subspace. For the n-gram language model, we assume that utterance specific n-gram language models can be adapted from a background n-gram language model trained over utterances from all languages while we constrain the parameters of the adapted model to live within a low-dimensional subspace. This way, we can represent each utterance specific n-gram language model in terms of the corresponding low-dimensional coordinates in the subspace. This low-dimensional representation can then be used as feature vector to system back-end. We refer to this low-dimensional representation of the utterance specific n-gram language model as a phonotactic iVector. Having a Gaussian distributed phonotactic iVector, we can use a simpler back-end for language classification e.g. Gaussian linear classifier.

In the next sections, we explain basic elements of phonotactic language

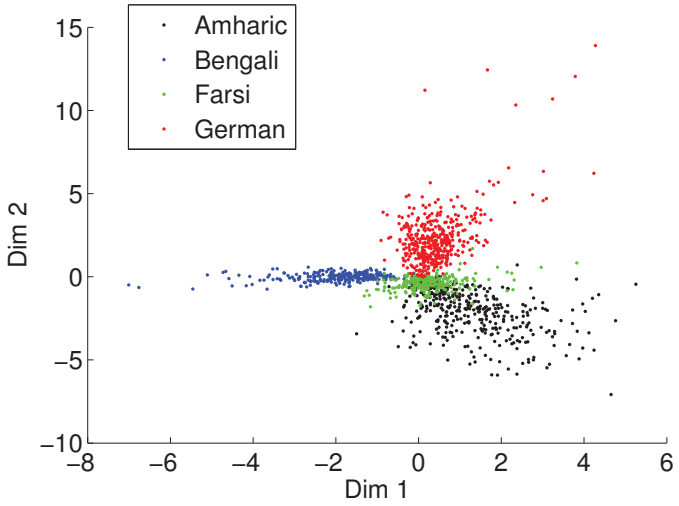



Figure 4.2: Distribution of PCA transformed and mean normalized BUT HU 3-gram statistics for the first four NIST LRE09 languages.

modeling and then we propose our phonotactic iVector extraction technique.

## 4.2 Speech tokenizer

As we explained in Section 4.1, a tokenizer is normally used in phonotactic LID front-end to convert the continuous stream of speech signal into discrete units. In this thesis, we use phoneme recognizers as the tokenizer. It has been shown that the quality of the phoneme recognizer plays a crucial role in the overall performance of an LID system [49][68]. In [47] many different architectures for a phoneme recognizer were studied. One of the most successful ones is a hybrid HMM/NN architecture based on split temporal context [67]. The structure of this phoneme recognizer is depicted in Figure 4.3.

The Mel filter bank energies, which are obtained in the conventional way along with the temporal evolution of critical band spectral densities are used as the input features to the neural network. Use of the split temporal context [66] allows for more precise modeling of the whole trajectory while limiting the size of the model (number of weights in the NN) and reducing the amount of necessary training data. The input for each of the left and right contexts is processed by discrete cosine transform (DCT) in the time domain

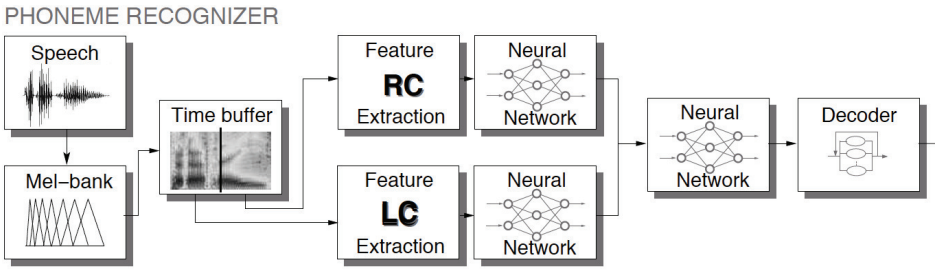



Figure 4.3: Hybrid HMM/NN phoneme recognition block diagram based on split temporal context [67].

to de-correlate and reduce dimensionality. Two NNs are then trained to produce the phoneme posterior probabilities for the left and right contexts. The third NN functions acts as a merger and produces the final set of posterior probabilities [49]. Next, the phoneme posterior probabilities are sent to a Viterbi decoder to obtain phoneme lattices. Only acoustic scores are used for the decoding and no language model is applied. It has been shown that using soft counts extracted from phone lattices rather than hard counts from one-best decoder output, improves performance of the LID system [28]. In fact, we are not looking for exact match of phonemes in the decoded speech. We are trying to obtain statistics from projection of a new language acoustic space to the the phoneme recognizer acoustic space. These statistics are expressed in the form of phoneme counts. Soft counts based on the phone lattices can give us more robust statistics. Furthermore, along with having a high quality phoneme recognizer, it is preferred to have more phonemes in the decoded stream, compared to the phoneme recognition task. This can be adjusted by the phoneme insertion penalty during the decoding. In this thesis, most of the experiments on phonotactic LID are carried out using Hungarian, Russian and Czech phoneme recognizers from Brno University of Technology. We will refer to them as BUT HU, RU and CZ, respectively. They are all trained using the same hybrid HMM/NN architecture. These phoneme recognizers are publicly available[2]. In the case of BUT HU, we merge long and short versions of the same phoneme to achieve a smaller phoneme set with 33 phonemes. This speeds up the experiments without significant degradation of the LID system performance. The corresponding mapping table is given in Appendix B. In all experiments

[2]http://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context

in this thesis, we use soft count n-gram statistics obtained from phoneme lattices. However, for simplicity, we explain all the techniques as if we were using one–best decoded stream of phonemes. The only difference is that, in practice, numbers will be soft counts instead of hard counts.

## 4.3 N-gram model

The joint probability of a phoneme label stream with length $W$ is expressed by the chain rule as:

$$P(l_1 l_2 l_3 ... l_W) = P(l_1)P(l_2|l_1)P(l_3|l_1 l_2)...P(l_W|l_1 ... l_{W-1}) \tag{4.1}$$

With the n-gram assumption, the probability of each phoneme label depends only on the $(n-1)$ previous phonemes. We can write the likelihood of a phoneme stream with length $W$ as:

$$P(l_1 l_2 l_3 \dots l_W) = P(l_1) \prod_{j=2}^{n-1} P(l_j|l_1 \dots l_{j-1}) \prod_{i=n}^{W} P(l_i|l_{i-n+1} \dots l_{i-1}). \tag{4.2}$$

For example, with a 3-gram model assumption, (4.2) becomes:

$$P(l_1 l_2 l_3 \dots l_W) = P(l_1)P(l_2|l_1)P(l_3|l_1 l_2)P(l_4|l_2 l_3) \dots P(l_W|l_{W-2} l_{W-1}). \tag{4.3}$$

Notice that the first 2 terms ($P(l_1)$ and $P(l_2|l_1)$) are evaluated only for the beginning of a sentence and their value dynamic range is normally bigger than the 3-gram probabilities. We can exclude them and approximate the likelihood function as:

$$P(l_1 l_2 l_3 \dots l_W) \approx P(l_3|l_1 l_2)P(l_4|l_2 l_3) \dots P(l_W|l_{W-2} l_{W-1}). \tag{4.4}$$

The ML estimate of 3-gram probabilities are calculated as:

$$P(l_k|l_i l_j) = \frac{\text{freq}(l_i l_j l_k)}{\text{freq}(l_i l_j)}, \tag{4.5}$$

where $\text{freq}(l_i l_j l_k)$ stands for number of occurrences of the phoneme sub-sequence $l_i l_j l_k$. In case of utterance specific 3-grams probabilities, (4.5) is calculated over a sentence and in case of language specific 3-grams probabilities, it is calculated by pooling phoneme sequences of all utterances in a language.

Normally, to get rid of numerical precision problems, logarithms of the probabilities are used. Using the approximation from (4.4) we can write

$$\log(P(l_1 l_2 l_3 \dots l_W)) \approx \sum_{i=n}^{W} \log P(l_i | l_{i-n+1} \dots l_{i-1}). \tag{4.6}$$

This gives us a useful tool for statistical analysis of a language phonotactics.

## 4.4 Feature transformation using principal component analysis

In Principal component analysis (PCA), we are interested to find the directions in which we observe the highest variability in our data. This can be done by finding eigenvectors of the covariance matrix corresponding to the largest eigenvalues.

In the case of phonotactic LID, data consist of vectors of stacked n-gram statistics. By putting these vectors into a matrix, we form a document matrix $\mathbf{L}_{c\times d}$ where $c$ is the number of n-grams and $d$ is the number of utterances in the data. To get the eigenvectors of the covariance matrix corresponding to the biggest eigenvalues, we use a randomized PCA algorithm proposed in [63] that outputs decomposition of the $\mathbf{L}$ matrix in the form of singular value decomposition (SVD) as:

$$\mathbf{L}_{c\times d} = \mathbf{U}_{c\times r} \mathbf{\Sigma}_{r\times r} \mathbf{V}^T_{r\times d}, \tag{4.7}$$

where the $\mathbf{\Sigma}$ is a diagonal matrix holding singular values of $\mathbf{L}$ sorted in descending order. The columns of $\mathbf{U}$ are eigenvectors of $\mathbf{L}\mathbf{L}^T$ and the columns of $\mathbf{V}$ are eigenvectors of $\mathbf{L}^T\mathbf{L}$. $r$ is the number of non-zero (i.e strictly positive) eigenvalues of the covariance matrix where $r \leq \min(c, d)$ [76].

Once we have $\mathbf{U}$, we can project the $\mathbf{L}$ into directions of the eigenvectors corresponding to the largest eigenvalues of the covariance matrix as $\mathbf{U}^T\mathbf{L}$. The transformed $\mathbf{L}$ can then be used as feature vector for phonotactic language modeling.

Researchers have proposed different pre-processing of the document matrix. Authors in [42] proposed to use latent semantic indexing (LSI) techniques. The LSI uses SVD to identify patterns and relationships between the terms and concepts contained in an unstructured collection of text. To apply the LSI for the phonotactic LID, the authors treat phoneme labels as terms in text documents and replace n-gram counts by their corresponding latent

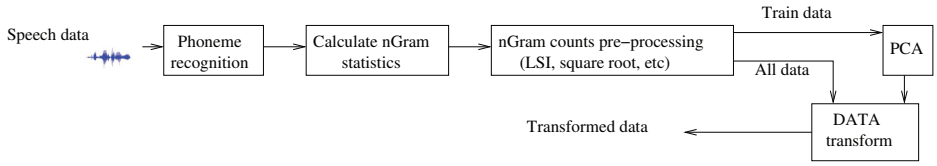



Figure 4.4: Steps in transforming n-gram statistics using PCA.

semantic index (LSI) representations [6] and decompose the document matrix holding the LSI representations with SVD. This chain of transforming n-gram statistics is called *vector space modeling*. Authors in [50] proposed to use square roots of n-gram counts that squeezes dynamic ranges of the n-gram counts and transform the resulting data with $\mathbf{U}$ and they call it *PCA-based feature extraction*. Steps in these two methods are shown in Figure 4.4.

## 4.5 Phonotactic iVectors

The term iVector[3] was adapted by the SID community for the low-dimensional representation of the super vector of GMM means. In general, the iVector refers to an efficient low-dimensional representation of a huge parameter vector that can serve as a feature vector for SID and LID tasks. Let us have a quick review of the iVector model for continuous features and the intuition behind it so that we can explain how our subspace model for discrete features was inspired by acoustic iVectors. We shall mention in the beginning that since we are dealing with discrete features in our proposal of iVector extraction, a different mathematical model than the iVector model for continuous features is deployed.

### 4.5.1 Background

In the iVector model for the continuous features, all the observations are represented by means of sufficient statistics that are generated using UBM. Unlike language specific GMM, we assume that the observations $\mathbf{o}_{in}$ in the utterance $n$ are drawn from an utterance specific GMM and all the utterance specific GMMs share the same $\boldsymbol{\Sigma}$ and $\boldsymbol{\eta}$ (i.e. the same as those in the UBM). We further assume that the the alignments $\gamma^c$ can be taken from the UBM

[3]Many equivalent are proposed for the iVector such as intelligent vector, informative vector and etc.

(i.e. the sufficient statistics can be collected using the UBM instead of the utterance specific GMM). This way we can write:

$$\mathbf{o}_{in} \sim \text{GMM}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}, \boldsymbol{\eta}), \tag{4.8}$$

Then, we limit the utterance specific GMM means supervector $\boldsymbol{\Phi}_n$ to live in a low-dimensional subspace as:

$$\boldsymbol{\Phi}_n = \mathbf{m} + \mathbf{T}\boldsymbol{\phi}_n,$$

where $\mathbf{T}$ is a low–rank matrix that linearly spans the low-dimensional subspace to the original space of the GMM means and $\boldsymbol{\phi}_n$ are the coordinates of the utterance specific GMM model in the low-dimensional subspace. In other words, we adapt the UBM–GMM to each utterance and represent the utterance $n$ with a point estimate of the corresponding model's latent variable $\boldsymbol{\phi}_n$.

In the case of phonotactic LID, the inputs to the system are sequences of phoneme labels for each utterance. Note that the label values are limited to the phoneme recognizer's phoneme set. In other words, this means that the LID system inputs are discrete features. By fixing the phoneme recognizer, we assume that the phoneme set (phonetics) for all the target languages is the same and we try to characterize each target language by the repetition pattern of the phonemes (phonotactics). The information about this repetition patterns are given by n-gram statistics as explained in Section 4.3.

In the traditional n-gram language model, we assume that phoneme $l_i$ is drawn from a language specific n-gram language model $\boldsymbol{\phi}_L$ as:

$$l_i \sim \text{nGram}(\boldsymbol{\phi}_L) \tag{4.9}$$

So all the within-class variability reflected in the n-gram statistics is considered as language specific information and is affecting training of the language specific n-gram language model. In our proposal for phonotactic iVector extraction, we assume that the phoneme $l_i$ is drawn from an utterance specific n-gram language model as:

$$l_i \sim \text{nGram}(\boldsymbol{\phi}_n) \tag{4.10}$$

We expand the $\boldsymbol{\phi}_n$ in a form inspired by the iVector model for continuous features in (2.18). Similar to the iVector model for continuous features, we represent the utterance specific n-gram language model $\boldsymbol{\phi}_n$ with a low-dimensional vector that is later used for training the language model using discriminative or generative classifiers. This way, we leave the intersession variability modeling to the back-end classifiers.

Before moving forward, let us take a closer look at n-gram language modeling. While training a traditional n-gram language model, observations of all phoneme labels $l_i$ with a same history $h$ in the decoded stream of phonemes are treated as set of i.i.d observations drawn from a categorical distribution. The categorical distribution is a special case of the multinomial distribution in which the number of sampled items is fixed at 1 [51][2]. A decoded stream of phonemes for an utterance with length $M$ can be seen as $M$ observations drawn from a multinomial distribution where length of each observation is fixed at 1. The conditional probability $P(l_i|h)$ is calculated using (4.5). Observations of a phoneme label with different histories are considered to be drawn from different multinomial distributions.

In the following sections we first explain how to represent utterance specific multinomial distribution using a low-dimensional vector. To do so, we first assume that all the observations are outputs of a single multinomial distribution and likelihood of an utterance given the observations can be calculated using (4.4). This model is referred to as subspace multinomial model (SMM) and is our first proposal of iVector for discrete features. In the next step, we introduce an extension to the SMM that allows us to model language phonotactics that is consistent with the n-gram model. This latter model is referred to as subspace n-gram model (SnGM).

### 4.5.2 Subspace multinomial model (SMM)

Assume that all the phoneme labels in the decoded sequence are outcomes of a single multinomial distribution. We use a unigram approximation of (4.3) as:

$$P(l_1 l_2 l_3 \dots l_W) \approx P(l_1)P(l_2)\dots P(l_W). \tag{4.11}$$

Now assume that the unigram probabilities in 4.11 are utterance-dependent. Using (4.11), we can write the likelihood of an utterance as:

$$P(O_n) = \prod_{e=1}^{E} (\phi_{en})^{\nu_{en}}, \tag{4.12}$$

where $\phi_{en}$ is the probability of the unigram $e$ in the $n^{th}$ utterance, $\nu_{en}$ is the number of occurrences of the corresponding phoneme in the utterance and $E$ is the total number of the unigrams. As usual, we prefer to work with the logarithm of the likelihood function for the sake of precision and math simplification. The log likelihood of the observation set $\mathbf{O}$ comprising $N$ utterances then becomes:

$$\log P(\mathbf{O}) = \sum_{n=1}^{N} \sum_{e=1}^{E} \nu_{en} \log \phi_{en}. \tag{4.13}$$

Now we can use a model similar to (2.18) to synthesize the utterance specific multinomial distribution. We could be tempted to simply expand the utterance-dependent unigram probabilities similar to (2.18) as:

$$\phi_{en} = m_e + \mathbf{t}_e \mathbf{w}_n, \tag{4.14}$$

where $t_e$ is corresponding row of the $\mathbf{T}$ matrix to the $e^{th}$ multinomial probability and $m_e$ is an offset for $\phi_e$ calculated by pooling all the data together. However, we need to put additional constraints to satisfy the following conditions:

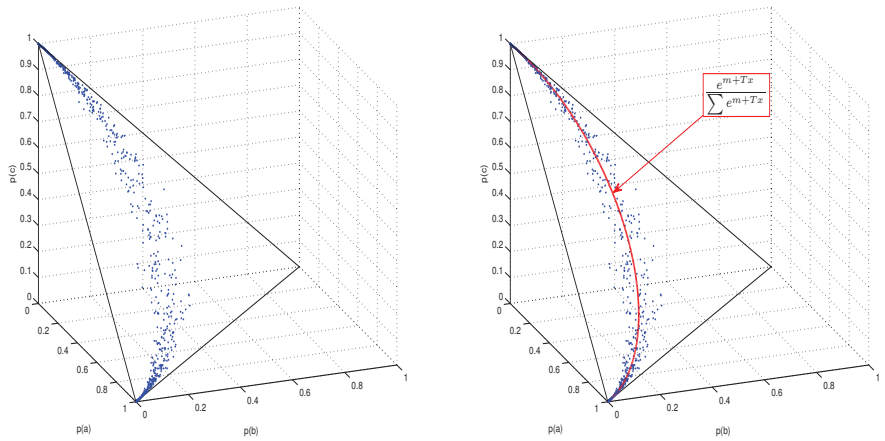
$$\begin{aligned} \sum_{e} \phi_{en} &= 1, \\ \phi_{en} &> 0. \end{aligned} \tag{4.15}$$

Our solution is to represent the "unnormalized" logarithm of the $\phi_{ne}$ using a linear subspace model and normalize it in a softmax style as:

$$\phi_{en} = \frac{\exp(m_e + \mathbf{t}_e \mathbf{w}_n)}{\sum_{j=1}^{E} \exp(m_j + \mathbf{t}_j \mathbf{w}_n)} \tag{4.16}$$

to obtain correct multinomial distribution. Representing the log probabilities in iVector-style and exponentiating it not only guarantees the positive value for the probabilities but also simplifies derivation of (4.13) for the parameter estimation.

In (4.16), the $\mathbf{m}$ is a column vector holding logarithms of unigram probabilities and is calculated by pooling data from all languages. $\mathbf{t}_e$ is the $e^{th}$ row of a low rank hyper parameter matrix $\mathbf{T}$ that linearly spans the subspace, which is not linear in the space of unigram probabilities due to the softmax term in (4.16). $\mathbf{w}_n$ is the low-dimensional vector corresponding to the $n^{th}$ utterance, which determines the position of each utterance specific




(a) Distribution of unigram probabilities for sample data.

(b) Distribution of unigram probabilities obtained by expanding corresponding iVectors.

Figure 4.5: Distribution of unigram probabilities obtained from data and by spanning iVectors.

model in the low-dimensional subspace. In the case of iVector model for continuous features, we can impose a Gaussian prior over the low-dimensional latent vector $\mathbf{w}$. However we do not impose any prior on $\mathbf{w}$ in our subspace multinomial model[4].

Figure 4.5 shows the distributions of unigram multinomial probabilities (Figure 4.5 assumes that we have just 3 unigram multinomial probabilities). The blue dots represent the utterance specific multinomial distribution estimated using ML (see (4.5)). All the blue dots lay on a simplex as a space of all possible 3-dimensional multinomial distributions. The red curve represents all possible distributions (for different $\mathbf{w}_n$; one-dimensional in our example) for a particular choice of $\mathbf{T}$. The depicted red curve is actually composed of red dots corresponding to the distributions that are represented by ML estimated $\mathbf{w}$ as will be explained in Section4.5.3. This is depicted in Figure 4.6.

[4]Perhaps, a better model could be achieved by imposing a prior on the $\mathbf{w}$. However, estimating $\mathbf{w}$ in such a model and in a fully Baysian way is not easily mathematically tractable.

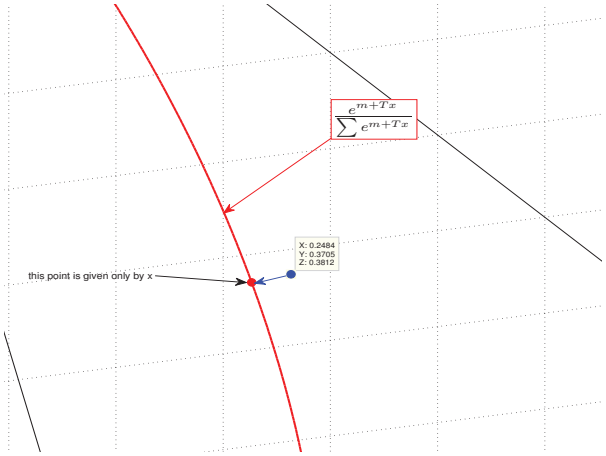



Figure 4.6: ML projection of 3-dimensional multinomial probabilities to subspace multinomial probabilities.

### 4.5.3 Parameter estimation

So far, the problem of estimating utterance-dependent unigram probabilities has effectively translated into estimation of the hyper parameter $\mathbf{T}$, the latent variables $\mathbf{w}$ and $\mathbf{m}$. We can merge $\mathbf{m}$ and $\mathbf{T}$ by setting the last column of $\mathbf{T}$ to the $\mathbf{m}$ and forcing the last row of $\mathbf{w}$ to be 1 for all the utterances. This way, we can jointly estimate $\mathbf{m}$ and $\mathbf{T}$. However, we do not believe that reestimating the offset in each iteration would contribute that much to the system performance. In our implementation, we calculate $\mathbf{m}$ over all the training data as explained in Section 4.5.4 and keep it fixed during the parameter estimation.

Defining the log likelihood in (4.13) as our objective function, we can estimate the model parameters $\mathbf{T}$ and $\mathbf{w}$ using ML parameter estimation. Once the model hyper parameter $\mathbf{T}$ is trained, we can use the trained model for feature extraction by using the point estimate of the latent variable $\mathbf{w}$ as a representation of the corresponding utterance. To estimate $\mathbf{T}$ we need $\mathbf{w}$ and the other way around. We start with an initialization of $\mathbf{T}$ and iterate between estimation of $\mathbf{w}$ based on the fixed $\mathbf{T}$ and the other way round for estimating $\mathbf{T}$. Due to the nonlinearity in (4.16), there is no closed form solution for the maximum likelihood estimation of model parameters $\mathbf{T}$ and latent variable $\mathbf{w}$. We need to resort to a non-linear numerical optimization to solve it. We use a similar quadratic optimization as proposed in [60] that is based on an efficient optimization scheme from [24]. Before explaining

how to estimate $\mathbf{w}$ with fixed $\mathbf{T}$, we need to work on our likelihood function in (4.13). Similar treatment can be used for estimating $\mathbf{T}$ with fixed $\mathbf{w}$.

By substituting (4.16) in (4.13), we can rewrite our objective function as:

$$logP(\mathbf{O}|\mathbf{T},\mathbf{w}_n) = k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \log\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)], \tag{4.17}$$

where $k$ is a constant part that depends neither on $\mathbf{T}$ nor on $\mathbf{w}$. Now, we can use the inequality $1 - x/\bar{x} \leq -\log(x/\bar{x})$, which is an equality at $x = \bar{x}$, to remove the logarithm from (4.17). Considering $x$ to be the normalization term ($\sum_{hi}\exp(m_{hi} + \mathbf{t}_{hi}\mathbf{w}_n)$) and $\bar{x}$ to be its current value (i.e. for current $\mathbf{T}$ and $\mathbf{w}_n$), we can write:

$$\begin{aligned}
&logP(\mathbf{O}|\mathbf{T},\mathbf{w}_n) = \\
&k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \log\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)] = \\
&k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \log\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n) + \log\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n) - \\
&\log\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)] = \\
&k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \log\frac{\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)} - \log\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)] \geq \\
&k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n + 1 - \frac{\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)} - \log\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)],
\end{aligned} \tag{4.18}$$

where $\bar{\mathbf{w}}_n$ is the current value for $\mathbf{w}_n$. Representing all the constant terms with $k'$, we can write the new objective function as:

$$Q = k' + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \frac{\sum_i \exp(m_e + \mathbf{t}_e\mathbf{w}_n)}{\sum_j \exp(m_j + \mathbf{t}_j\bar{\mathbf{w}}_n)}]. \tag{4.19}$$

This makes the denominator constant with respect to $\mathbf{w}_n$ and simplifies the calculus. In fact, the new objective function $Q$, is a lower bound for the objective function in (4.13) with equality for $\bar{\mathbf{w}}_n = \mathbf{w}_n$, which means that,

if we increase the $Q$, the likelihood also increases. For estimation of $\mathbf{w}$, the following Newton Raphson-like update using the the lower bound $Q$ is used:

$$\mathbf{w}_n^{new} = \mathbf{w}_n^{old} + \mathbf{H}_n^{-1}\nabla_n, \tag{4.20}$$

where, $\boldsymbol{\nabla}_n$ is the gradient of the objective function in (4.19) with respect to the $\mathbf{w}_n$ and $\mathbf{H}_n$ is the Hessian (i.e. the matrix of the second derivatives w.r.t. $\mathbf{w}_n$). Taking the first derivative of (4.19) with respect to $\mathbf{w}_n$ gives us the gradient as:

$$\boldsymbol{\nabla}_n = \sum_{e=1}^{E} \mathbf{t}_e^T(\nu_{ne} - \boldsymbol{\phi}_{ne}^{old} \sum_{i=1}^{E} \nu_{in}). \tag{4.21}$$

where $\boldsymbol{\phi}_{ne}^{old}$ is the current estimate of the $\boldsymbol{\phi}_{ne}$. The $\mathbf{H}_n$ is:

$$\mathbf{H}_n = \sum_{e=1}^{E} \mathbf{t}_e^T \mathbf{t}_e \boldsymbol{\phi}_{ne}^{old} \sum_{i=1}^{E} \nu_{in}. \tag{4.22}$$

The full derivations of $\mathbf{H}_n$ and $\boldsymbol{\nabla}_n$ are given in Appendix A. $\mathbf{H}_n$ is a square $r \times r$ matrix where $r$ is the dimension of the subspace. Unfortunately, this quadratic approximation of the objective function gives us a too big update step particularly in the first optimization iteration. As a result the parameter update does not result in higher likelihood which results in frequent update step modification as we will explain later. A similar problem was reported in [60] and we use a similar solution to what was proposed there. The following $\mathbf{H}_n$ is used in our case.

$$\mathbf{H}_n = \sum_{e=1}^{E} \mathbf{t}_e^T \mathbf{t}_e \max(\nu_{ne}, \boldsymbol{\phi}_{ne}^{old} \sum_{i=1}^{E} \nu_{in}). \tag{4.23}$$

In other words, as long as we have not obtained a reasonable estimate for $\boldsymbol{\phi}_{ne}^{old}$, we trust the value of $\nu_{ne}$. This modification speeds up the model convergence significantly as we will see in Section 4.5.5. The $\boldsymbol{\phi}_{ne}^{old}$ is calculated according to (4.16) using the current estimate of $\mathbf{w}_n$.

By fixing the $\mathbf{w}$ and applying the same technique, we get our lower bound of the objective function in (4.13) for estimation of $\mathbf{T}$:

$$Q' = k'' + \sum_n \sum_e \nu_e [\mathbf{t}_e \mathbf{w}_n - \frac{\sum_i \exp(m_e + \mathbf{t}_e \mathbf{w}_n)}{\sum_j \exp(m_j + \bar{\mathbf{t}}_j \mathbf{w}_n)}]. \tag{4.24}$$

We use the following update formula for each $\mathbf{T}$ matrix row $\mathbf{t}_e$:

$$\mathbf{t}_e^{new} = \mathbf{t}_e^{old} + \mathbf{H}_e^{-1}\boldsymbol{\nabla}_e, \tag{4.25}$$

where $\boldsymbol{\nabla}_e$ is the gradient of the likelihood function in 4.24 with respect to $\mathbf{t}_e$ ($e^{th}$ row of $\mathbf{T}$) defined as:

$$\boldsymbol{\nabla}_e = \sum_{n=1}^{N}(\nu_{ne} - \boldsymbol{\phi}_{ne}^{old}\sum_{i=1}^{E}\nu_{in})\mathbf{w}_n^T, \tag{4.26}$$

and $\mathbf{H}_e$ is an $r \times r$ square matrix as an Hessian approximate:

$$\mathbf{H}_e = \sum_{n=1}^{N}\max(\nu_{ne}, \boldsymbol{\phi}_{ne}^{old}\sum_{i=1}^{E}\nu_{in})\mathbf{w}_n\mathbf{w}_n^T. \tag{4.27}$$

The update of the $\mathbf{T}$ or $\mathbf{w}$ in (4.25) and (4.20), respectively, may fail to increase the overall likelihood in (4.13). In that case, we keep reducing the update step in (4.25) or (4.20) by half until it results in higher overall likelihood. If halving the step size does not result in overall likelihood increase after a few times (normally less than 10 tries), we can resume the previous value of $\mathbf{t}_e$ or $\mathbf{w}_n$. The iterations of estimating $\mathbf{t}_e$ and $\mathbf{w}_n$ continue until the change in the overall training data likelihood given the model parameter for consecutive global iterations become negligible. At this point, the model hyper parameter $\mathbf{T}$ is trained and it can be used as a feature extraction model by obtaining the ML estimate of $\mathbf{w}$ for any utterance of interest.

The quadratic optimization of $\mathbf{T}$ and $\mathbf{w}$ in (4.20) and (4.25) is similar to Newton-Raphson optimization. However, the updates are applied for each $\mathbf{w}_n$ and each row of the $\mathbf{T}$ matrix and we always assume independence among different $\mathbf{w}_n$ and $\mathbf{t}_e$. This way, instead of the Hessian matrix used in Newton-Raphson optimization, we get a block diagonal estimate of the Hessian matrix where elements of the block diagonal matrix are calculated using the corresponding equation of (4.27) and (4.23).

The parameter estimation of the model and iVector feature extraction are algorithmically shown in Algorithm 2 and Algorithm 3, respectively.

### 4.5.4 Model initialization

As stated in Section 4.5.3, the value of $\mathbf{m}$ is fixed during estimation of the other parameters. It is calculated as:

$$m_e = \log(\frac{\sum_{n=1}^{N}\nu_{ne}}{\sum_{j=1}^{E}\sum_{n=1}^{N}\nu_{nj}}), \tag{4.28}$$

**Algorithm 2:** Subspace multinomial model training

```
r ← subspace dimension;
m ← initialize with (4.28);
T ← Random small numbers or PCA transform matrix (4.7) over log
probabilities;
w ← 0;
for iter ← 1 to number of the global iterations do
    for wIter ← 1 to number of iteration on w do
        foreach w_n in TRAIN set do
            LL ← calculate log likelihood of n^th utterance using (4.13)
            ;
            do w_wIter update using (4.20);
            newLL ← calculate (4.13) ;
            while newLL < LL do
                reduce w_n update step by half ;
                newLL ← calculate (4.13) ;
        LL ← newLL;
    LL ← calculate log likelihood of the training data using (4.13) ;
    for TIter ← 1 to number of iteration on T do
        for dim ← 1 to number of rows in T do
            do T_TIter update for T rows using (4.25);
            newLL ← calculate (4.13) ;
            while newLL < LL do
                reduce t_dim update step by half ;
                newLL ← calculate log likelihood of the training data
                using (4.13) ;
            LL ← newLL;
```

**Algorithm 3:** Subspace multinomial model, feature Extraction

```
w^old ← 0;
for wIter ← 1 to number of iteration on w do
    foreach w_n corresponding to a utterance do
        LL ← calculate (4.13) ;
        do w_wIter update for w_n;
        newLL ← calculate (4.13) ;
        while newLL < LL do
            reduce w_n update step by half ;
            newLL ← calculate (4.13) ;
        LL ← newLL;
```

where $m_e$ is log of the ML estimate of the $e^{th}$ multinomial probability and $\nu_{ne}$ is the number of occurrences of the corresponding phoneme in utterance $n$. To avoid the problem of getting minus infinity for the multinomial probabilities that are not observed through the whole training set, we filter low–frequent phonemes. More information on input feature filtering is given in Section 7.8. Since we have to have the $\mathbf{T}$ matrix fixed before estimating the $\mathbf{w}$, we need to define an initialization for $\mathbf{T}$. One solution is to initialize the $\mathbf{T}$ matrix with very small random numbers. Another possibility is to set the initial $\mathbf{T}$ to the PCA transformation matrix $\mathbf{U}$ over log probabilities of unigrams as explained in Section 4.4. It is worth mentioning that even though it is proven that estimating each of $\mathbf{T}$ or $\mathbf{w}$ is a convex optimization problem [8, chapter 4], the joint estimation of them does not have a unique solution. For example, any scaling of $\mathbf{w}$ as $\alpha\mathbf{w}$, can be compensated by inverse scaling of $\mathbf{T}$ as $1/\alpha\mathbf{T}$, which gives us a different joint solution for estimating $\mathbf{w}$ and $\mathbf{T}$. Nevertheless, based on the discussion in [79], we believe that there is a region of global maxima for the objective function. Our expectation is that any random initialization of $\mathbf{T}$ should lead us to the intended region of global maxima. Figure 4.7 shows how the data likelihood changes with two different initializations of the $\mathbf{T}$ matrix. It is clear that initializing $\mathbf{T}$ with SVD transformation matrix does not give a better likelihood. The only advantage is that we start with a higher likelihood in the first iteration. After all, both initializations need the same number of iterations for the likelihood convergence. Since estimation of the PCA transformation matrix is computationally expensive, we use the random initialization. In the case of $\mathbf{w}$, we do not need any initialization since the $\mathbf{w}_n$ vectors are estimated using the initialized $\mathbf{T}$. We set $\mathbf{w}_n^{old}$ in (4.20) to zero in

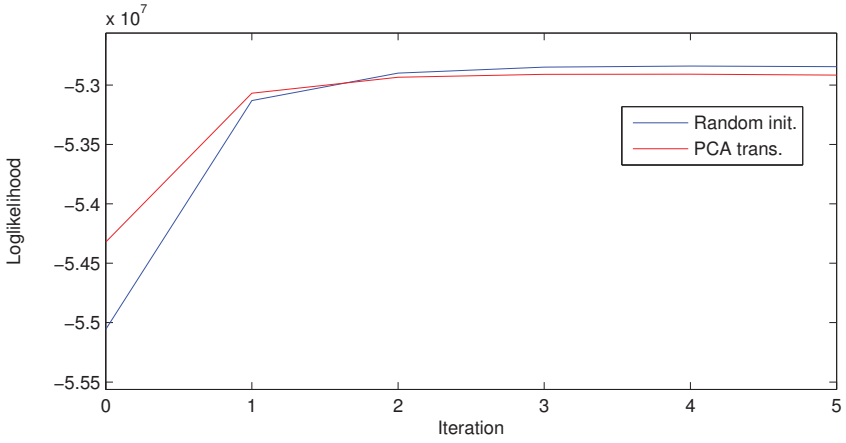



Figure 4.7: Log likelihood change for different **T** initializations for NIST LRE09 with 600 dimensional subspace. The iteration 0 refers to the initialization step.

the first iteration. The model typically seems to converge after 4 iterations.

### 4.5.5 Numerical optimizations

We already explained our quadratic numerical optimization. Here, we give a comparison between standard gradient descent (GD) and our quadratic numerical optimization.

I ) **gradient descent(GD)**: Estimation of the subspace **T** and the iVectors **w** is done with a fixed step size $\alpha$ in the optimization. This can be easily implemented by replacing the **H** matrix in (4.23) and (4.27) with a fixed scalar value $\alpha$. This method is very sensitive to the value of $\alpha$. Since we do not need to calculate the second derivative of the objective function, each iteration becomes much faster than quadratic numerical optimizations. However, even with a proper $\alpha$ value, it takes many more iterations to converge.

II ) **Our quadratic numerical optimization**: This is our main approach. In our solution, to simplify the calculus, we apply two simplifications: first, we use a lower bound for the objective function (4.13) that leads to a simpler second derivative of the objective function and second, to speed up the convergence, we modify the **H** according to (4.27) and (4.23). This modifications were inspired by work in [60]. It

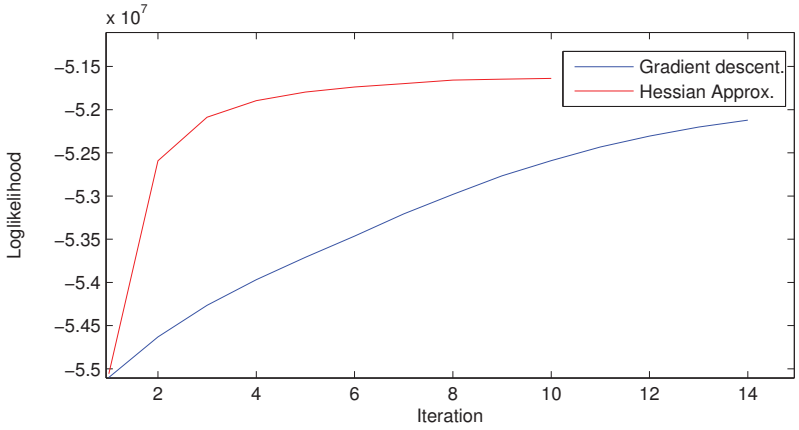



Figure 4.8: Convergence of the quadratic optimization vs. gradient descent over TRAIN set with 600 dimensional subspace.

is claimed that similar modification of the second derivative function, results in faster convergence compared to Newton-Raphson [39].

The speed of convergence (in terms of number of iterations) between GD and our quadratic numerical optimization is depicted in Figure 4.8. It is not easy to give exact comparison between speed of convergence in terms of time since the optimization is done using a computation cluster and many factors can affect the speed. In our implementation, 5 iterations of our quadratic numerical optimization takes roughly the same time as 15 iterations of GD. However, the GD needs many more iterations to produce a likelihood close to the likelihood produced by our quadratic numerical optimization after 4 iterations.

### 4.5.6 Subspace n-gram model

So far, we have explained the SMM and steps toward representing utterance specific unigram probabilities with a low-dimensional vector that we refer to iVector for discrete features. As we mentioned in Section 4.5.1, in the n-gram model, we assume that higher order n-grams with the same history are drawn from the same multinomial distribution and n-grams with different histories are drawn from different multinomial distributions. Here, we explain how to model 3-gram probabilities within the iVector model for discrete features [72].

In Section 4.3 we showed that, assuming the 3-gram model, the likelihood of a sequence of phonemes with length $M$ is:

$$P(l_1 l_2 l_3 \dots l_M) = P(l_3|l_1 l_2) P(l_4|l_2 l_3) \dots P(l_M|l_{M-2} l_{M-1}). \tag{4.29}$$

In order to model the phoneme generation process, we assume that the conditional distribution of a phoneme $l$ given a history $h$ is a multinomial distribution with parameters $\phi_{hl}$:

$$\log P(l|h) = \log \phi_{hl}, \tag{4.30}$$

where $\phi_{hl} > 0$ and $\sum_l \phi_{hl} = 1$. The joint log likelihood of a sequence of phonemes $l_1\ l_2 \dots l_M$ in the general n-gram model can then be computed as:

$$\log P(l_1\ l_2 \dots l_M) = \sum_i\ \log P(l_i|h_i) = \sum_i \log\ \phi_{h_i l_i}, \tag{4.31}$$

where $h_i = (l_{i-n+1} l_{i-n+2} \dots l_{i-1})$ denotes the history for the observed phoneme $l_i$. Let us denote the number of times that phoneme $l$ with history $h$ appeared in a phoneme stream as $\nu_{hl}$. We can rewrite (4.31) as:

$$\log P(l_1\ l_2 \dots l_M) = \sum_i \log P(l_i|h_i) = \sum_h \sum_l \nu_{hl} \log \phi_{hl}, \tag{4.32}$$

where $\sum_l \phi_{hl} = 1$. Note that the objective function that we used in Section 4.5.2 represents a different probability model. The objective in Section 4.5.2 was given as:

$$\log P(l_1\ l_2 \dots l_M) = \sum_i \log P(l_i, h_i) = \sum_e \nu_l \log \phi_e, \tag{4.33}$$

where $\phi_e$ is a multinomial probability and $e$ ranges over all possible phonemes and $\sum_e \phi_e = 1$.

To model 3-gram probabilities with multinomial distributions and extract phonotactic iVectors by maximizing the likelihood function in (4.32), we condition the probability of each phoneme on the corresponding history. This is accomplished by modeling every history with a separate multinomial distribution. We call this model, the subspace n-gram model (SnGM). Using the similar intuition as explained in Section 4.5.2, we can express the conditional probability of $\phi_{hl}$ as:

$$\phi_{hln} = \frac{\exp(m_{hl} + \mathbf{t}_{hl}\mathbf{w}_n)}{\sum_i \exp(m_{hi} + \mathbf{t}_{hi}\mathbf{w}_n)}, \tag{4.34}$$

where $m_{hl}$ is the log-probability of n-gram $hl$ calculated over all the training data, $\mathbf{t}_{hl}$ is a row of a low–rank rectangular matrix $\mathbf{T}$ and $\mathbf{w_n}$ is utterance–specific low-dimensional vector, which can be seen as a low-dimensional representation of the utterance–specific n-gram model. Note that the summation in the denominator is over the n-grams with the same history. The parameters $m_{hl}$ and the matrix $\mathbf{T}$ are the parameters of the proposed SnGM. Normalizing the $\phi_{hln}$ over the n-grams with the same history to satisfy condition of the probability in (4.34) is the main difference between SMM and SnGM. This results in slightly different equations for the first and second derivatives of the objective function in (4.19). We give the new equations along with an enhanced SnGM in Section 4.5.7.

### 4.5.7 Regularized subspace n-gram Model

In the case of iVector model for continuous features, the utterance–dependent parameters $\mathbf{w}_n$ are treated as latent random variables with a standard normal prior. The subspace parameters are then trained using the standard EM algorithm, where the M-step integrates over the latent variable posterior distributions from the E-step. Unfortunately, the calculation of the posterior distribution for $\mathbf{w}_n$ is intractable in the case of SnGM. Instead, SnGM parameters are updated using only $\mathbf{w}_n$ point estimates, which can negatively affect the robustness of SnGM parameter estimation. To mitigate this problem, we propose to regularize the ML objective function using L2 regularization terms for both the subspace matrix $\mathbf{T}$ and the vectors $\mathbf{w}_n$. This corresponds to imposing an isotropic Gaussian prior on both the SnGM parameter ($\mathbf{T}$) and $\mathbf{w}_n$, and obtaining MAP rather than ML point estimates. In order to train our model, we maximize the regularized likelihood function

$$\sum_{n=1}^{N} \sum_{h} \sum_{l} (\nu_{hln} \log \phi_{hln} - \frac{1}{2}\lambda \left\|\mathbf{t}_{hl}\right\|^2 - \frac{1}{2}\lambda \|\mathbf{w}_n\|^2), \tag{4.35}$$

where the sum extends over all $N$ training utterances. The term $\lambda$ is the regularization coefficient for both the model parameters $\mathbf{T}$ and for $\mathbf{w}_n$ and $\|.\|$ stands for the norm of a vector. Notice that we should regularize both $\mathbf{T}$ and $\mathbf{w}$ since limiting the magnitude of $\mathbf{T}$ without regularizing $\mathbf{w}$ would be compensated by a dynamic range increase in $\mathbf{w}$.

### 4.5.8 Parameter estimation

The model parameters $m_{hl}$ are shared for all utterances and can be initialized as the logarithm of the conditional probability of a phoneme given

its history computed over all training utterances (i.e. log ML estimate of n-gram probabilities).

$$m_{hl} = \log\left(\frac{\sum_n \nu_{hln}}{\sum_n \sum_i \nu_{hin}}\right). \tag{4.36}$$

Similar to the SMM, we assume that the terms $m_{hl}$ do not require retraining. In order to alternately maximize the objective function (4.35) with respect to $\mathbf{T}$ and $\mathbf{w}$, we use the same approach as for the subspace multinomial model. The update formulas for $\mathbf{T}$ and $\mathbf{w}$, are the same as (4.25) and (4.20), respectively. Here, we just write the new equations for calculating the gradient and $\mathbf{H}$. The gradient of the regularized objective function in (4.35) with respect to every $\mathbf{w}_n$ for estimating $\mathbf{w}$ with a fixed $\mathbf{T}$ matrix is:

$$\boldsymbol{\nabla}_{w_n} = \sum_h \sum_l \mathbf{t}_{hl}^T(\nu_{hln} - \phi_{hln}^{old} \sum_i \nu_{hin}) - \lambda \mathbf{w}_n, \tag{4.37}$$

where the terms $\phi_{hln}^{old}$ are the n-gram probabilities computed from the current estimate of $\mathbf{w}_n$ and $\mathbf{H}_{\mathbf{w}_n}$ is

$$\mathbf{H}_{\mathbf{w}_n} = \sum_h \sum_l \mathbf{t}_{hl}^T \mathbf{t}_{hl} \max(\nu_{hln}, \phi_{hln}^{old} \sum_i \nu_{hin}) - \lambda \mathbf{I}. \tag{4.38}$$

The gradient of the regularized objective function in (4.35) with respect to every row $\mathbf{t}_{hl}$ of $\mathbf{T}$ is:

$$\boldsymbol{\nabla}_{t_{hl}} = \sum_s (\nu_{hln} - \phi_{hln}^{old} \sum_i \nu_{hin}) \mathbf{w}_n^T - \lambda \mathbf{t}_{hl}, \tag{4.39}$$

and

$$\mathbf{H}_{t_{hl}} = \sum_s \max(\nu_{hln}, \phi_{hln}^{old} \sum_i \nu_{hin}) \mathbf{w_n} \mathbf{w_n}^T - \lambda \mathbf{I}. \tag{4.40}$$

Notice that in (4.39), since we only need the n-gram statistics corresponding to the n-gram history $h$, there is no need to load the whole vector of n-gram statistics. This reduces the memory requirements of the $\mathbf{T}$ matrix update. Moreover, the updates of the $\mathbf{T}$ rows belonging to different histories are completely independent, which simplifies parallel estimation of $\mathbf{T}$.

The matrix $\mathbf{T}$ is initialized with small random numbers. Similar to SMM, update of $\mathbf{T}$ or $\mathbf{w_n}$ may fail to increase the objective function in (4.35). In that case, we do a similar step halving strategy as explained in Section 4.5.3.

Note that in the SnGM, we model each n-gram history with a separate distribution. However, parameters of all the distributions are constrained

to live in a single low-dimensional subspace. In other words, for each utterance, the parameters of all the distributions are controlled with a single low-dimensional vector $\mathbf{w}_n$.

## 4.6 SMM model and 3-gram statistics

In Section 4.5.2 we described the SMM model using the unigram counts as the input features. However, the unigram statistics provide information only about the phonetics of a language and not the phonotactics of a language. The 3-gram statistics can be used as input to the SMM model [74] while using the same mathematics. To do so, we assume that all the 3-gram statistics extracted from the decoded sequence of phonemes for an utterance are drawn from a single multinomial distribution. We shall mention that this treatment of the 3-gram statistics is not consistent with the i.i.d assumption of observations in the SMM model. However, as we will see in Section7.4, we can still get better LID system performance compared to the PCA feature transformation. All the reported results on SMM in Section7 are using 3-gram statistics as the input.

## 4.7 Soft count n-gram statistics

As we mentioned earlier in this chapter, we use n-gram soft counts calculated from phoneme lattices instead of the hard count n-grams based on the single best output of a phoneme recognizer. The only difference for the techniques explained in this chapter is that the repetition of n-gram statistics ($\nu$ in (4.12)) are floating point numbers instead of integers. For generating n-gram soft counts, the output of the third neural network in Figure 4.3 that generates the final phoneme posterior probabilities per frame, is sent to *HVite* from HTK[5] to generate phoneme lattices. The generated phoneme lattices are then sent to lattice-tool[6] to calculate soft count 3-gram statistics.

[5] http://htk.eng.cam.ac.uk/

[6] http://www.speech.sri.com/projects/srilm/manpages/lattice-tool.1.html

# Chapter 5

# Statistical language modeling

So far, we explained extraction of the iVectors for continuous and discrete features. The next step in an LID system is training of language models using the generated iVectors. According to what we depicted in Figure 4.3, we are talking about the back-end in LID system. Figure 5.1 briefly shows what we want to do in the back-end. We refer to all processing steps that result in extraction of iVectors as *feature extraction*. As an example, expansion of this module for phonotactic iVectors is depicted in Figure 5.2. In this chapter, we use the term iVector in general form referring to all kinds of low-dimensional representation of speech utterances that comprise PCA-transformed n-gram features as explained in Section 4.4, phonotactic iVectors as explained in Section 4.5 and acoustic iVectors as explained in Section 2.4.

An specific LID application is usually defined in terms of priors over languages of interest and the cost for making mistakes. As we mentioned before, we deal with the LID problem as defined in NIST LREs. The goal in NIST tasks is to have the lowest cost function that is defined for each NIST LRE (see Section6.4.1). All of the NIST tasks that are studied in this thesis can be broken down to a set of verification tasks that involve making binary decisions of whether the trial utterance belongs to the defined target language or not. To accomplish NIST tasks, it is enough to have class (language) likelihoods so that we can make an optimal Bayesian decision on the language of a trial. The optimal Bayesian decision could be made if our LID system delivers optimal likelihoods for the languages of interest. E.g. if the task is to minimize the probability of language misclassification, we can select the most likely language, where the language posteriors can be obtained using Bayes rule from the priors and likelihoods. Having iVectors, our back-end would be a single multi-class probabilistic classifier (e.g.

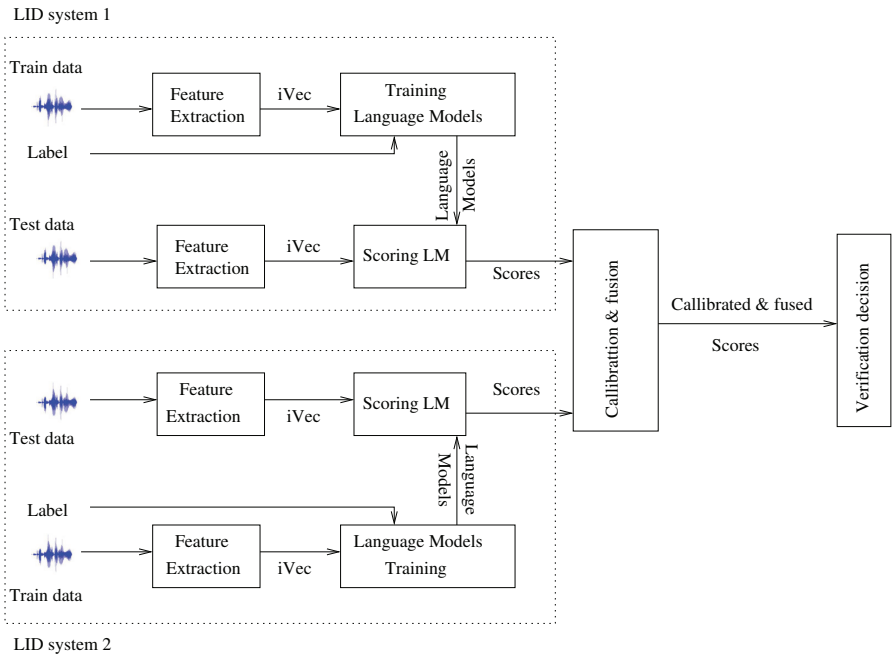



Figure 5.1: Detailed LID back-end block diagram.

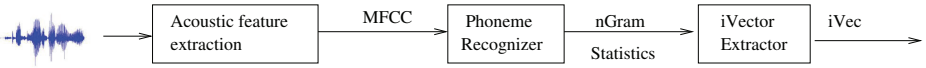



Figure 5.2: Expansion of LID front-end for phonotactic iVectors.

multi-class logistic regression, Gaussian linear classifier and etc.) that takes iVectors as inputs and, by definition, delivers class likelihoods.

An alternative for training multi-class classifiers is to use binary classifiers (binary logistic regression or SVM) to detect each class (i.e. language in our case) against all the other classes. This solution can be used in the cases where we do not want to train multi-class classifiers on the vector representation of the utterances. Note that in this case, we do not obtain class likelihoods from the binary classifiers. Each binary classifier gives us a class score. This means that we need yet another multi-class classifiers to convert the scores obtained from the binary classifiers to the class likelihoods which would be well calibrated and suitable for the final Bayesian decision making.

Even if our classifiers give us class likelihoods (in case of multi-class classifier), the produced class likelihoods may not be well calibrated likelihoods for the target (EVAL) data because of the mismatch between training and EVAL data distribution or because of the incorrect assumption made by the model. For these reasons, the classifiers' scores can be used as features to another multi-class probabilistic classifier, which is trained on possibly smaller but well matched training (development) set to obtain calibrated class likelihoods. The input to the multi-class classifier in the second stage may also be scores from multiple LID subsystems (e.g. phonotactic and acoustic) in order to perform system fusion.

In the following, we describe training of language models in different configurations and, in the next step, we explain different *c*alibration fusion schemes used in this thesis.

## 5.1 Binary language models

A common configuration for training language models is to train a separate two class classifier for each language of interest. This is denoted as *one-vs-all*. To do so, all the training iVectors corresponding to language $k$ are labeled as in–class data and all the other iVectors are considered as out–of–class data. There are many possibilities to train classifiers in this form. In this thesis, we use binary SVM classifiers as proposed in [16] and binary logistic regression (BLR). All the $K$ classifiers in *one-vs-all* configuration receive all the training iVectors. However, the labeling varies for each classifier. In this configuration, some languages may have more training iVectors than others. To alleviate this problem, we use balanced training set of iVectors for all languages as explained in Section 6.2.1. We shall mention that in *one-vs-all* configuration, we have more out–of–class data than of–class data while training the classifiers and to avoid biasing the classifier

toward the out–of–class data, a proper weighting should be applied.

In this thesis, we use SVM and logistic regression (LR) for discriminative modeling of language classes. SVM discriminates between observation from different classes based on a maximum margin model and logistic regression decides on the separation class boundary based on minimizing the cross entropy between the classifier output probabilities and the correct class labels.

Among different optimizations that are proposed to find the class separation boundary (hyperplane), we use the formulation proposed in [8, Chapter 6] for SVM and [8, Chapter 4] for logistic regression. All the derivations for the formula are given in the corresponding chapters of [8]. The implementations of SVM and logistic regression used in this thesis are done by Sandro Cumani from BUT and Niko Brummer from AGNITiO. We briefly describe these discriminative classifiers and provide reference to more descriptive articles on the optimization process.

### 5.1.1 Support vector machines

Support vector machines (SVM) were in principle proposed to discriminate between linearly separable observations of two classes. In linear classifiers, we assume that a set of $N$ $P$-dimensional observations $\mathbf{w}_n$ belongs to either of the two classes $C_1$ or $C_2$ and the hyperplane separating these two classes is defined as:

$$\mathbf{x}^T \mathbf{w}_n + b = 0, \tag{5.1}$$

where $\mathbf{x} \in \Re^P$ and $b$ is and offset.

Assuming that there is a margin between observations of a class and the decision hyperplane (i.e. distance between the hyperplane and the closest observation point) called $M$, the objective in a maximum margin linear classifier is to find $\mathbf{x}$ corresponding to the hyperplane that separates the classes $C_1$ and $C_2$ and maximizes $M$. In [8, Chapter 4], a full optimization process to find class separation hyperplane with a maximum margin using Lagrange multipliers is provided. Using the coding scheme $y_n \in \{-1, +1\}$ to represent class labels for the binary problem, the maximum margin solution classifier is presented in terms of minimizing the error function

$$\mathbf{x}^*, b^* = \arg\min_{\mathbf{x},b} \{ \sum_{n=1}^{N} E_\infty ( \underbrace{(\mathbf{x}^T \mathbf{w}_n + b)}_{\mathcal{D}(\mathbf{w}_n)} y_n - 1) + \frac{1}{2} \|\mathbf{x}\|^2 \}, \tag{5.2}$$

where $E_\infty(z)$ is a function that is zero if $z \geq 0$ and $\infty$ otherwise. Note that based on the error function in (5.2), we have to classify all observations correctly otherwise we get an infinite penalty for a wrongly classified observation. However, in many of the real world problems, class-conditional distributions that result in non-linear separable observations from the corresponding classes may overlap. This problem is addressed by proposing a soft–margin that allows misclassification of observations. To do so, for each observation, $n$ a penalty function $\xi_n$ is defined which is zero for the observations that respect the margin and $\xi_n = |y_n - \mathcal{D}(\mathbf{w}_n)|$ for others. The $\xi_n$ is a linear function of distance between the observation and the decision boundary for the misclassified observations. The soft margin solution SVM is presented in terms of minimizing the error function:

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\mathbf{x}\|^2 , \tag{5.3}$$

where $C$ is a constant that controls the trade-off between the penalty $\xi_n$ and the margin. $\sum_n \xi_n$ is an upper bound on the number of misclassified observations. The $C$ is therefore playing a similar role as the regularization coefficient since it controls the trade-off between minimizing training errors and maximizing the margin (i.e. having low *model complexity*). In case $C \to \infty$ the model changes to SVM with maximum margin. The solution for soft margin SVM corresponds to:

$$\begin{aligned} \mathbf{x}^*, b^* &= \arg\min_{x,b} \{ \frac{1}{2} \|\mathbf{x}\|^2 + C \sum_{n=1}^{N} \xi_n \} \\ &= \arg\min_{x,b} \{ \frac{1}{2} \|\mathbf{x}\|^2 + C \sum_{n=1}^{N} \underbrace{\max\{0, 1 - y_n(\mathbf{x}^T \mathbf{w}_n + b)\}}_{l(\mathbf{x}, \mathbf{w}_n, y_n)} \}, \end{aligned} \tag{5.4}$$

and score for each observation $\mathbf{w}_n$ in this model is calculated as:

$$\mathbf{x}^T \mathbf{w}_n + b. \tag{5.5}$$

An in interesting characteristic of SVM is the possibility to transform a non-linear separable problem in the original space to a linear separable problem in a higher dimensional space by means of non-linear kernel functions. We shall mention that, for the SVM implementations that are used in this thesis, only the linear kernel is used. Binary SVM classifiers are referred to as BSVM in this thesis.

### 5.1.2 Binary logistic regression

In a two class problem, the posterior probability for class $C_1$ can be written as

$$\begin{aligned} p(C_1|w_n) &= \frac{p(\mathbf{w}_n|C_1)p(C_1)}{p(\mathbf{w}_n|C_1)p(C_1) + p(\mathbf{w}_n|C_2)p(C_2)} \\ &= \frac{1}{1 + exp(-a)} = \sigma(a), \end{aligned} \tag{5.6}$$

where we define $a$ as

$$a = \frac{p(\mathbf{w}_n|C_1)p(C_1)}{p(\mathbf{w}_n|C_2)p(C_2)}, \tag{5.7}$$

and $\sigma$ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1 + exp(-a)}. \tag{5.8}$$

For two classes $C_1$ and $C_2$ it holds that $p(C_1|\mathbf{w}_n) + p(C_2|\mathbf{w}_n) = 1$. LR assumes that $a$ can be modeled using a linear function of $\mathbf{w}_n$

$$a = \mathbf{x}^T\mathbf{w} + b. \tag{5.9}$$

Logistic regression provides a discriminative framework to estimate parameters $\mathbf{x}$ and $b$ that maximize the likelihood of training set class labels.

Similar to the binary SVM case, we assume that the observation $\mathbf{w}_n$ belongs to either of $C_1$ or $C_2$ classes with a corresponding label $t_n \in 0, 1$. Let $\gamma_n = \sigma(\mathbf{x}^T\mathbf{w} + b)$ denote the posterior probability $p(C_1|\mathbf{w}_n)$. The log likelihood of the target labels can then be expressed as :

$$\log p(t_1, \ldots, t_n|\mathbf{w}) = \sum_{i=n}^{N}(t_n \log \gamma_n + (1 - t_n) \log(1 - \gamma_n)), \tag{5.10}$$

We can obtain the parameters $\mathbf{x}$ and $b$ by maximizing (5.10). Note that in case of linearly separable classes, there would be unlimited number of solutions with an infinitely long vector $\mathbf{x}$ assigning the training data to the

correct class with probability one. Standard ML estimation of the parameters results in such solution [8, Chapter 4]. To avoid this problem we put a prior on vector $\mathbf{x}$ and obtain its MAP estimate. The use of prior can be interpreted as regularization. This variant of logistic regression is called *regularized logistic regression*. By expanding $\gamma_n$ in (5.10) and adding the regularization term on $\|\mathbf{x}\|^2$, the objective for regularized binary logistic regression is:

$$\mathbf{x}^*, b^* = \arg\min_{\mathbf{x},b}\{\frac{\lambda}{2}\|\mathbf{x}\|^2 + \frac{1}{N}\sum_{n=1}^{N} log(1 + e^{-y_n(\mathbf{x}^T\mathbf{w}_n+b)})\} \tag{5.11}$$

where $\frac{1}{N}$ is just scaling of the error function with respect to the number of the observations. Comparing (5.11) and (5.4) shows similarity of the objective functions for SVM and logistic regression classifiers. Binary logistic regression classifiers are referred to as BLR in this thesis.

## 5.2 Multi-class language models

As we explained in the beginning of this Chapter, we need to train multi-class classifiers either directly on the vector representation of the utterances or on top of language scores obtained by means of binary classifiers. For multi-class SVM and multi-class LR, the parameter $\mathbf{X}$ is actually a matrix of parameters $\mathbf{X} = [\mathbf{x}_1, \dots \mathbf{x}_K]$ representing hyper-planes separating classes and $K$ is number of the classes.

The process is similar to the one-vs-all technique, however, the training stage involves slightly different loss functions and the hyper-planes are jointly optimized.

### 5.2.1 Multi-class logistic regression

Multi-class logistic regression assumes that the posterior for each class can be computed as:

$$p(C_i|\mathbf{w}_n) = \frac{\exp(\mathbf{X}_i^T\mathbf{w}_n + b_i)}{\sum_k \exp(\mathbf{X}_k^T\mathbf{w}_n + b_k)}, \tag{5.12}$$

where $\mathbf{X_i}$ is the matrix of model parameters representing hyperplanes that results in assigning $\mathbf{w}_n$ to class $i$. For the multi-class variant of logistic regression, the multi-class cross entropy loss function

$$l(\mathbf{X}, \mathbf{w}_n, y_n) = -\log \frac{\exp(\mathbf{X}_{y_n}^T\mathbf{w}_n + b_{y_n})}{\sum_{y'} \exp(\mathbf{X}_{y'}^T\mathbf{w}_n + b_{y'})}, \tag{5.13}$$

is used, which again results in an objective function where the probability of the correct labeling of all training examples is maximized. The objective function for multi-class is given as:

$$\mathbf{X}^*, b^* = \arg\min_{\mathbf{X},b}\{\frac{\lambda}{N}tr(\mathbf{X}^T\mathbf{X}) + \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N_k}\sum_{n\in S_k} l(\mathbf{X}, \mathbf{w}_n, y_n)\}, \quad (5.14)$$

where $N = \sum_k N_k$, $N_k$ is the number of training samples for language $k$, $S_k$ is a subset of training utterances corresponding to class $k$ and $N$ is the total number of training utterances. We refer to the first term in (5.14) as *regularization penalty* and the second term as *multi-class cross entropy*. For each observation $\mathbf{w}_n$ the scores are calculated as $\mathbf{X}^T\mathbf{w}_n + b$. Multi-class logistic regression classifiers are referred to as MLR in this thesis.

### 5.2.2 Multi-class SVM

The objective function for the the multi-class SVM [78] is an extension of the binary objective function given by

$$l(\mathbf{X}, \mathbf{w}_n, y_n) = \max_{y'}\{(\mathbf{X}_{y'}^T\mathbf{w}_n + b_{y'}) - (\mathbf{X}_{y_n}^T\mathbf{w}_n + b_{y_n}) + \Delta(y', y_n)\} \quad (5.15)$$

where $\Delta(y_1, y_2)$ is the cost of misclassifying class $y_1$ for $y_2$. In our context, we use $\Delta(y_i, y_j) = 1 - \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta. Multi-class SVM can be interpreted as a joint optimization of $N$ SVMs where the hyper-planes are trained to maximize the margin between each class and all the remaining classes [8]. For each observation $\mathbf{w}_n$ the scores are calculated as $\mathbf{X}\mathbf{w}_n + b$. Multi-class SVM classifiers are referred to as MSVM in this thesis.

### 5.2.3 LRE11 multi-class logistic regression

For evaluating various iVectors over the LRE11 evaluation set, we use another variant of MLR. An affine transform is used to convert the $M$-dimensional iVector $\mathbf{w}_n$, into a $K$-dimensional score-vector, $\mathbf{s}_n$ as follow:

$$\mathbf{s}_n = \mathbf{X}\mathbf{T}\mathbf{w}_n + \mathbf{b}, \quad (5.16)$$

where the $\mathbf{T}$ is an $M \times M$ matrix which does the within-class covariance normalization (WCCN) [31]. The WCCN squashes the within-class covariance matrix so that it becomes identity. This is necessary since in regularized

parameter estimation of LR, the dynamic range of the different iVector dimensions should have the same effect in the regularization term. $\mathbf{X}$ is estimated by minimizing the regularized objective function from [12]:

$$\mathbf{X}^*, b^* = \arg\min_{\mathbf{X},b}\{\frac{\lambda}{N}tr(\mathbf{X}^T\mathbf{X}) - \frac{1}{K\ln K}\sum_{k=1}^{K}\frac{1}{N_k}\sum_{n\in S_k}\ln\frac{\exp(s_{kn})}{\sum_{j=1}^{K}\exp(s_{jn})}\}, \tag{5.17}$$

where $N = \sum_k N_k$, $s_{kn}$ is the $k^{th}$ component of $\mathbf{s}_n$ and $S_k$ is a subset of training utterances corresponding to class $k$. The regularization weight, $\lambda$, is set to[12]:

$$\lambda = \left(\frac{1}{N}\sum_{n\in S}\sqrt{\mathbf{w}_n^T\mathbf{T}^T\mathbf{T}\mathbf{w}_n}\right)^2, \tag{5.18}$$

where $\mathcal{S} = \bigcup_k \mathcal{S}_k$ [12]. Note that (5.17) and (5.14) differ only in including the $\ln K$ term in the normalization of the loss function in (5.17), regularization coefficient value and use of the transformed iVectors as the input instead of the iVectors themselves.

### 5.2.4 Gaussian linear classifier

A Gaussian classifier is an alternative for modeling language in the space of the iVectors. This way, we train a single Gaussian for each language using an ML estimate of the Gaussian mean as:

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{w_n\in S_k}\mathbf{w}_n, \tag{5.19}$$

where $\boldsymbol{\mu}_k$ is the mean for language $k$ and $S_k$ is the subset of training iVectors corresponding to language $k$. The within-class covariance $\boldsymbol{\Sigma}_{wc}$ is shared among all the Gaussians and estimated using data from all the languages as:

$$\boldsymbol{\Sigma}_{wc} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}_n - \boldsymbol{\mu}_{\mathbf{w}_n})(\mathbf{w}_n - \boldsymbol{\mu}_{\mathbf{w}_n})^T, \tag{5.20}$$

where the $\boldsymbol{\mu}_{\mathbf{w}_n}$ is the mean of the language that $\mathbf{w}_n$ belongs to. The likelihood of an iVector w.r.t. each Gaussian is:

$$P(\mathbf{w}_n|C_k) = \mathcal{N}(\mathbf{w}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_{wc}^{-1}). \tag{5.21}$$

By expanding the logarithm of (5.21) we get

$$\log P(\mathbf{w}_n|C_k) = -\frac{1}{2}\mathbf{w}_n^T\boldsymbol{\Sigma}_{wc}^{-1}\mathbf{w}_n + \mathbf{w}_n^T\boldsymbol{\Sigma}_{wc}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_{wc}^{-1}\boldsymbol{\mu}_k + const. \tag{5.22}$$

Note that, since we use a shared covariance matrix for all language models, the first quadratic term is constant over all Gaussians for every iVector. If there is no need for calibrating the class likelihoods obtained using (5.22), we can directly calculate the posterior probabilities of languages using Bayes rule $p(C_k|\mathbf{w}_n) = \frac{p(\mathbf{w}_n|C_k)p(C_k)}{\sum_j p(\mathbf{w}_n|C_j)p(C_j)}$ where the quadratic term cancels out since it scales both the nominator and the denominator by the same amount. Therefore, we can simplify the likelihood calculation by omitting this term. However, it may be important to include the quadratic term in case the scores are used as features for the following calibrating classifier. In [46], the authors claimed that using the correct likelihood as an input to the calibration marginally outperforms the simplified likelihood calculation. However, we did not observe a significant difference in the final system performances. Note that by omitting the quadratic term in (5.22), the likelihood calculation changes to a linear transformation of the iVectors. As a result, we call this style of classifiers as Gaussian linear classifier (GLC).

## 5.3 Discussion

Each of the mentioned classifiers have advantages and disadvantages. In general discriminative classifiers (e.g. SVM and LR) are modeling class borders rather than class distributions which makes them less sensitive to the outlier iVectors. However, they are prone to get overtrained. On the other hand, generative language classifiers are less likely to get overtrained while they are sensitive to outlier iVectors that do not follow the assumed distribution. In fact, the use of the generative GLC assumes that the input (iVectors in our case) are drawn from Gaussian distribution. We will show more results on this in Section 7.2.

As we mentioned, one of the thesis objectives is producing Gaussian distributed iVectors, which simplifies the back-end of the LID system. In the case of the GLC back-end the shared within–class covariance matrix is

calculated over iVectors from all languages. We can calculate the shared within–class covariance matrix in advance over a reasonable amount of data from many languages and keep it fixed. This way, we can add a new language to the list of target languages by simply calculating the mean from iVectors corresponding to the new language. However, in case of the discriminative classifiers, we would need to train all the language models again after adding a new language to the list of target languages. In both cases, we need to retrain the calibration.

Another interesting question in this area is, whether to use a binary formulation of language classifiers or to use a multi-class formulation. Our experiments showed that, as long as we keep two layers of classifiers in our back-end (i.e. a first layer to generate class scores and a second layer for score calibration), the choice of binary or multi-class discriminative classifiers in the first layer does not have a significant effect on the LID performance [73].

## 5.4 Calibration and fusion

As we mentioned earlier in this chapter, we need a multi-class classifier as the second layer of classifiers that we call *calibration and fusion* in Figure 5.1 to produce calibrated class likelihoods for each utterance. The input to this module can be class likelihood ratio (in case of BLR in the first classification layer), class likelihoods (in case of MLR classifier in the first layer) or scores from other LID systems. There are two main reasons for the *calibration and fusion* module: obtaining likelihood scores calibrated on a development set that has data distribution close to the evaluation set and fusing outputs of multiple LID systems. The calibration and fusion can be done either jointly or separately. In case we do the fusion separately there will be a third multi-class classifier after generating calibrated class likelihoods. We use different configurations of calibration and fusion in this thesis. In the following, we explain our NIST LRE09 calibration and fusion that does calibration and fusion jointly and is used for most of the experiments in this thesis. We also explain our NIST LRE11 calibration and fusion that does calibration and fusion separately. The main reason for having two different calibration fusion plans is simply the availability of these plans as I was working on this thesis. For main part of the this thesis, we had only NIST LRE09 calibration and fusion. The NIST LRE11 calibration and fusion was developed for NIST LRE11 at BUT and after NIST LRE11, we were still interested in comparing some of the systems with previous experiments.

### 5.4.1 LRE09 calibration and fusion

Assume that there are $M$ separate LID systems each producing a $K$-dimensional score vector $\mathbf{s}_{in}$ and corresponding ancillary information ($d_{in}$) for each utterance $n$ and system $i$. In our experiments $K$ is the number of target languages. Note that the score vector does not necessarily has to have the same dimensionality as the number of the target languages. For example, we can train classifiers using other criteria or data that can be seen as side information providing a useful information about test utterances. The ancillary information $d_{in}$ represents duration-related information for each utterance. In the case of the acoustic front-end, it can be the number of the speech frames in each utterance and in the case of the phonotactic systems, it can be the expected number of phonemes in the corresponding utterance. For every front-end, three GLC classifiers, denoted by $B1$, $B2$ and $B3$, are trained over different normalizations of score vectors: the pure score vectors, the score vectors normalized by corresponding ancillary values and score vectors normalized by square root of corresponding ancillary values that are denoted as $B1(\mathbf{s}_{in})$, $B2(d_{in}^{-0.5})$ and $B3(d_{in}^{-1}\mathbf{s}_{in})$, respectively . Based on the output of these three GLC transforms and ancillary informations, we can write the output of the calibration fusion module as

$$\mathcal{I}_n = \sum_{i=1}^{M}(a_{1i}B1(\mathbf{s}_{in}) + a_{2i}B2(d_{in}^{-0.5}\mathbf{s}_{in}) + a_{3i}B3(d_{in}^{-1}\mathbf{s}_{in})) + \mathbf{b} + \mathbf{C}\vec{\gamma} \quad (5.23)$$

where $d_{in}$ is the ancillary information for the utterance $n$ from LID system $i$, $a_{ji}$ is the scalar weights for each GLC classifier, $\mathbf{b}$ is a $K$-dimensional offset vector, $\mathbf{C}$ is a square matrix corresponding to the dimensionality of $\vec{\gamma}$ and $\vec{\gamma}$ is a vector of concatenated ancillary information from different LID systems. The $d_{in}$ varies for different system. $B1()$, $B2()$, $B3()$ are calculated using (5.22) replacing $w_n$ with $s_{in}$, $d_{in}^{-0.5}$ and $s_{in}d_{in}^{-1}$, respectively. In our acoustic LID system, the number of voiced frames in each utterance is used as the ancillary information and for the phonotactic LID system, we set it to number of the expected phonemes (sum of all the 3-grams in each utterance) in each utterance. The parameters of the calibration fusion, $(a_{ji}, \mathbf{b}, \mathbf{C})$, are discriminatively trained to minimize multi-class cross-entropy (as in multi-class logistic regression) as implemented in [8],[58].

### 5.4.2 LRE11 back-end

The LRE11 calibration fusion plan is simpler than LRE09. We shall mention that this calibration fusion plan is only used for LRE11 and RATS

experiments. For each system, the output scores are calibrated using an affine transform trained on the development set:

$$\mathbf{r}_n = \mathbf{C}\mathbf{s}_n + \mathbf{d}, \tag{5.24}$$

where $\mathbf{C}$ is a full $K \times K$ matrix and $\mathbf{d}$ is a $K$-dimensional vector. Note that this calibration plan does not change the score vector dimensionality. The model parameters, $\mathbf{C}, \mathbf{d}$, are trained by same regularized logistic regression as explained in Section 5.2.3 without WCCN. After this step, for all the utterances where we failed to generate scores[1], an $\mathbf{r}_n = 0$ is inserted. In the next step, the calibrated scores are sent to the fusion. For each utterance $n$, the output of the fusion is:

$$\mathcal{I}_t = \sum_{i=1}^{M} \alpha_i \mathbf{r}_{ti} + \mathcal{B}. \tag{5.25}$$

where $\alpha_i$ is the fusion weight for the system $i$ and $\mathcal{B}$ is a $K$-dimensional bias vector. These parameters are also trained to minimize the cross-entropy objective function as explained in Section 5.2.3.

[1]This may happen when an utterance is too short or has no speech in it

# Chapter 6

# Data selection and preparation

In this chapter, we first explain the meaning of different measures which are used to express different aspects of LID system performance. In the next step, we give a full description of the data used for training of our front-ends and back-ends.

## 6.1 LID evaluation

We shall first specify what we mean by evaluation. The LID problem can be defined either as an identification or a verification task. In general we can define an LID task as:

"*Given a trial utterance and a set of target languages, determine what is its language identity.*"

The target list may contain only the known language (presented in the training data) or additionally more unknown (out-of-set) languages. The first task is regarded as closed-set LID and the latter is called open-set LID problem. In this thesis, we regards the LID problem as a verification task. This means that, for each verification trial, we should verify a hypothesis. The hypothesis for a verification task is: "test utterance $X$ belongs to a target language $Y$". For the verification task, we accept or reject the hypothesis. However, as we explained in Section 5.4, language scores are continuous values and to make a hard decision, we need to define a threshold based on which, we either accept or reject the hypothesis. This way, we have four cases:

I ) **Correct acceptance**: correctly accepting the hypothesis.

II ) **False acceptance**: wrongly accepting the hypothesis.

III ) **Correct rejection**: correctly rejecting the hypothesis.

IV ) **False rejection**: wrongly rejecting the hypothesis.

Among all four cases, *false acceptance* and *false rejection* are the system errors that are denoted as *false alarm* and *miss*, respectively for the rest of this thesis.

## 6.2 NIST evaluations

National Institute of Standards and Technology (NIST[1]) is the US federal technology agency that works with industry to develop and apply technology, measurements, and standards. NIST is the main organizer of the speaker and language recognition evaluations. NIST started language recognition evaluations for comparing system performances from all over the world by means of organized evaluations. For each evaluation, evaluation data is sent to the participants. In case no data is publicly available for a subset of target languages, development data is also provided by NIST. Normally, participants use the data from previous NIST LRE evaluations and any other available data sources to train their systems. We briefly summarize the earlier NIST evaluation campaigns. For LRE09 and LRE11, which we based our experiments on, more descriptive information is provided.

**NIST LRE1994**
The first NIST LRE took place in 1994. The evaluation data was taken from OGI multilingual telephony speech corpus[2]. The OGI corpus is monologue speech data.

**NIST LRE1996**[3]
The second NIST LRE was in 1996. The evaluation data was selected from different sources, mainly OGI multilingual telephony speech and Switchboard (wide-band and narrow-band). Starting with this evaluation, three standard duration conditions have been defined in all the subsequent NIST LRE evaluations: 30s, 10s and 3s. The parallel phoneme recognizer followed

[1] http://www.nist.gov/
[2] http://www.cslu.ogi.edu/corpora/mlts/
[3] http://www.itl.nist.gov/iad/mig//tests/lang/1996/LRE96EvalPlan.pdf

Table 6.1: NIST LRE 2003 target language list.

| | | | |
|---|---|---|---|
| Arabic (Egyptian) | German | Farsi | French (Canadian) |
| Hindi | Japanese | Korean | English (American) |
| Mandarin | Tamil | Vietnamese | Spanish (Latin American) |

Table 6.2: NIST LRE 2005 target language list.

| | | | | |
|---|---|---|---|---|
| English (American) | *English (Indian)* | Hindi | Japanese | Korean |
| Mandarin (Mainland) | *Mandarin (Taiwan)* | Spanish (Mexican) | Tamil | |

by language model (PPRLM) was the dominant approach used by LRE1996 participants.

**NIST LRE2003**

The NIST LRE 2003 was very similar to NIST LRE96 [44]. The evaluation data was selected from *CallFriend* Conversational Telephony Speech (CTS) for twelve languages as listed in Table 6.1. Similar to LRE1996, the PPRLM was the dominant approach used by the LRE2003 participants [70][65].

**NIST LRE2005**[4]

The NIST LRE 2005 was similar to previous NIST LRE evaluations in the sense that it was on CTS data. However, NIST started to include more challenges to push the LID community to improve their technologies. The task was defined on dialect and language recognition. There were seven languages for the primary condition[5] and dialects for English and Mandarin were added to the extended condition. Table 6.2 shows the LRE05 languages. In this evaluation, discriminative training of GMM-based acoustic systems was used and, for the first time, performance of the acoustic systems became comparable to the dominant PPRLM approach.

**NIST LRE2007**[6]

There was a significant increase in the number of target languages in this evaluation compared to NIST LRE05. The list of target languages consisted of 14 languages as listed in Table 6.3. The task was again defined on CTS

[4] http://www.itl.nist.gov/iad/mig//tests/lang/2005/LRE05EvalPlan-v5-2.pdf

[5] NIST often chooses to define a subset of evaluation trials as representing the primary conditions of interest in an evaluation. The rest of the trials are regarded as extended condition.

[6] http://www.itl.nist.gov/iad/mig//tests/lang/2007/LRE07EvalPlanv8b.pdf

Table 6.3: NIST LRE 2007 target language list.

| Test Lang. or Dial. Rec. | General LR | Chinese LR | Mandarin DR | English DR | Hindustani DR | Spanish DR |
|---|---|---|---|---|---|---|
| Arabic | ✓ | | | | | |
| Bengali | ✓ | | | | | |
| Farsi | ✓ | | | | | |
| German | ✓ | | | | | |
| Japanese | ✓ | | | | | |
| Korean | ✓ | | | | | |
| Russian | ✓ | | | | | |
| Tamil | ✓ | | | | | |
| Thai | ✓ | | | | | |
| Vietnamese | ✓ | | | | | |
| Chinese | ✓ | | | | | |
| Cantonese | | ✓ | | | | |
| Mandarin | | ✓ | | | | |
| Mainland | | | ✓ | | | |
| Taiwan | | | ✓ | | | |
| Min | | ✓ | | | | |
| Wu | | ✓ | | | | |
| English | ✓ | | | | | |
| American | | | | ✓ | | |
| Indian | | | | ✓ | | |
| Hindustani | ✓ | | | | | |
| Hindi | | | | | ✓ | |
| Urdu | | | | | ✓ | |
| Spanish | ✓ | | | | | |
| Caribbean | | | | | | ✓ |
| non-Caribbean | | | | | | ✓ |

data with three conventional duration conditions: 30s, 10s and 3s. The main focus of the task was dialect recognition and the task was defined as two language recognition tests along with 4 different dialect recognition tests.

In this evaluation, acoustic systems obtained the best performance. The best performance was achieved by combining channel compensation techniques (eigenchannel adaptation in the feature domain) and discriminative training of GMM [33].

### 6.2.1 NIST LRE2009

At the time of starting this thesis, the NIST LRE09[7] was the most recent NIST LRE. It was also the most complicated one. There are 23 languages

[7] http://www.itl.nist.gov/iad/mig//tests/lang/2009/LRE09_EvalPlan_v6.pdf

Table 6.4: NIST LRE 2009 target language list.

| | | | |
|---|---|---|---|
| Amharic | Bosnian | Cantonese | Creole(Haitian) |
| Croatian | Dari | English(American) | English(Indian) |
| Farsi | French | Georgian | Hausa |
| Hindi | Korean | Mandarin | Pashto |
| Portuguese | Russian | Spanish | Turkish |
| Ukrainian | Urdu | Vietnamese | |

in the LRE09 target language list as shown in Table 6.4. There are three main evaluation conditions: open-set, closed-set and pairwise as:

I ) **closed-set**: For each trial, the test utterance comes from one of the known target languages and is hypothesized to be from one of the known target languages. This was the mandatory condition for all the participants.

II ) **open-set**: For each trial, the test segment can also come from an additional set of *unknown* languages. The identities of the *unknown* languages were not disclosed at the time of the evaluation.

III ) **pairwise**: For each trial, the test utterance comes from one of two suggested languages.

There are three duration conditions as:

I ) **3** seconds nominal. 2-4 seconds actual.

II ) **10** seconds nominal. 7-13 seconds actual.

III ) **30** seconds nominal. 25-35 seconds actual.

The primary performance measure is $C_{avg}$ (see Section 6.4.1). The evaluation data contains both conversational telephone speech (CTS) and telephony speech that is broadcast through Voice of America (VOA) broadcast news. The latter involves people making phone calls to the broadcast studio. The evaluation data comprise at least 100 segments from each duration condition for the languages in the target language list. The complete Evaluation set contained 41793 utterances from all durations and all the languages including out-of-set languages. The number of evaluation segments for the closed-set condition is 31178.

Table 6.5: NIST LRE 2011 target language list.

| Language | Abbreviation |
|---|---|
| Arabic Iraqi | arir |
| Arabic Levantine | arle |
| Arabic MSA | arms |
| Arabic Maghrebi | arma |
| English Indian | engi |
| English American | enga |
| Russian | russ |
| Farsi | fars |
| Slovak | slvk |
| Hindi | hind |
| Spanish | span |
| Lao | laot |
| Tamil | tami |
| Bangali | bang |
| Mandarin | mand |
| Thai | thai |
| Czech | czec |
| Panjabi | pjbc |
| Turkish | turk |
| Dari | dari |
| Pashto | pash |
| Ukrainian | ukra |
| Polish | poli |
| Urdu | urdu |

### 6.2.2 NIST LRE2011

The NIST LRE 2011 evaluation differs from all the prior ones in emphasizing the language pair condition, which was introduced in LRE09. The target language list comprises 24 languages as listed in Table 6.5. Like LRE09 it contains both conversational telephone speech (CTS) and broadcast narrow-band speech (BNBS), generally involving people making phone calls to the broadcast studio. Multiple broadcast sources are included. For LRE09, we called it VOA since that was the only broadcast news source. As for the NIST LRE09, there are three duration conditions 30s, 10s and 3s.

Table 6.6: RATS target language list.

| Arabic Levantine | Dari | Farsi | Pashto | Urdu |
|---|---|---|---|---|

## 6.3 RATS

RATS stands for robust automatic transcription of speech and is a DARPA program which seeks to advance the state of the art in speech technology using audio from highly degraded communication channels. The goal of the RATS program is to create technology capable of accurately determining speech activity regions, detecting keywords, identifying language and speakers in highly degraded, weak and/or noisy communication channels. Here, we only talk about the language recognition task of RATS. There are five target languages in RAST LID as shown in Table 6.6 and 10 non-target languages in four duration conditions as 120s, 30s, 10s and 3s.

## 6.4 Evaluation metrics

There are many different ways to measure the performance of an LID system which take into accounts errors of a LID system i.e. *false alarm* and *miss*.

### 6.4.1 $C_{avg}$ average cost

This is the main metric that we use for this thesis and it was introduced for NIST LRE07. The main idea behind the $C_{avg}$ metric is to have an application dependent evaluation metric. This is because in different applications, we might be interested to check the system performance for different operating points. Each operating point is defined by a miss cost ($C_{miss}$), a false alarm cost ($C_{FA}$) and prior probability of target, non–target and out–of–set languages denoted as $P_{target}$, $P_{non-target}$ and $P_{out-of-set}$, respectively. The $C_{avg}$ for a given operating point is calculated as:

$$C_{avg} = \frac{1}{N_L} \sum_{L_T} \left\{ \begin{array}{c} C_{miss} \cdot P_{target} \cdot P_{miss}(L_T) \\ + \sum_{L_N} C_{FA} \cdot P_{non-target} \cdot P_{FA}(L_T, L_N) \\ + C_{FA} \cdot (1 - P_{out-of-set}) \cdot P_{FA}(L_T, L_O) \end{array} \right\}, \qquad (6.1)$$

where $N_L$ is the number of languages in the closed set of target languages, $L_T$ and $L_N$ denote target and non-target language respectively and $L_O$ is

denoting the out-of-set languages. $C_{miss}$ and $C_{FA}$ are application-dependent costs for miss or false alarm, respectively. The operating point for NIST LRE09 and RATS in terms of parameter values is defined as:

$$
\begin{aligned}
&P_{out-of-set} = 0 : \textit{closed set} \\
&P_{out-of-set} = 0.2 : \textit{open set} \\
&\\
&P_{Non-target} = (1 - P_{target} - P_{out-of-set})/(N_L - 1) \\
&\\
&C_{miss} = C_{FA} = 1, \\
&P_{target} = 0.5
\end{aligned}
\tag{6.2}
$$

We shall mention that $P_{out-of-set}$ is used in *open set* evaluation which is reported only for RATS in this thesis. In this thesis, we address the *closed set* condition for NIST LREs. $P_{Non-target}$ defines the prior probability for the test segments coming from other languages than the hypothesized *target* language. The input from each LID system to the $C_{avg}$ formula is the probability of *miss* for a target language $L_T$ denoted as $P_{miss}(L_T)$ and probability of *false-alarm* for each language pair $(L_T, L_N)$ denoted as $P_{FA}(L_T, L_N)$. These probabilities are calculated as:

$$
\begin{aligned}
P_{miss}(L_T) &= \frac{\#miss(L_T)}{\#Trial(L_T)}, \\
P_{FA}(L_T, L_N) &= \frac{\#FA(L_T, L_N)}{\#Trial(L_N)}.
\end{aligned}
\tag{6.3}
$$

where # denotes number of trials. The $FA(L_T, L_N)$ means that a trial belongs to $L_N$ according to the key but the LID system labeled it as $L_T$, $\#Trial(L_N)$ defines the number of non-target trials, $\#Trial(L_T)$ defines the number of trials for the target language $L_T$ and $\#miss(L_T)$ defines the number of *misses* for the target language $L_T$. We are always interested in $P_{miss}(L_T)$ and $P_{FA}(L_T, L_N)$ that give us a lower $C_{avg}$. Now let us explain how we calculate these parameters. To have the $P_{miss}(L_T)$ and the $P_{FA}(L_T, L_N)$, we first need to make a binary decision whether the trial belongs to the corresponding $L_T$ or not. Notice that at the time of making this binary decision, we do not know the correct label of the trial. In other words, rejecting that the trial belongs to the target language ($L_T$) means that it belongs to the other languages in list. Let us denote the set of other languages than the target language $L_T$ as $L_{Imposters}$. Assume that we have

language likelihood $P(X|L_T)$ as the output of the language classifier. Having the prior probabilities of the target and *other* languages defined by the application (For NIST tasks is defined as (6.2)), we make our decision based the value of the posterior probabilities $P(L_T|X)$ as:

$$\begin{aligned} &\text{If} \quad P(L_T|X) > 0.5 \;\Rightarrow\; X \;\text{ belongs to } L_T \\ &\text{If} \quad P(L_T|X) < 0.5 \;\Rightarrow\; X \;\text{ belongs to } L_{Imposters} \end{aligned} \tag{6.4}$$

where $P(L_T|X)$ is calculated using the Bayes rule

$$\begin{aligned} P(L_T|X) &= \frac{P(X|L_T)P(L_T)}{P(X)} \\ &= \frac{P(X|L_T)P(L_T)}{P(X|L_T)P(L_T) + (1 - P(X|L_T))(1 - P(L_T))} \end{aligned} \tag{6.5}$$

Based on this decision for all the hypotheses and after having correct language labels of all the trials, we can calculate $P_{miss}(L_T)$ and $P_{FA}(L_T)$.

### 6.4.2 Pair-wise system evaluation

This is the primary performance measure for the NIST LRE11[8]. Trials consist of a test segment along with a specified target language pair. The full set of trials consist of all combinations of an evaluation test utterance and a target language pair. Thus if $K$ is the number of target languages, each test segment will be used for K * (K-1) / 2 trials. For each language pair $(L_1, L_2)$, the miss probability for $L_1$ and $L_2$ is calculated. These probabilities are combined into a single number that represents the cost performance of a system for distinguishing the two languages, according to an application-dependent cost model as:

$$C(L_1, L_2) = C_{L1}P_{L1}P_{miss}(L1) + C_{L2}(1 - P_{L1})P_{miss}(L_2), \tag{6.6}$$

where $C_{L1}$, $C_{L2}$ and $P_{L1}$ are application-dependent parameters. Here $C_{L1}$ and $C_{L2}$ may be viewed as the costs of a miss for $L1$ and $L2$, respectively, and $P_{L1}$ as the prior probability for $L1$ with respect to this language pair. These parameters will be set to give equal cost and probability to each language:

$$\begin{aligned} &C_{L1} = C_{L2} = 1, \\ &P_{L1} = 0.5 \end{aligned} \tag{6.7}$$

[8] http://www.nist.gov/itl/iad/mig/upload/LRE11_EvalPlan_releasev1.pdf

For the NIST LRE11, systems are evaluated based on the 24 language pairs with the highest costs. To do so, 24 language pairs with highest pairwise cost for the 30$s$ duration are selected. The performance measure for each duration will be the mean of the pairwise cost for the corresponding 24 language pairs. This is denoted as Pair Error Rate (PER) in this thesis.

This performance measure is a good analytical tool for spotting defects of a LID system. However, since different systems may have different 24 worst pairs, it is not a useful performance criterion for system comparisons.

In this work, most of our system analyses are based on the NIST LRE09. We also report performance of our LID systems on NIST LRE11 evaluation set.

## 6.5 Development data collection and preparation

### 6.5.1 NIST LRE09

All the previous NIST LRE evaluations were based on CTS data. From LRE09, NIST utilized telephony bandwidth broadcast radio speech for most of the target languages as well. This includes telephone conversations that are broadcast through radio channels. This has two advantages: firstly, we should deal with channel variability which was not considered as a challenge for the LID problem during the previous LREs and secondly, it is much easier to collect the telephony data transmitted over radio channel. Two DVDs were provided by the NIST that includes the last 2 years archive of Voice Of America (VOA) from the languages in the target language list. The BUT group also downloaded other similar sources from the Internet. Table 6.7 shows sources from which the training and development data were taken.

We split the training data into two independent subsets, which we denote as TRAIN and DEV. The TRAIN set has 23 languages (according to the NIST LRE09) and has about 49190 segments in total that sum to 1572 hours of speech. The DEV set also has 23 languages and a total of 38135 segments. The DEV set is split into balanced subsets having nominal durations of 3s, 10s and 30s. It is mainly based on segments from the previous evaluations plus additional segments extracted from longer files from CTS and VOA databases (which were not contained in the TRAIN set). The detailed data selection for different languages from the two main channels (CTS and VOA) are shown in Table 6.8.

In the next step, we run the BUT speaker identification system over DEV and TRAIN sets to spot overlapping speakers in both sets. If found,

Table 6.7: Train & Development data source for NIST LRE 2009.

| | |
|---|---|
| CF | CallFriend |
| F | Fisher English Part 1.and 2. |
| F | Fisher Levantine Arabic |
| F | HKUST Mandarin |
| SRE | Mixer (data from NIST SRE 2004,2005,2006, 2008) |
| LDC07 | development data for NIST LRE 2007 |
| OGI | OGI-multilingual |
| OGI22 | OGI 22 languages |
| FAE | Foreigners Accented English |
| SpDat | SpeechDat-East |
| SB | SwitchBoard |
| VOA | Voice of America radio broadcast |

we removed data of such speakers from one of the sets so that the TRAIN and DEV sets contain data from disjoint sets of speakers. This way, we make sure that our LID system is learning classes based on language information, not speaker information.

As shown in Table 6.8 the languages in the LRE09 do not have the same amount of available data. Number and durations of the recordings for each language are different. To avoid the problem of unbalanced training data for the languages in the target list, we further limit the number of recordings for each language to maximum of 500 while keeping longer utterance rather than shorter utterances. Some languages may have less than 500 utterances. This gives us in total 9810 recordings that sum up to 359 hours of speech.

The TRAIN set is used for the front-end training and the DEV set is used for training of the calibration and fusion in the back-end. This means that we do not have a separate set for internal evaluation of our system after calibrating and tuning the performance on the DEV set. To avoid the problem of over-tuning to the DEV set, tuning on the DEV set is done using a jackknifing scheme. We do 5 outer iterations, where in each, we randomly partition the DEV data into 5 subsets balanced across all 23 languages. In each outer iteration, we run 5 inner iterations. In each inner iteration, one subset is held out as test data, while the other 4 are used for back-end training. The $C_{avg}$ is calculated over all 25 subsets and averaged over all. The same is done for the fusion and the calibration parameters. The averaged fusion/calibration weights were applied to the NIST LRE09 evaluation set [34].

Table 6.8: Data distribution of DEV sets for NIST LRE 2009.

| | CTS | | VOA | |
|---|---|---|---|---|
| Language | #files | #hours | #files | #hours |
| Amharic | 0 | 0 | 1724 | 77.7 |
| Bosnian | 0 | 0 | 268 | 7.0 |
| Cantonese | 482 | 6.9 | 34 | 2.1 |
| Creole_Haitian | 0 | 0 | 425 | 14.8 |
| Croatian | 0 | 0 | 150 | 5.3 |
| Dari | 0 | 0 | 2410 | 78.8 |
| English_Indian | 714 | 2.2 | 0 | 0 |
| Englisg_American | 10560 | 290.9 | 3963 | 132.5 |
| Farsi | 656 | 22.6 | 1673 | 70.6 |
| French | 403 | 21.8 | 3679 | 88.7 |
| Georgian | 0 | 0 | 100 | 4.7 |
| Hausa | 0 | 0 | 2599 | 74.4 |
| Hindi | 755 | 26.0 | 358 | 15.7 |
| Korean | 691 | 21.3 | 342 | 16.3 |
| Mandarin | 1321 | 64.8 | 1049 | 35.7 |
| Pashto | 0 | 0 | 6317 | 102.3 |
| Portuguese | 294 | 0.5 | 1069 | 48.7 |
| Russian | 643 | 8.4 | 3071 | 82.2 |
| Spanish | 1001 | 47.5 | 1623 | 67.6 |
| Turkish | 0 | 0 | 262 | 9.8 |
| Ukrainian | 0 | 0 | 105 | 3.0 |
| Urdu | 24 | 1.4 | 1242 | 67.2 |
| Vietnamese | 743 | 25.7 | 113 | 8.9 |
| SUM | 18287 | 540 | 32576 | 1004 |

### 6.5.2 NIST LRE11

For each of the new languages in NIST LRE11, 100 segments with 30$s$ length were provided by NIST through the Linguistic Data Consortium (LDC) at the time of the evaluation. They are all selected from BNBS and are human-audited and are marked as the first development set, D1. The longer files from which the D1 segments are selected are also provided in 16kHz format for possible use by participants (marked as D2). Similar to LRE09, we downloaded data from other available radio archives on the web. Sources from which LRE11 development data is selected are listed in Table 6.10. We split the data into three independent sets denoted as TRAIN, DEV and TEST. They all contain data from the 24 target languages. The TRAIN set has about 60000 segments, the DEV set has about 38000 segments and the TEST set has about 26000 segments in total. The DEV and TEST sets are split into balanced subsets having nominal durations of 3s, 10s and 30s.

The DEV set is based on LRE09 DEV set and contains data from the previous LRE evaluations up to and including LRE07. The data for the new languages include additional segments extracted from longer D2, CTS and BRBS (which were not contained in the TRAIN set). The TEST set mainly consists of the NIST LRE09 evaluation data, plus data for the new languages. In the case of LRE11, since we have an internal held out TEST set, we do not use the jackknifing as we did for LRE09.

Table 6.9 shows details on the numbers of segments and lengths of recordings in hours for TRAIN, DEV and TEST sets. The pairwise evaluation metric is explained in Section 6.4.2. The official NIST LRE11 evaluation metric is based on the 24 language pairs that has the worst PER. In fact, the system performance is represented in terms of just the 24 most difficult language pairs and does not reflect the overall performance of an LID system. This makes the comparison between systems from different sites hard since the worst pairs for each subsystem and each site may vary. For this reason, we provide our system performance in terms of PER over the worst 24 language pairs and PER over all language pairs.

### 6.5.3 RATS

The linguistic data consortium (LDC) provided the training and test data for the RATS participants. The annotated audio recordings were selected from the existing and newly collected data sources as follows:

1. Fisher conversational telephone speech (CTS) : Arabic Levantine

Table 6.9: Data distribution over TRAIN, DEV and TEST sets for NIST LRE 2011.

| | TRAIN | | DEV | | TEST | |
|---|---|---|---|---|---|---|
| Language | #files | #hours | #files | #hours | #files | #hours |
| Arabic_Iraqi | 476 | 17.01 | 579 | 2.63 | 585 | 2.84 |
| Arabic_Levantine | 3442 | 158.74 | 576 | 2.52 | 585 | 2.66 |
| Arabic_Moroccan | 212 | 6.83 | 399 | 1.77 | 438 | 2.05 |
| Arabic_MSA | 201 | 4.72 | 435 | 2.71 | 391 | 1.96 |
| Bengali bang | 4084 | 88.54 | 633 | 2.32 | 563 | 1.91 |
| Czech | 2279 | 19.49 | 1015 | 5.49 | 694 | 4.02 |
| Dari | 2410 | 78.83 | 579 | 1.79 | 1167 | 3.33 |
| English_Indian | 1444 | 4.72 | 1174 | 3.82 | 1167 | 3.85 |
| English_American | 14523 | 423.33 | 8170 | 24.81 | 2615 | 8.80 |
| Farsi | 2477 | 96.63 | 1718 | 5.53 | 1540 | 4.89 |
| Hindi | 1113 | 41.79 | 2222 | 6.68 | 1922 | 6.26 |
| Laotian | 147 | 3.68 | 357 | 1.82 | 336 | 1.82 |
| Mandarin | 2370 | 100.55 | 6281 | 18.43 | 2976 | 9.94 |
| Pashto | 6317 | 102.35 | 588 | 1.87 | 1185 | 3.52 |
| Panjabi | 160 | 4.36 | 360 | 2.17 | 348 | 2.10 |
| Polish | 2098 | 17.63 | 772 | 4.02 | 489 | 3.30 |
| Russian | 8792 | 122.45 | 3014 | 10.26 | 2247 | 7.68 |
| Slovakian | 1776 | 13.55 | 505 | 3.26 | 513 | 3.41 |
| Spanish | 2624 | 115.08 | 4784 | 14.44 | 1155 | 3.44 |
| Tamil | 623 | 19.59 | 900 | 2.49 | 888 | 2.67 |
| Thai | 267 | 7.52 | 943 | 3.05 | 595 | 2.08 |
| Turkish | 262 | 9.77 | 579 | 1.90 | 1182 | 3.43 |
| Ukrainian | 967 | 24.07 | 572 | 1.91 | 1535 | 4.65 |
| Urdu | 1266 | 68.65 | 1016 | 3.53 | 1133 | 3.41 |
| total | 60330 | 1549.91 | 38171 | 129.23 | 26249 | 94.00 |

Table 6.10: Data sources for TRAIN, DEV and TEST sets of NIST LRE 2011.

| | |
|---|---|
| CF | CallFriend |
| F | Fisher English Part 1. and 2. |
| F | Fisher Levantine Arabic |
| F | HKUST Mandarin |
| SRE | Mixer (data from NIST SRE 2004, 2005, 2006, and 2008) |
| LRE | Development and evaluation data from previous NIST LRE |
| OGI | OGI-multilingual |
| OGI22 | OGI 22 languages |
| FAE | Foreign Accented English |
| SpDat | SpeechDat-East[9] |
| SB | SwitchBoard |
| VOA | Voice of America radio broadcast |
| RFEL | Radio Free Europe broadcast |
| AR-IR | Iraqi Arabic Conv. Tel. Speech (LDC2006S45) |
| AR-MSA | 2003 NIST Rich Transc. Eval Data (LDC2007S10) |
| AR-MSA | Arabic Broadcast News Speech (LDC2006S46) |

2. CallFriend conversational telephone speech : Farsi

3. NIST LREs: Dari, Farsi, Pashto, Urdu and non target languages

4. RATS: Farsi, Urdu, Pashto, Levantine

All recordings were retransmitted through 8 different communication channels, labeled by the letters A through H. A *push-to-talk* (PTT) transmission protocol was used in all channels except G. PTT states produce some regions where two or more non-transmission (NT) segments may occur. In addition to the speech (S) and non-speech (NS) regions, these NT regions are supposed to be marked in the annotations.

There are four conditions with durations 120s, 30s, 10s and 3s. Only recordings from the 120s condition were released for training and development. We therefore had to construct our own development samples for the shorter durations from the 120s audio files, based on BUT's voice activity detection (VAD).

During the development period, LDC delivered three incremental data releases for training and test. Only the first two releases are used for the main training and development sets that we report here. The whole data is split into TRAIN and DEV sets. We try to keep longer files in the TRAIN

Table 6.11: Distribution of DEV2 set for RATS language evaluation.

| Language | All files | 120s Trials | 30s Trials | 10s Trials | 3s Trials |
|---|---|---|---|---|---|
| Arabic | 1085 | 307 | 277 | 289 | 212 |
| Dari | 184 | 59 | 50 | 44 | 31 |
| Farsi | 947 | 291 | 259 | 238 | 159 |
| Pashto | 1032 | 309 | 280 | 259 | 184 |
| Urdu | 980 | 275 | 261 | 263 | 181 |

set. So, recordings with more than 60s of speech (according to the VAD output), are mainly included in the TRAIN set. This results in results in a TRAIN set containing 30774 files, that is not balanced among five target languages (e.g. we have 668 files for Dari, 12778 for Arabic Levantine). In the next step, we balanced the TRAIN set to have approximately 700 recordings for each target language.

The DEV set comprises 2432 recordings from non-target languages and 1000 recordings for each target language (Except Dari). These files are chosen in a way to have 600-900 files for each channel/language and 7120 files for each duration.

As we mentioned in Section 6.3, the key for RATS evaluation data is not disclosed to the participants. However, LDC released a development set called DEV2 that was sent through similar channels as the evaluation data. All the results that we report in this thesis are on the DEV2 set. Since the DEV2 set is not the official RATS evaluation set, some statistics about distribution of the recordings and numbers of the recordings are given in Table 6.11.

# Chapter 7

# Experiments

Any new proposal should be accompanied and confirmed by the experimental results. In this part, we first present results of our baseline system and state-of-the-art results at the time of starting this thesis. In the next step, we experiment with tuning the parameters of the proposed multinomial subspace and subspace n-gram models. All the tunings are done using NIST LRE09 as explained in Section 6.2.1. All the experiments use 3-gram statistics.

We shall initially mention in the beginning that all the parameter tunings and training of calibration and fusion are done over the DEV sets and EVAL sets are considered as unseen new data. This way, we simulate the real evaluation conditions where we do not have any information about the EVAL sets at the time of system training and tuning. We provide results on EVAL sets to show how our systems would perform if they had been used for the corresponding evaluations. It is obvious that the results on the DEV sets should be considered as cheating experiments that give us an impression of system performance upper bound. It is also a good way of discovering accidentally good results on EVAL set in some situations.

We should also mention that all the calibration and fusion plans that are used in this thesis are implemented by our colleagues at BUT and mainly by Niko Brummer from AGNITiO. Part of the NIST LRE09 calibration and fusion is publicly available[1]. The rest of what is reported in this section as phonotactic LID systems is developed by the author of this thesis. Training of the subspace and iVector extraction for continuous features (that comprises acoustic and prosodic features) is done using codes from our BUT colleagues.

[1] https://sites.google.com/site/nikobrummer/

Let us first describe pre-processing of the iVectors before giving the details on language models training.

## 7.1 iVector pre-processing

All the experiments presented in this chapter use mean removal and length normalization of the iVectors unless otherwise stated. We use only 3-gram statistics in the form of soft counts as explained in Section 4.7. We shall mention that the term iVector refers to all the feature vectors used as an input to the language models. For the mean removal, during the train phase, the iVectors' mean over $N$ iVectors in the train set is calculated. This mean is subtracted from all the iVectors in the train, development and evaluation sets. The intuition behind the mean removal is that the data sources for train, development and evaluation sets might be different and shifted with respect to each other.

The length normalization is done as:

$$\bar{\mathbf{w}}_n \leftarrow \frac{\mathbf{w}_n}{\sqrt{\sum_i (w_n^i)^2}}, \tag{7.1}$$

where $w_n^i$ is the $i^{th}$ dimension of the $r$-dimensional iVector $\mathbf{w}_n$.

## 7.2 Establishing the baseline

At the time of starting this thesis, I was inspired by the LID solution proposed in [17]. In [17] each utterance is decoded by a phoneme recognizer and soft count 3-grams are extracted from phoneme lattices as explained in Section 4.7. 3-gram soft counts extracted for each utterance is then put into a fixed length vector. These vectors of 3-gram statistics are then directly modeled by binary SVM classifiers to generate language scores using binary classifiers as defined in Section5.1.1. The fact that we can replace the n-gram language model in the back-end with discriminative classifiers and the simplicity of the approach was my main interest.

As our baseline system, we replicated the system in [17] using the 3-gram statistics obtained from the reduced phoneme set of BUT HU phoneme lattices as described in Section 4.7. This results in 35937 3-grams. The system is denoted as FULL-BSVM. On the other hand, results in [50] showed promising advantages of reducing feature dimensionality using PCA feature extraction. We were curious to compare the performance of the reduced feature vector to the baseline with full feature space so we replicated

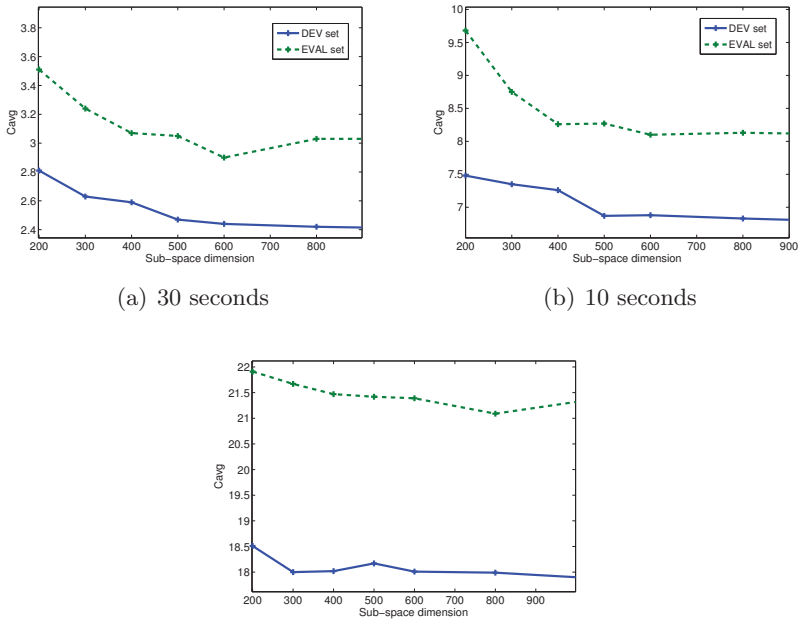
Table 7.1: The $C_{avg} \times 100$ for baseline systems using full and PCA-transformed 3-gram statistics from BUT HU with BSVM and BLR on DEV and EVAL sets over all the conditions of NIST LRE09 EVAL set.

| | | DEV | | | EVAL | | |
|---|---|---|---|---|---|---|---|
| System | Dimension | 30s | 10s | 3s | 30s | 10s | 3s |
| FULL-BSVM | 35937 | 2.56 | 6.75 | 17.64 | 3.08 | 8.5 | **20.93** |
| **PCA-BSVM** | 35937→400 | 2.91 | 7.25 | 18.05 | 3.81 | 9.09 | 21.39 |
| **PCA-BSVM** | 35937→600 | 2.83 | 7.05 | 17.77 | 3.62 | 8.89 | 21.09 |
| **PCA-BSVM** | 35937→1000 | 2.83 | 6.98 | 17.74 | 3.60 | 8.82 | 21.00 |
| PCA-BLR | 35937→600 | **2.22** | **6.22** | **17.26** | **2.93** | **8.29** | 22.60 |
| PCA-GLC | 35937→600 | 2.81 | 8.25 | 19.83 | 3.5 | 9.88 | 22.88 |

PCA-based feature extraction as explained in [50]. After length normalization and mean removal, the PCA-transformed features are sent to the binary support vector machines and binary logistic regression classifiers. The corresponding systems are denoted as PCA-BSVM and PCA-BLR, respectively. The PCA transformation is calculated over the square root of the n-grams as suggested in [50]. Both the BSVM and BLR classifiers are trained in a one-vs-all configuration. For the classifiers, we used LibSVM[2] and LIBLinear[3] publicly available packages. Table 7.1 compares performance of the FULL-BSVM with PCA feature extraction on NIST LRE09 task. It also shows the effect of using BLR instead of BSVM as the language classifiers. For all systems, scores from the classifiers are calibrated using LRE09 back-end as explained in Section 5.4.1.

Performances of the $400, 600$ and 1000 dimensional PCA-transformed features are shown in Table 7.1. Similar to what was reported in [50], we do not obtain notable improvement for dimensionalities higher than 600. We shall mention that the PCA-BSVM system was the state-of-art phonotactic LID result at the time of starting this thesis and other better phonotactic LID results that are reported in this chapter are contributions of this thesis. In the first step, we obtained notable improvement by using binary logistic regression instead of binary SVM classifiers. This finding is confirmed by using binary logistic regression for phonotactic iVectors as will be presented in Table 7.3. Even though the Gaussian linear classifier is a multi-class

[2]www.csie.ntu.edu.tw/ cjlin/libsvm/
[3]www.csie.ntu.edu.tw/ cjlin/liblinear/




(a) 30 seconds

(b) 10 seconds

(c) 3 seconds

Figure 7.1: The $C_{avg} \times 100$ on DEV and LRE09 EVAL set for different subspace dimensions over 30s, 10s and 3s conditions for SMM using BUT HU 3-gram statistics.

classifier, we report it here to show that Gaussian linear classifier does not perform well on PCA transformed n-gram statistics. The system is denoted as PCA-GLC. The main reason is that PCA-transformed features are not Gaussian distributed as explained in Section 4.1.1.

## 7.3 Subspace multinomial model system tuning

The theoretical part of extracting phonotactic iVectors using subspace multinomial model(SMM) was presented in Section 4.5. There are two unanswered questions for the SMM. What is a proper number of iterations in the parameter estimation and more importantly, what is an optimal phonotactic iVectors dimensionality. We make the decision on the number of iterations based on log likelihood change in each iteration.

Figure 4.7 shows that there is no notable change in the log likelihood after the $4^{th}$ iteration. Based on this, we choose 5 iterations of the parameter

estimation.

In the case of the iVector dimensionality, the final system performance on the DEV set in terms of $C_{avg}$ is used. Basically, higher iVector dimension may provide us with more discriminative information. However, extracting high-dimensional iVectors has a high memory overhead and is also computationally expensive. Furthermore, we would need more training data for higher-dimensional hyperparameter ($\mathbf{T}$) estimation since it may easily get overfit to the training data.

Figure 7.1 shows change in the $C_{avg}$ with respect to the subspace dimension for iVectors extracted using the BUT HU. The system settings are explained in detail in Section 7.4 as SMM-BLR. The blue lines show the $C_{avg}$ for different duration conditions over the DEV set of NIST LRE09. For the 30s and 10s conditions, no significant increase in the performance is observed for dimensions higher than 600. In the case of 3s, we observe some noise for dimensionalities higher than 300 but considering the total trend, 600 dimensions is a reasonable choice. We also plot the performance over the NIST LRE09 Evaluation set. We see that our conclusion about the subspace dimension generalizes to the Evaluation set. However, we can see performance degradation for dimensionalities higher than 600. We believe it is due to the hyper parameter, $\mathbf{T}$, overfitting.

## 7.4 SMM versus PCA-transformed feature

So far, we have the phonotactic iVectors generated using SMM. As we mentioned in Section 7.1, we use normalized iVectors. Normalizing iVectors is a common technique that improves performance in speaker identification using acoustic iVectors [27]. Table 7.2 shows the effect of each normalization on the overall performance for phonotactic iVectors using BLR. The MR stands for mean removal and LN denotes length normalization. As we expected, mean removal improves the results significantly since the sources of the training and evaluation data are different. In the next step, LN slightly improves the performance but in the case of within class covariance normalization (WCCN), the improvement is not notable. For rest of the experiments in this chapter, we use only MR and LN unless otherwise stated.

It is time to compare performances of our baseline systems with our proposed phonotactic iVector. Table 7.3 compares FULL-BSVM with PCA feature extraction and SMM. It also shows performances of PCA and SMM using BSVM and BLR classifiers. Table 7.3 confirms that on the NIST LRE09 EVAL set, BLR is outperforming BSVM on the iVector features

Table 7.2: $C_{avg} \times 100$ for different iVector normalization for BLR on BUT HU phonotactic iVectors on NIST LRE09 EVAL set.

| System | Normalization | 30s | 10s | 3s |
|---|---|---|---|---|
| SMM-BLR | - | 6.53 | 15.95 | 29.52 |
| SMM-BLR | MR | 2.93 | 8.70 | 21.75 |
| SMM-BLR | MR+LN | **2.81** | 8.33 | 21.39 |
| SMM-BLR | MR+LN+WCCN | 2.83 | **8.09** | **21.34** |

Table 7.3: The $C_{avg} \times 100$ for PCA-transformed 3-grams and phonotactic iVectors form BUT HU with BSVM and BLR classifiers on DEV and EVAL sets over all the conditions of NIST LRE09 EVAL set.

| | | DEV | | | EVAL | | |
|---|---|---|---|---|---|---|---|
| System | Dimension | 30s | 10s | 3s | 30s | 10s | 3s |
| FULL-BSVM | 35937 | 2.56 | 6.75 | 17.64 | 3.08 | 8.4 | **21.06** |
| PCA-BSVM | 35937→600 | 2.83 | 7.05 | 17.77 | 3.62 | 8.89 | 21.09 |
| PCA-BLR | 35937→600 | **2.22** | **6.22** | **17.26** | **2.93** | 8.29 | 22.60 |
| SMM-BSVM | 35937→600 | 2.52 | 7.38 | 18.41 | 3.25 | 9.47 | 23.59 |
| SMM-BLR | 35937→600 | 2.44 | 6.88 | 18.01 | 3.05 | **8.10** | 21.39 |
| SMM-BLR + PCA-BLR | Fusion | 2.05 | 5.74 | 16.71 | 2.79 | 7.63 | 21.05 |

as well. We will mainly be using BLR for the rest of the experiments. It is interesting to note that the SMM-BLR is performing better than the FULL-BSVM in 10$s$ and 30$s$ conditions of the EVAL set. However, it is not performing as well on the DEV set.

The last row in Table 7.3 shows results of score level fusion of PCA-BLR and SMM-BLR systems according to LRE09 fusion plan explained in Section 5.4.1. Even though the performances of these two systems are similar on this evaluation set, their score level fusion shows notable improvement which shows that these system are exploiting different information in the n-gram statistics.

## 7.5 Multi-class versus binary classifiers

The LID task is a multi-class classification task. It is interesting to know how multi-class formulations of our classifiers perform on the LID task. For

Table 7.4: $C_{avg} \times 100$ for different multi-class and binary classifiers on BUT HU phonotactic iVectors on NIST LRE09 EVAL set.

| System | 30s | 10s | 3s |
|---|---|---|---|
| SMM-BLR | **2.81** | 8.33 | 21.39 |
| SMM-BSVM | 3.07 | 8.55 | 21.68 |
| SMM-MLR | 3.16 | 8.66 | 21.82 |
| SMM-MSVM | 3.89 | 10.60 | 23.92 |
| SMM-GLC | 2.92 | **8.03** | **21.13** |

this purpose, we use iVectors generated with SMM to train multi-class SVM, multi-class logistic regression (MLR) and GLC as explained in Section 5.2.2, 5.2.1 and 5.3. Results are shown in Table 7.4

The results show that multi-class formulation of our SVM and logistic regression does not give us better performance on this task compared to a binary formulation of the corresponding classifiers. The GLC has the best results on 10s and 3s conditions and it performs reasonably well on 30s condition as well. This is in contrast to training GLC on PCA-transformed n-gram statistics reported in Table 7.1. We believe this is due to the distribution of the iVectors extracted with SMM model and we expect these iVectors to be more Gaussian distributed than the PCA-transformed n-gram statistics. More details on this is given in Section 7.8.

## 7.6 iVector fusion and score level fusion

In principal, our phonotactic iVectors can be fused in two different way: feature level fusion and score level fusion. The iVectors feature level fusion means that we concatenate iVectors from two systems in one feature vector. For example, 600-dimensional iVectors from BUT Hungarian phoneme recognizer (BUT-HU) and 400-dimensional iVectors from BUT RU form a 1000-dimensional feature vector. The choice of 400-dimensional iVectors from BUT RU was just due to the memory and computational overhead of 600-dimensional iVectors. The concatenated iVectors are then treated as one system and we train BLR language classifiers on the resulting 1000-dimensional iVectors. This way, the language classifier sees information in the two iVectors together. In the case of score fusion, two different sets of BLR classifiers are trained for each of the iVectors extracted from BUT HU and BUT RU n-gram statistics and the produced scores of each

Table 7.5: $C_{avg} \times 100$ for different phonotactic iVectors and their feature and score level fusions on NIST LRE09 EVAL set.

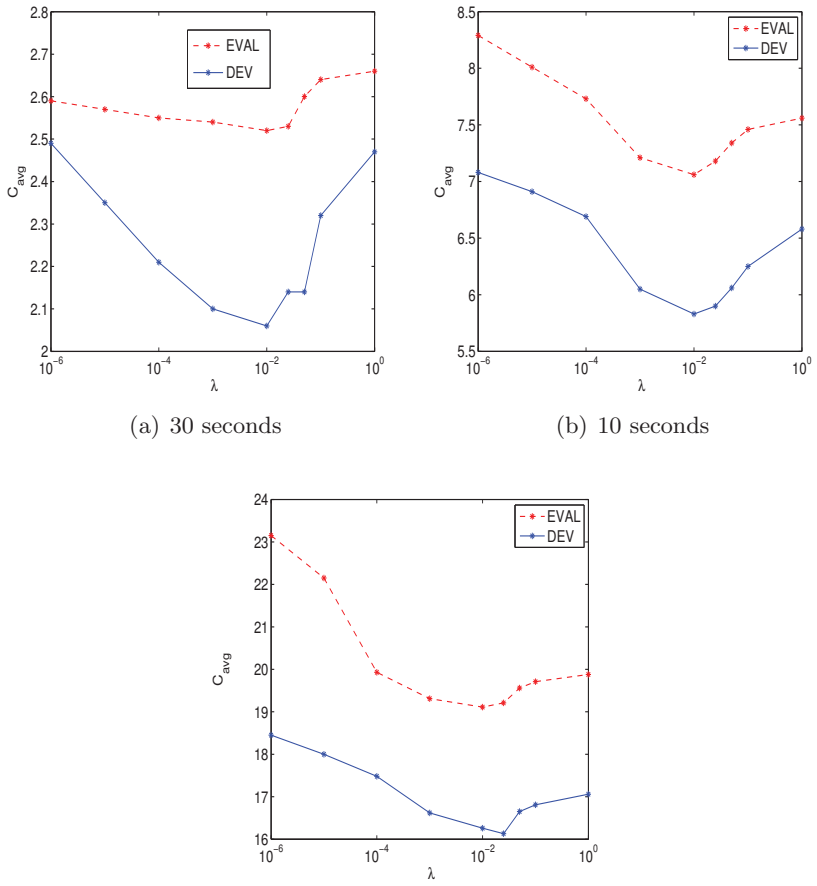
| System | Feature | Dimension | 30s | 10s | 3s |
|---|---|---|---|---|---|
| SMM-BLR | HU | 600 | 3.05 | 8.10 | 21.05 |
| SMM-BLR | RU | 400 | 2.59 | 7.42 | 19.83 |
| SMM-BLR | iVec fusion HU & RU | 1000 | 2.10 | 5.80 | 17.79 |
| SMM-BLR | Score fusion HU+RU | 600,400 | **2.09** | **5.34** | **16.53** |

set of classifiers goes into the LRE09 back-end as explained in Section 5.4.1. Table 7.5 shows performances of the individual SMM-BLR systems using HU and RU phoneme recognizers along with results of the corresponding score fusion and iVector fusion. The results show that we can achieve significant improvements using either of the fusion styles. For longer segments, the score level fusion gives the better performance on this task.

## 7.7 Regularized subspace n-gram model parameter tuning

In the case of the regularized SnGM, we use the same number of iterations for the parameter estimation since no significant change in the objective function in (4.35) was observed after the $4^{th}$ iteration. The next step is to define the regularization coefficient $\lambda$. Basically, there is no analytical way to define the value of $\lambda$ and it is normally set experimentally. Here, we tune it with respect to the $C_{avg}$ on DEV set. In Figure 7.2, the performances of the phonotactic iVectors LID system for the DEV set of NIST LRE09 and different duration conditions are depicted in blue lines. The best performance is obtained with $\lambda = 0.01$. We also plot the $C_{avg}$ on the evaluation set of the NIST LRE09 EVAL set in red lines. We see again that the best LID performance is obtained with $\lambda = 0.01$ over different duration conditions. This means that the tuning of the $\lambda$ over the DEV set generalizes very well to an independent evaluation set.

Optimizing the objective function in (4.35) with an $L2$ regularizer can be seen as obtaining a maximum a-posteriori point estimate of the model parameters $\mathbf{T}$ and $\mathbf{w}$ with Gaussian priors imposed on these parameters.




(a) 30 seconds

(b) 10 seconds

(c) 3 seconds

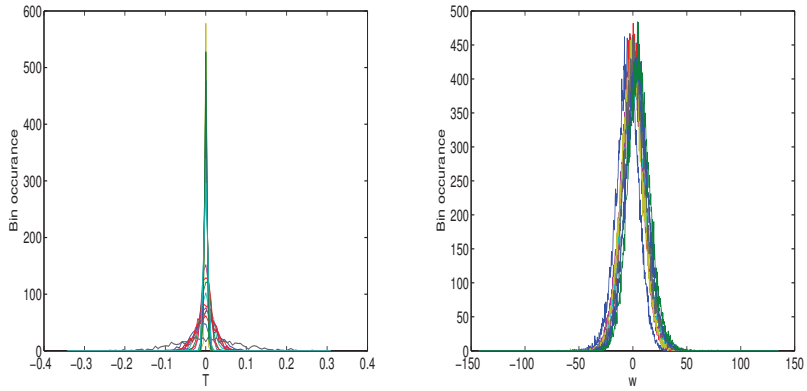
Figure 7.2: Tuning of $\lambda$ for RSnGM using BUT HU. $C_{avg} \times 100$ for SnGM-BLR over DEV and EVL set for 30s, 10s and 3s conditions on NIST LRE09 EVAL set.

## 7.8 Regularized subspace n-gram model

The regularized subspace model over clustered n-gram statistics (3-gram in our case) is our final proposal for phonotactic iVectors (see Section 4.5.7). The first step toward this model is to use a separate multinomial distribution for each 3-gram history as explained in Section 4.5.6. This model allows us to extract iVectors by maximizing the log likelihood of the phoneme labels conditioned on their corresponding histories according to the n-gram assumption. In order to estimate the model parameter and then extract iVectors, we need counts of 3-grams. However, there are many 3-gram histories that are rarely occurring in the TRAIN set. For some histories, there are no observation of the corresponding 3-grams in the TRAIN set. In the traditional n-gram language model, we use different smoothing techniques to assign a small probability to the unseen 3-grams. This is necessary for calculating the likelihood of a phoneme sequence. We have the same problem in SnGM as well. In case we have no occurrences of a 3-gram in the training data, the ML training of SnGM would assume zero probability for such a trigram which would result in pushing the weights ($\mathbf{w}$) and the model parameter ($\mathbf{T}$) to produce large negative log likelihood that corresponds to those zero probabilities. This can be seen as model overfitting to those rare 3-grams. In fact this problem is more serious in SnGM than SMM. In the SMM model all the multinomial probabilities are assumed to be the outcomes of a single multinomial distribution. This means that many probabilities are competing for the probability mass and this acts as a natural regularizer that limits the dynamic range of updates for $\mathbf{T}$ and $\mathbf{w}$. In case of SnGM, much fewer n-gram probabilities modeled in each multinomial distribution are competing for the probability mass. As a result, the SnGM focuses on these rare n-gram probabilities in a n-gram history and tends to introduce large values in $\mathbf{T}$ and $\mathbf{w}$ in order to be able to produce the corresponding close-to-zero probabilities. This is also why we found regularization more effective in the case of SnGM as will be seen in Table 7.6. In this thesis, as the first step, we use pruning of the 3-gram statistics to deal with the problem of the low-frequent 3-grams. We prune the 3-grams according to the following rules.

I ) Filter out all the 3-grams that have occurred less than 10 times throughout the whole TRAIN set.

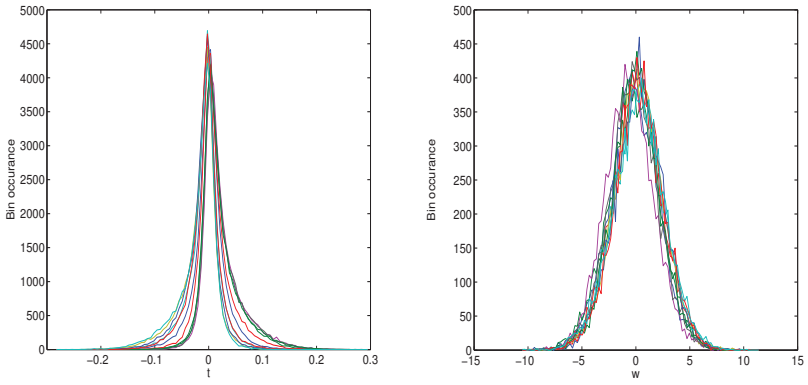
II ) If there are less than 5 3-grams with the same history, remove all 3-grams with this history.




(a) Distribution of values in 30 random rows of **T** in SnGM

(b) Distribution of values in 10 random dimensions of **w** in SnGM

Figure 7.3: Distribution of the values in dimensions of iVector and **T** rows for SnGM using BUT HU 3-grams.




(a) Distribution of values in 10 random rows of **T** in RSnGM.

(b) Distribution of values in 10 random dimensions of **w** in RSnGM.

Figure 7.4: Distribution of the values in dimensions of iVector and **T** rows for RSnGM using BUT HU 3-grams.

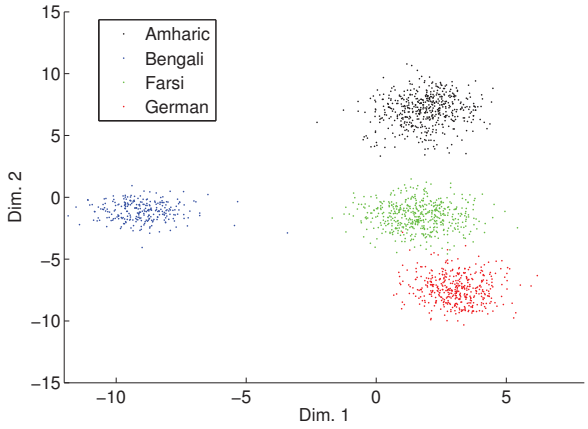



Figure 7.5: Distribution of iVectors for first four NIST LRE09 languages extracted with RSnGM using BUT HU 3-grams statistics. iVectors are projected into 2-dimensional space using LDA.

Note that the threshold 10 for the frequency of the 3-grams is just an ad hoc number. In fact, we did not observe any significant system degradation for using a frequency threshold of 20. Using the second rule, we also filter histories that contain less that 5 3-grams after the first filter. We noticed that such histories are also very low-frequent. By filtering low-frequent 3-grams, we also reduce the dimensionality of the 3-grams statistics which alleviates computational and memory overhead of the model parameter estimation and iVector extraction. We shall mention that the filtering on 3-grams is applied both at the time of model parameter estimation and iVector extraction. In other words, the filtered 3-grams are excluded from 3-gram statistics of all utterances in all sets.

Now, let us compare performance of our regularized subspace nGram model with our previous phonotactic iVector models. Table 7.6 shows the state-of-the-art results obtained from the subspace multinomial model (denoted as SMM) and the subspace n-gram model (denoted as SnGM). The results are given for BLR and GLC language classifiers. We can see that we get better result with SnGM compared to SMM in case of long utterances. However, results degrade for the shorter conditions. To explore this behavior, let us look at the values in model parameter $\mathbf{T}$ and $\mathbf{w}$. Figure 7.3 shows the distribution of the values for randomly chosen dimensions of iVectors over the TRAIN set and 10 random rows of the $\mathbf{T}$ matrix. As Figure 7.3

shows, the distribution of the values in the **T** matrix is very noisy. We can also see large numbers for the **w** values. This implies that the model is still trying to estimate the distribution of those low-frequent 3-grams by means of weights (**w**) and basis (**T**). In other words model is overfitting to those low-frequent 3-grams. Remember that the SnGM model is trying to model utterance-specific n-gram probabilities. This means that during iVector extraction and for the short utterances, we would have many low-frequent 3-grams. This pushes the weights (**w**) to produce large negative log likelihoods that corresponds to those low-frequent 3-grams. In Section 4.5.7, we effectively addressed the problem of model overfitting by means of regularized model parameter estimation and iVector extraction using the filtered 3-grams statistics. Now, let us look at the distribution of values for the subspace matrix **T** and weights **w** for the regularized SnGM (RSnGM). In Figure 7.4, a histogram of 10 random dimensions of **w** over TRAIN set and a histogram of 10 random rows of the matrix **T** are depicted. It is clear that L2 regularization of both **T** and **w** has smoothed the distribution of values in both **T** and **w**. It can be seen from Figure 7.4 that in case of **w**, the values are close to Gaussian distributed which confirms assumption of the Gaussian priors over **w** vectors. On the other hand, in the case of **T** rows, values seem to be Laplace distributed. This is mainly because the subspace matrix **T** is expanding the iVector space to the sparse original space of n-gram log-probabilities. Intuitively, this suggests the use of an $L1$ regularizer that corresponds to the assumption of Laplace prior for the **T** matrix. This is not done in this thesis and is suggested as a future work in Section 8.2. The RSnGM can effectively deal with the overfitting problem and we do get significant improvement over all conditions compared to the other systems. We observe the same behavior by using GLC classifiers as well. This is the best LID performance that is obtained for NIST LRE09 task using BUT HU.

The distributions of the iVectors are depicted in Figure 7.5 where the 600-dimensional iVectors from RSnGM are reduced to 2 dimensions using LDA. Comparing Figure 7.5 with Figure 4.2 shows that iVectors are more Gaussian-like distributed compared to PCA-transformed features. Note that we used iVectors corresponding to the same first four languages in the NIST LRE09 target language list to demonstrate the effect of our phonotactic iVector extraction compared to PCA-transformed features. Distributions of iVectors for all 23 languages in the NIST LRE09 target language list are shown in Figure 7.6.

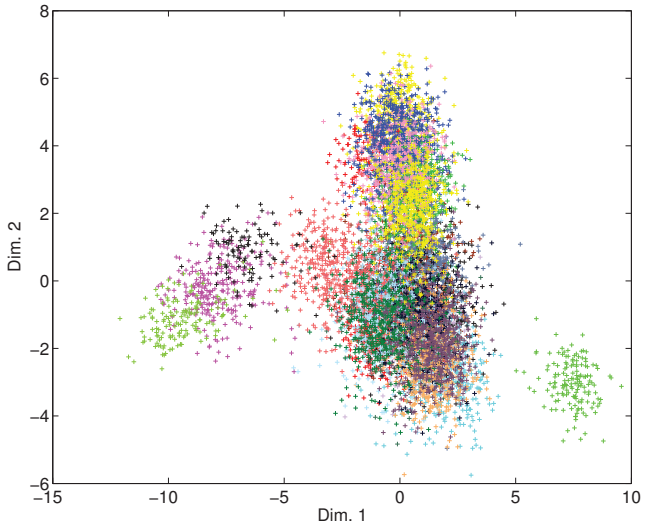



Figure 7.6: Distribution of iVectors for all 23 NIST LRE09 languages extracted with RSnGM using BUT HU 3-grams statistics. iVectors are projected into 2-dimensional space using LDA.

Table 7.6: $C_{avg} \times 100$ for different iVector feature extractions using BUT HU over all conditions of NIST LRE09 EVAL set.

| System | Reg. Coef. | 30s | 10s | 3s |
|---|---|---|---|---|
| SMM-BLR | - | 2.81 | 8.33 | 21.39 |
| SMM-GLC | - | 2.92 | 8.03 | 21.13 |
| SnGM-BLR | - | 2.68 | 8.63 | 23.15 |
| SnGM-GLC | - | 2.94 | 8.79 | 21.98 |
| **RSnGM-BLR** | 0.01 | **2.52** | **7.06** | **19.11** |
| RSnGM-GLC | 0.01 | 2.68 | 7.46 | 19.45 |

Table 7.7: $C_{avg} \times 100$ for different acoustic iVectors dimension using GLC on NIST LRE09 Evaluation task over 30s, 10s and 3s conditions. Taken from [46]

| | Dim. | | | | | |
|---|---|---|---|---|---|---|
| Condition | 200 | 300 | 400 | 500 | 600 | 700 |
| 3s | 16.29 | 15.87 | 15.63 | 15.50 | 15.29 | **15.25** |
| 10s | 5.55 | 5.25 | 5.11 | 4.90 | **4.76** | 4.79 |
| 30s | 2.36 | 2.08 | **1.88** | 1.90 | **1.88** | 1.93 |

## 7.9 iVectors for acoustic continuous features

As we mentioned before, acoustic iVectors based on continuous features is not the main focus of this thesis. Nevertheless, for comparison and fusion of different systems, we trained the acoustic iVector extractor based on [29] and [46]. We trained 2048 component GMMs with diagonal and full covariance matrix as our UBM. Our recipe for acoustic iVector extraction is the same as the one used in [46] where the effect of the iVectors' dimensionality on system performance in terms of $C_{avg}$ was studied. Their results are shown in Table 7.7. It shows that higher iVector dimensionalities are generally better. We can observe that as we increase the dimensionality, improvement in performance is not consistent for all the conditions. However, the best dimensionality on NIST LRE09 task is 600. On the other hand, training the iVectors extractor for a dimensionality higher than 400 has high memory and computational overhead. Since the performance loss for dimensionality of 400 is not significant, we chose dimensionality 400 for our iVectors. In accordance with [29], the iVector extractor is trained using 10 iterations of EM algorithm. Performances of our 400 dimensional iVectors using GLC, BLR and MLR classifiers are shown in Table 7.8. Systems are denoted as iVEC-GLC, iVEC-BLR and iVEC-MLR, respectively. For the MLR, we used the same language classifiers as we used for the NIST LRE11 which was explained in Section 7.12. The comparison shows that the GLC classifiers are performing better than BLR and MLR. We also tried training our GMM-UBM with full covariance to see how it affects the LID performance. The results show that using the full covariance matrix does not give a performance improvement over using a diagonal covariance

Table 7.8: Effect BLR, MLR and GLC classifier over acoustic iVector using diagonal and full covariance matrix in terms of $C_{avg} \times 100$ on NIST LRE09 Evaluation task over 30s, 10s and 3s conditions.

| System | Cov. | 30s | 10s | 3s |
|---|---|---|---|---|
| iVEC-GLC | Diag | 1.88 | **4.39** | **14.2** |
| iVEC-GLC | Full | 1.92 | 4.5 | 14.36 |
| iVEC-BLR | Diag | 2.08 | 4.61 | 14.56 |
| iVEC-MLR | Diag | **1.83** | 4.5 | 14.36 |

matrix. It is also much slower to train the UBM with full covariance matrix.

## 7.10 iVectors for prosodic continuous features

The extraction and classification of the prosodic iVectors is very similar to the acoustic iVectors. They only differ in the extraction of the frame-by-frame continuous prosodic features. The first step is extraction of the prosodic features (i.e. F0 and energy contour in our case). We use the Snack toolkit[4] for this purpose. It uses the RAPT algorithm that was explained earlier in Section 3.2. The pitch and energy values are then converted to the log domain to simulate human perception. We also normalize the energy values by subtracting the maximum energy in the corresponding utterance in the log scale. This gives us more robustness against non-speech effects like channel variation. In the SID problem, we do not normalize the log F0 values since they contain information about the speakers. However, in the case of LID, we are interested only in the language information and therefore this normalization at the utterance level helps. We normalize the log F0 value by removing the mean over each utterance [45].

In [39], the authors showed that the best performance for SID is obtained by defining the syllable boundaries based on the output of an LVCSR system. However, running an LVCSR system for this purpose is expensive. It also requires to know the identity of the language which, in fact, is our task. The fixed-length segmentation was the second best choice in [39] and we chose it. In [45], the author showed that 200ms (20 frames) segments with

[4] http://www.speech.kth.se/snack

Table 7.9: $C_{avg} \times 100$ for prosodic systems on NIST LRE09 Evaluation task over 30s, 10s and 3s conditions.

| | DEV | | | EVAL | | |
|---|---|---|---|---|---|---|
| System | 30s | 10s | 3s | 30s | 10s | 3s |
| PRS-iVEC-GLC | 9.25 | 16.11 | 25.41 | 16.98 | 23.73 | 32.95 |

50ms (5 frames) shift is a better choice for the LID problem. The authors also showed that fixed segmentation of speech gives a better performance compared to defining segment boundaries based on local minima of energy contours. This is mainly because we generate more feature vectors and we can provide more information to the GMM model. This helps particularly for the languages with small amounts of training data. In this thesis, we use a fixed length segmentation of 200ms (20 frames) with a shift of 50ms (5 frames) for the consecutive segments.

In Section 3, we introduced F0 and energy contours as prosodic features. Other prosodic features may also be considered. One can e.g. model formant contours with the same curve fitting approach. That is not explored in this work, though. In [20] it was proposed to include the number of voiced frames in each segment as an extra dimension in the final feature vector. In this work, we use a feature vector of length 13 comprising the first 6 Legendre polynomial coefficients for F0 and energy contours plus the number of voiced frames in corresponding segment.

The TRAIN, DEV and EVAL data setup is the same as NIST LRE09 data setup that was explained in Section 6.2.1. A UBM-GMM model is trained over the 13-dimensional feature vectors. In [45], performance of the different UBM sizes and iVector dimensionalities is studied. Based on [45], we trained a 200-dimensional iVector extractor based on the sufficient statistics extracted by a 2048 component GMM with the diagonal covariance matrix. Results are given in Table 7.9. The results are consistent with those reported in [45]. As we expected, prosodic LID system based on prosodic iVectors performs worse than phonotactic and acoustic iVectors (see Table 7.8 and Table 7.6). Nevertheless, we are interested to see whether it improves the LID performance in the system fusion.

## 7.11 System fusion

As we mentioned before, the main point of developing different LID systems with different front-end is to exploit complementary informations that might be presented in the different front-ends. Studying LID system fusions, gives us useful clues on which system to focus on and how system fusion can improve the overall LID system performance.

The experiments are conducted on LRE09 fusion explained in Section 5.4.1. We use 200-dimensional prosodic iVectors as in Section 7.10 denoted by PRS-GLC, 400-dimensional acoustic iVectors as in Section 7.9 denoted by iVEC-GLC and phonotactic iVectors for BUT HU as in Section 7.8 denoted by RSnGM-HU-GLC. In LRE09 fusion, the vectors of scores (vectors of class log likelihoods in case of the GLC. See Section 5.4.1) from individual classifiers are the input to the fusion module. To make sure that all reported improvements are the result of having the different front-ends (not the language classifiers), we use the the same language classifiers for all the front-ends. Table 7.10 shows performances of each stand-alone system along with all the possible system fusion combinations. The first conclusion is that fusion of any two systems out of three is improving the overall LID performance. The highest gain is obtained with fusion of the acoustic iVectors and phonotactic iVectors. In the next step, we fused all the three systems. Even though, fusion of the prosodic iVectors with iVectors extracted by either of the other two iVector models improves the LID performance, the fusion of all three systems is only slightly better than the fusion of the acoustic and phonotactic iVectors and only for 3s condition.

For the next experiment, we use the best language classifier for each iVector system and we do the same fusions. According to the results in Tables 7.9, 7.8 and 7.6, GLC is the best choice for the acoustic and prosodic features and BLR is the best for the phonotactic iVectors. Table 7.11 lists performances of each stand alone system along with all possible systems fusions.

Comparing Table 7.11 and Table 7.10 shows that not only the phonotactic system with BLR gives us better result than GLC classifier but also its fusion with the prosodic system gives us slightly better improvement. Fusion of all the three systems give us just slightly better results than the fusion of the acoustic and phonotactic systems on 10s conditions. As we can see from Table 7.11 and 7.10, most of the gain comes from fusing phonotactic and acoustic iVectors. Another conclusion is that even though BLR language classifiers gives us the best performance for RSnGM system, it has a marginal effect while fusing all three systems. We can see that fusion of

Table 7.10: $C_{avg} \times 100$ for different system fusions of different LID systems with the GLC classifiers on NIST LRE09 Evaluation task over 30s, 10s and 3s conditions.

| System | 30s | 10s | 3s |
|---|---|---|---|
| PRS-GLC | 16.98 | 23.73 | 32.95 |
| RSnGM-HU-GLC | 2.68 | 7.46 | 19.45 |
| iVEC-GLC | 1.88 | 4.39 | 14.2 |
| PRS-GLC + iVEC-GLC | 1.89 | 4.08 | 13.16 |
| RSnGM-HU-GLC + iVEC-GLC | 1.56 | 3.29 | 11.75 |
| PRS-GLC + RSnGM-HU-GLC | 2.52 | 6.21 | 17.80 |
| PRS-GLC + RSnGM-HU-GLC + iVEC-GLC | 1.56 | 3.26 | 11.32 |

Table 7.11: $C_{avg} \times 100$ for different system fusions of different LID systems with the best classifiers on NIST LRE09 Evaluation task over 30s, 10s and 3s conditions.

| System | 30s | 10s | 3s |
|---|---|---|---|
| PRS-iVEC-GLC | 16.98 | 23.73 | 32.95 |
| RSnGM-HU-BLR | 2.52 | 7.06 | 19.11 |
| iVEC-GLC | 1.88 | 4.39 | 14.2 |
| PRS-iVEC-GLC + iVEC-GLC | 1.89 | 4.08 | 13.16 |
| RSnGM-HU-LR + iVEC-GLC | 1.51 | 3.24 | 11.71 |
| PRS-iVEC-GLC + RSnGM-HU-BLR | 2.44 | 6.05 | 17.52 |
| PRS-iVEC-GLC + RSnGM-BHU-LR + iVEC-GLC | 1.57 | 3.16 | 11.34 |

Table 7.12: $C_{avg} \times 100$ for different phonotactic and acoustic iVector fusions on NIST LRE09 Evaluation task over 30s, 10s and 3s conditions.

| System | 30s | 10s | 3s |
|---|---|---|---|
| SMM-HU-BLR | 2.81 | 8.33 | 21.39 |
| SnGM-HU-BLR | 2.68 | 8.63 | 23.15 |
| RSnGM-HU-BLR | **2.52** | **7.06** | **19.11** |
| iVEC-GLC | 1.88 | 4.39 | 14.2 |
| SMM-HU-BLR + iVEC-GLC | 1.60 | 3.46 | 12.24 |
| SnGM-HU-BLR + iVEC-GLC | 1.54 | 3.45 | 12.46 |
| RSnGM-HU-BLR + iVEC-GLC | **1.51** | **3.24** | **11.71** |

all three systems in Table 7.11 and Table 7.10 is giving almost the same results.

An interesting experiment is to study the fusion of the different phonotactic iVectors with acoustic iVectors. Table 7.12 shows results of fusing our three phonotactic iVectors with acoustic iVectors. Results show that the gradual improvement of the LID performance, coming from the different phonotactic iVector extractions, is reflected in the system fusions as well. In Table 7.6, we showed that SnGM performs better than SMM on the 30s condition and it degrades for short conditions. We see the same behavior in the system fusion as well. The fusion of SnGM and iVEC gives better performance than the fusion of SMM and iVEC on 30s conditions. However, it degrades for shorter conditions. After all, we can see that RSnGM gives us the best fusion result as it does as the stand-alone phonotactic iVector system.

As a conclusion, we can see again that the acoustic and phonotactic iVectors have the highest effect on the overall system performance and after fusion of these two, the prosodic system improves the performance just marginally. During the time of writing this thesis, we came to the same conclusion for other tasks (Robust Automatic Speech Transcription (RATS) [48] and NIST LRE11 [12]) as well.

## 7.12 NIST LRE 2011

In this part, we report our system performances on NIST LRE11 evaluation set. This comprises some of the systems that were part of the AGNITIO–BUT–CRIM (ABC276) [12] submission to the NIST LRE11 evaluation along with some complementary experiments.

### 7.12.1 Phonotactic systems

The NIST LRE11 is the most recent NIST LRE. As we mentioned in Section 6.2.2, the evaluation metrics of NIST LRE11 is PER and is different from the metrics used in all the other experiments. However, for any future work, we considered it appropriate to compare the performance of our systems in NIST LRE11. We built four phonotactic systems: phonotactic iVectors (SnGM) and PCA-based feature extraction using BUT HU and BUT RU 3-gram statistics. We shall mention that our results of the worst 24 language pairs are as evaluated by NIST at the time of the NIST LRE11 evaluation and since the RSnGM was proposed after the NIST LRE11, we are not reporting results of RSnGM on the NIST LRE11.

The NIST LRE11 evaluation metric is an average of PER over the worst 24 language pairs (see Section 6.4.2). This makes the comparison task difficult since the worst 24 pairs may vary for different systems from different labs. To make the comparison between systems more reasonable, we provide PER for the worst 24 and for all language pairs. We shall mention that results on all language pairs are scored at BUT. NIST did not provide results on all language pairs at the time of the NIST LRE11 evaluation.

Table 7.13 shows the result for the worst 24 language pairs and all language pairs over the NIST LRE11 evaluation set for phonotactic systems [12]. All the phonotactic iVectors are centered to the TRAIN set by removing the mean and length normalizing before scoring as explained in Section 7.1. Table 7.13 shows that on the NIST worst 24 language pairs, the SnGM model and the PCA-transformed features are performing similarly. The SnGM model is slightly better particularly on BUT RU. Nevertheless, on all language pairs, the SMM model is outperforming the PCA-transformed features. The improvement is more notable in the case of BUT RU.

### 7.12.2 System fusion and submission

Table 7.14 shows results of different system fusions for phonotactic iVectors using BUT HU phoneme recognizer, PCA-based features using BUT RU

Table 7.13: PER% results for worst 24 and all language pairs over NIST LRE11 EVAL set.

| | | NIST24 | | | ALL | | |
|---|---|---|---|---|---|---|---|
| System | Dim. | 30s | 10s | 3s | 30s | 10s | 3s |
| PCA-HU | 1000 | 16.12 | 25.03 | 35.99 | 4.16 | 10.34 | 23.92 |
| PCA-RU | 1000 | 14.69 | 24.16 | 34.96 | 3.99 | 10.34 | 22.74 |
| SnGM-HU | 600 | 16.45 | 24.96 | 35.89 | 3.91 | 10.09 | 23.90 |
| SnGM-RU | 600 | **14.67** | **22.83** | **34.06** | **3.13** | **8.26** | **21.91** |

Table 7.14: Fusion results in PER% for worst 24 and all language pairs over NIST LRE11 EVAL set.

| | | NIST24 | | | ALL | | |
|---|---|---|---|---|---|---|---|
| System | Dim. | 30s | 10s | 3s | 30s | 10s | 3s |
| SMM-HU+PCA-RU | 1000, 1000 | 12.2 | 19.2 | 30.86 | 2.61 | 6.92 | 18.62 |
| SMM-HU+iVEC | 600, 600 | 8.82 | 15.88 | 26.91 | 1.59 | 4.26 | 13.44 |
| PCA-RU+iVEC | 1000, 600 | 8.69 | 15.18 | **25.53** | 1.47 | 4.07 | 13.17 |
| SMM-HU+PCA-RU+iVEC | 600, 1000, 600 | **8.47** | **14.84** | 25.68 | **1.46** | **3.81** | **12.67** |

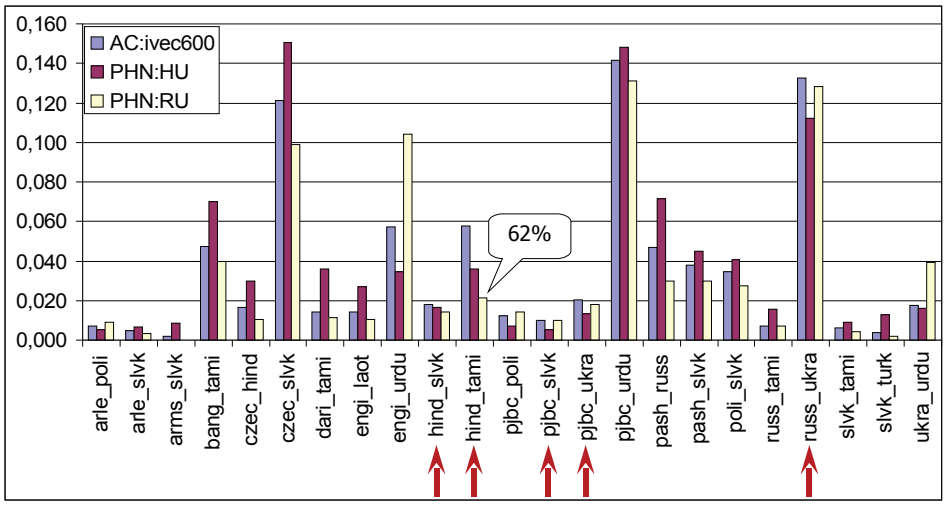



Figure 7.7: Analysis of the phonotactic iVectors from BUT HU 3-grams (red), PCA-features from BUT RU 3-grams(yellow) and acoustic iVectors (blue) systems over the NIST LRE11 worst 24 language pairs condition by means of PER. See Table 6.5 for abbreviations.

phoneme recognizer and acoustic iVectors. All the results are obtained with score fusion as explained before.

The results show that for both the NIST worst 24 language pairs and all language pairs, all systems are carrying complementary information and are useful to be included in the fusion. The improvement is more significant when acoustic iVectors are fused with phonotactic systems.

During the last few years, especially since the proposal of iVectors, acoustic LID has benefited from different subspace modeling solutions proposed in SID and acoustic LID based on the iVector model became the state-of-the-art LID solution. During ICASSP 2011 in Prague, Douglas Reynolds raised a question: why are we still working on phonotactic LID? The same question was asked during the NIST LRE11 workshop by George Doddington. Those questions made us to make a finer result analysis to see how effective the phonotactic systems are. Figure 7.7 shows performance of the three systems separately over the worst 24 language pairs.

We shall mention that for these particular experiments, we can not conclude superiority of phonotactic iVectors over PCA or the other way around since they are using different features. The findings from Figure 7.7 are as follows:

I ) PCA-RU provides better scores for 19 language pairs compared to

iVEC.

II ) SnGM-HU provides better scores for 9 language pairs compared to iVEC.

III ) For 5 language pairs, shown in red arrows, both PCA-RU and SnGM-HU are better than acoustic iVectors, iVEC.

The results are difficult to interpret. A first comparison with the sizes of data (see Table 6.8) would suggest that phonotactic systems outperform the acoustic system in cases with little training data. e.g. the SnGM model using BUT RU 3-grams produces 62% smaller PER for the (Hindi,Tamil) language pair than the acoustic iVector system. This, however, is not valid for the (Ukrainian,Russian) pair with abundant data. We might also suspect the correctness of data labeling. For example, Ukrainian is very similar to Russian for east Ukrainian speakers and the performance might depend heavily on the region in which the speakers were sampled. Similar cases are likely to occur in other language pairs. Also, we cannot rule out over-training of the acoustic system on a particular transmission channel (especially for Indian and Pakistani languages): in this case, phonotactic systems should provide better performance.

## 7.13 RATS

RATS is the most recent and challenging language recognition evaluation that addresses language recognition in highly degraded, weak and/or noisy communication channels. We shall mention that the real evaluation data for RATS are not disclosed to the participants. However, the DEV2 set which was delivered to participants, was already transmitted through a similar channels as the real evaluation data by LDC. Here, we provide results of our phonotactic iVectors models and PCA-based feature extraction on the DEV2 set of the RATS language evaluation to show the performance of our systems in noisy channel conditions. Note that, due to the adverse channel conditions, we may observe different behavior from our systems than those we reported on NIST LREs. We shall mention that we are not describing the RATS evaluation in detail in this thesis. Nevertheless, we briefly explain our systems and evaluation conditions that helps the reader to compare results of the phonotactic and acoustic iVectors with those described for NIST LREs. We refer the interested readers to the corresponding references.

Since we are dealing with different kind of channels and noise in RATS, a new VAD is trained to model noisy and non-speech effects of the channels

(e.g. non-transmission noise in walkie talkie) that does not exist in NIST LREs data. The development of the VAD is fully explained in [57]. All the data are sent through this VAD before any other data processing. Only the speech parts of the recordings are used by the following LID systems.

The phonotactic systems are trained using two phoneme recognizers: a Czech (CZ) phoneme recognizer was trained in the same way as explained in Section 4.2 with the difference that 30% of the phoneme recognizer training data was corrupted artificially with a noise at 10dB. An Arabic phoneme recognizer was trained on Levantine (LE) Arabic data labeled by LDC. The LE Arabic phoneme recognizer is trained in the same way as explained in Section 4.2. Nevertheless, unlike the CZ phoneme recognizer that is trained on artificially corrupted data, the training data for LE Arabic phoneme recognizer was sent through similar transmission channels as the evaluation data. In other words, the LE Arabic phoneme recognizer is trained using more realistic noisy data. We get $3-$gram statistics from the CZ and LE Arabic phoneme recognizers in the same way as described in Section 4.7. The PCA-based feature extraction, SnGM and RSnGM are developed as described in in Section 4.4, Section 4.5.6 and Section 4.5.7, respectively.

In order to study system performances in fusion as well as the stand-alone systems, we use acoustic iVectors that were generated at the time of the RATS evaluation. To generate the acoustic iVectors, the audio files are processed with a Wiener filter from the Qualcom-ICSI-OGI Aurora front-end [14]. The acoustic system uses the SDC feature extraction. After discarding silence frames, every 10ms speech frame is mapped to a 56-dimensional feature vector. The feature vector is the concatenation of an SDC-7-1-3-7 vector and 7 MFCC coefficients (including C0). Cepstral mean and variance normalization, as well as RASTA filtering are applied before SDC. The recipe used to generate the acoustic iVectors is the same as we described in Section 2.

All systems are scored using multi-class logistic regression (MLR) as explained in Section 5.2.3. The generated scores are calibrated and fused using the DEV set based on the calibration and fusion plan explained in Section 5.4.2.

Table 7.15 shows results of phonotactic systems (PCA, SnGM, RSnGM) along with the acoustic iVector system (IVEC). As we mentioned before, the CZ phoneme recognizer is trained on artificially corrupted data. Using the CZ phoneme recognizer, the PCA system performs better than both of SnGM and RSnGM on the long conditions (120s and 30s). The PCA system is even better than IVEC on 120s condition. The RSnGM model performs better on short conditions (10s and 3s). After all, we can see that, except

Table 7.15: $C_{avg} \times 100$ for PCA-based, SnGM, RSnGM and acoustic iVector and their fusions on RATS DEV2 language recognition task over 120s, 30s, 10s and 3s conditions.

| System | Dimension | Phn. Rec. | 120s | 30s | 10s | 3s |
|---|---|---|---|---|---|---|
| PCA | 1000 | CZ | **8.36** | **15.75** | 21.20 | 28.27 |
| SnGM | 600 | CZ | 10.19 | 16.66 | 22.13 | 30.70 |
| RSnGM | 600 | CZ | 9.92 | 15.76 | **20.55** | **27.52** |
| PCA | 1000 | LE | 6.45 | 12.76 | 19.70 | 30.04 |
| SnGM | 600 | LE | 6.67 | 12.69 | 20.30 | 29.19 |
| RSnGM | 600 | LE | **5.86** | **12.27** | **16.41** | **26.45** |
| IVEC | 600 | - | 9.00 | 13.71 | 18.32 | 23.43 |
| PCA+IVEC | 1000 | CZ, - | **7.25** | **11.64** | **15.56** | 21.53 |
| SnGM+IVEC | 600 | CZ, - | 8.20 | 12.06 | 16.12 | 20.77 |
| RSnGM+IVEC | 600 | CZ, - | 7.71 | 11.78 | 15.70 | **20.72** |
| PCA+IVEC | 1000 | LE, - | **6.37** | **10.29** | **14.16** | 21.98 |
| SnGM+IVEC | 600 | LE, - | 7.84 | 10.53 | 14.90 | 21.40 |
| RSnGM+IVEC | 600 | LE, - | 6.78 | 10.35 | 14.83 | **20.90** |

for the 120s condition, the performance of the PCA and RSnGM are very similar.

In case of the LE phoneme recognizer, the RSnGM is consistently outperforming PCA in all conditions. The SnGM is also better than PCA for 30s and 3s conditions. We believe that we can trust results obtained using the LE phoneme recognizer better than the CZ phoneme recognizer since it is trained on more realistic noisy data. In general, all systems based on LE phoneme recognizer are performing significantly better than those based on CZ phoneme recognizer.

The acoustic iVectors (denoted as IVEC) is known as the state-of-the-art stand-alone system. It has the main advantage of performing well even on the short conditions. Based on our results in Table 7.15, the IVEC system performs better than all the phonotactic systems based on CZ phoneme recognizer. However, in case of the LE phoneme recognizer, the RSnGM performs consistently better than IVEC system over all conditions except 3s condition. The improvement is more significant for longer conditions as we get more than 40% relative improvement for $C_{avg}$ on 120s condition.

The last two sections of Table 7.15 shows the results for the fusion of IVEC with all the phonotactic systems. In case of the phonotactic systems based on the CZ phoneme recognizer, the PCA system fuses better than RSnGM as it was performing marginally better as the stand-alone system compared to RSnGM. In case of the LE phoneme recognizer, however, results are surprising! We see that, as stand-alone systems, the RSnGM performs better than IVEC on almost all conditions and SnGM performs better than IVEC on 120s and 30s conditions. Fusion of IVEC and RSnGM degrades the results of stand-alone RSnGM on 120s condition. This is because we do duration independent calibration. On the other conditions, the fusion helps. Even though RSnGM performs significantly better than PCA as the stand-alone system, fusion with the PCA system gives us better performance than the fusion with RSnGM.

# Chapter 8

# Conclusion

During the last decade, solutions for LID have evolved a lot. Before subspace modeling of the GMM means was introduced, phonotactic approaches were the state-of-the-art solutions for LID problem. The subspace modeling has revolutionized SID and LID for the last five years and has become the state-of-the-art modeling technique in the SID and LID community. The main goal for this thesis was to transfer the idea of subspace modeling from GMM to n-gram model. To do so, we proposed a subspace modeling framework for discrete n-gram probabilities so that we can represent the utterance-dependent n-gram probabilities with a low-dimensional phonotactic iVector. This low-dimensional representation of the utterance-dependent n-gram probabilities can help us to achieve a better modeling of intersession and intra-session variability which can improve performance of a LID system.

## 8.1 Conclusion and summary

We started this thesis by describing subspace modeling of the GMM means. We referred to this model as the iVector model for continuous features since it deals with parameters of the continuous distribution (GMM means). We explained advantages of the GMM means subspace modeling compared to the previous language specific GMM training and GMM adaption techniques. Following that and based on the published recipes, we explained training of our iVector extractor for continuous features in Section 2.4. This model is then used to extract acoustic iVectors using SDC features for acoustic LID. Using the extracted acoustic iVectors, we could achieve similar state-of-the-art acoustic LID performance to reported results for the same NIST tasks (see Section7.9).

In Chapter 3, modeling of the prosodic features for LID was described. Estimation of the pitch and energy contours by means of Legendre polynomials were explained. This way, contours of the prosodic features are represented in terms of weights of a fixed number of Legendre polynomials. This can be also seen as a dimensionality reduction of the prosodic features through which variable-length prosodic features are represented by fixed number of coefficients for Legendre polynomials. This fixed length vector of coefficients is then used as an input feature for the iVector model for continuous features as explained in Section 2.4. Using the extracted prosodic iVectors, we achieved state-of-the-art results published for the prosodic iVectors (see Section 7.10).

As it was claimed in the objectives of this thesis, enhancing state-of-the-art phonotactic LID and proposing subspace modeling techniques inspired by the iVector model for continuous features formed our main inspiration to do this thesis. In Section 4.1.1, we described why a latent channel factor interferes with the assumption of the GMM training and how the iVector model for continuous features addresses this problem by separate modeling of inter-session and intra-session variabilities. In the next step, we explained how a similar problem in the language specific n-gram language models can be addressed by proposing iVector model for discrete features. Relation between n-gram probabilities and a multinomial distribution is discussed in Section 4.5.1 and then our proposed subspace multinomial model (SMM) for extracting iVector from discrete features is described in Section 4.5.2. In the next step, we enhanced the SMM to be consistent with the n-gram language model assumption and we proposed the subspace n-gram model for extraction of the phonotactic iVectors in Section 4.5.6. Finally, a regularized subspace n-gram model for robust phonotactic iVector extraction is proposed in Section 4.5.6.

In order to compare the performance of our iVector extractor model for discrete features with the previous state-of-the-art phonotactic and also other iVector based LID systems (i.e. acoustic and prosodic iVectors based on iVectors for continuous features), we set up our baselines based on the state-of-the-art phonotactic LID system reported in [17] using full n-gram statistics (referred to as FULL) and PCA-transformed n-gram statistics (referred to as PCA) as explained in Section 4.4. We could replicate the performance as was reported in the literature (see Section 7.4).

We can improve the performance of an LID system by enhancing the front-ends or the back-ends. Many different LID system comparisons with different settings are given in Chapter 7. We showed that by using the same back-end, phonotactic iVectors extracted using SMM slightly outperforms

the PCA and the FULL system on NIST LRE09 task. Our experiments showed that the use of logistic regression instead of support vector machines for generating language scores prior to the calibration and fusion of the scores, gives us a notable LID performance improvement (see Section 7.4). We also showed the effect of the feature vector normalization of the feature vectors produced by PCA-transformed n-gram statistics or SMM on the overall LID performance (see Section 7.1).

Since the LID is, in principle, a multi-class classification, we compared multi-class formulations of the logistic regression and SVM for producing language scores with binary classifiers in Section 7.5. Our conclusion is that, as long as we keep two layers of classifiers in our back-end (i.e. a first layer to generate class scores and a second layer for score calibration), the choice of binary or multi-class classifiers in the first layer does not have a significant effect on the LID performance.

The performance of the the SnGM was compared to both the baselines and to SMM in Section 7.8. We observed that SnGM is very sensitive to data sparsity and tries to generate rare n-gram probabilities which results in poor system performance on held out iVectors extracted from short observations. The RSnGM along with hard pruning of the rare n-gram statistics showed significant improvement in the performance of the phonotactic iVectors. The RSnGM outperformed the state-of-the-art baseline consistently up to 30% on all conditions. Table 8.1 summarizes performances of the baselines and different iVector extraction models on NIST LRE09.

Another objective of this thesis was to obtain Gaussian-like distributed phonotactic iVectors. This way, we could use simple Gaussian classifiers instead of the fancy support vector machines or logistic regression to generate language scores. It was already shown that a Gaussian linear classifier (GLC) produces better language scores in case of the acoustic iVectors extracted by the iVector model for continuous features [46]. In the case of the phonotactic iVectors, the GLC produces better language scores for the short conditions and worse language scores for the long conditions of the NIST LRE09 (see Section 7.5). Our conclusion is that, in general, GLC performs similar to the logistic regression on the reported tasks. An interesting observation is that the GLC fails to produce good language scores for the PCA-transformed n-grams and we get relatively poor LID system performance. We believe this is due to the distribution of the PCA-transformed n-gram statistics. A graphical description of the distribution of the values for phonotactic iVector from SnGM and PCA-transformed n-gram statistics from Chapter 4.1.1 are replicated in Figure8.1 for convenience. As we can see, iVector values are more Gaussian-like distributed

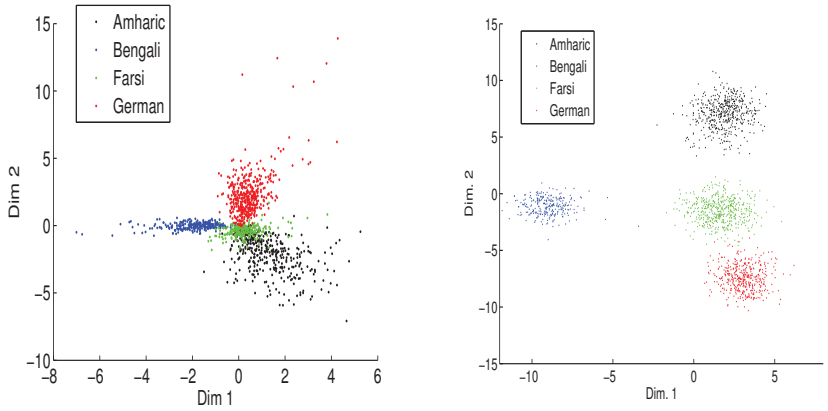
than PCA-transformed n-gram statistics.

During this thesis, we were also interested to study effect of different LID system fusions on the LID performance. Normally fusion of the systems that are exploiting different information sources, results in providing complementary information to the calibration and fusion module and can improve the overall LID performance. A simple fusion plan on the iVector level was described in Section 7.6. Two different score level fusion plans that are used during this thesis were also described in Section 5.4.1 and Section 5.4.2. We showed that iVector level fusion can also improve the LID performance as well as the score level fusion. However, score level fusion seems to produce slightly better LID performance on the reported task. Different combinations of the phonotactic, acoustic and prosodic LID systems fusions are reported in Section 7.11. An interesting observation is that fusing phonotactic LID systems based on phonotactic iVectors and PCA-transformed n-grams using n-gram statistics obtained from the same phoneme recognizers still improves system performance up to 10%. Fusion of the acoustic and phonotactic iVectors have the main contribution to the overall system performance.

To observe the performance of the proposed phonotactic iVector extraction models on other tasks, similar systems were developed for NIST LRE11 and RATS tasks. Our analysis on the individual system performance of NIST LRE11 task shows that even though acoustic iVectors are considered to be the-state-of-the-art stand-alone system for the LID tasks, it does not always provide the best language likelihoods and to get the best system performance it is necessary to fuse phonotactic iVector systems along with the acoustic iVectors. We also studied performance of our phonotactic iVector models for the most recent challenging LID task RATS in Section 7.11. Since RATS data is very different from NIST LREs data, it is hard to make a conclusion based on comparing the system performances on NIST and RATS tasks. Nevertheless, results in Section 7.13 shows effectiveness of the phonotactic iVector systems which in some cases outperform even the acoustic iVectors on the RATS task.

## 8.2 Future work

As we mentioned in Section 4.5.6, calculation of posterior distribution for $\mathbf{w}$ is intractable in the case of SnGM, which makes it difficult to treat $\mathbf{w}$ as a latent variable in the fully Bayesian way as is the case for iVectors for continuous features. Instead, SnGM parameters are updated using only $\mathbf{w}$ point estimates, which can negatively affect the robustness of SnGM param-




(a) Distribution of PCA transformed 3-gram statistics.

(b) Distribution of iVectors extracted with RSnGM.

Figure 8.1: Distribution of phonotactic iVectors and PCA transformed 3-gram statistics using BUT HU for the first four NIST LRE09 languages.

Table 8.1: The $C_{avg} \times 100$ for PCA-transformed 3-grams and phonotactic iVectors form BUT HU with BSVM and BLR classifiers on NIST LRE09 EVAL set.

| System | Dimension | 30s | 10s | 3s |
|---|---|---|---|---|
| FULL-BSVM | 35937 | 3.08 | 8.4 | 21.06 |
| PCA-BSVM | 35937→600 | 3.62 | 8.89 | 21.09 |
| PCA-BLR | 35937→600 | 2.93 | 8.29 | 22.60 |
| SMM-BSVM | 35937→600 | 3.25 | 9.47 | 23.59 |
| SMM-BLR | 35937→600 | 3.05 | 8.10 | 21.39 |
| **RSnGM-BLR** | 35937→600 | **2.52** | **7.06** | **19.11** |

eter estimation. To mitigate this problem, we propose to regularize the ML objective function using $L2$ regularization terms for both the subspace matrix $\mathbf{T}$ and the vectors $\mathbf{w}$ (see Section 4.5.7). This corresponds to imposing an isotropic Gaussian prior on both the SnGM parameters $\mathbf{T}$ and $\mathbf{w}$, and obtaining MAP rather than ML point estimates. In Figure 7.4, we showed that iVectors are Gaussian-like distributed that confirms L2 regularization of $\mathbf{w}$. However, the distribution of values of the model parameter $\mathbf{T}$ looks to be close to Laplacian. On the other hand, we know that $\mathbf{T}$ is expanding the low-dimensional latent variable subspace to the original space of n-gram log likelihoods and as a result, it is an extremely sparse matrix. A more mathematically consistent regularization would be to apply constrained L1 regularizer on $\mathbf{T}$ and L2 regularizer on $\mathbf{w}$. This modification may improve the performance of our phonotactic LID even further.

Training of the iVector extractor (i.e. iVector model hyperparameter $\mathbf{T}$) is normally the most computational expensive part of the iVector model training in both the iVector model for discrete features and the iVector model for continuous features. In the case of the iVector model for continuous features, many different optimization enhancement are proposed by the researcher to reduce computational and memory overhead of the iVector model training [11][29][84]. In the case of iVector model for discrete features, however, the numerical optimization part has not been studied as well as the iVector model for continuous features. We believe that there are rooms for improvement in this part of the iVector model for discrete features particularly since it is using a different mathematics compared to the iVector model for continuous features.

The proposed multinomial subspace modeling of the discrete features can be adapted for other purposes as well. An interesting extension of this work is to use this model as an adaptation model rather than feature extraction model. Such application is proposed in [4] for decomposition and adaptation of the GMM weights.

We believe that the proposed iVector model based on the discrete features could be used in other problems as well. An interesting potential application of this model could be in text classification using n-gram statistics collected over words. Since number of the words in a language is much more than the number of characters, application of this model would require more complicated filtering of the n-gram statistics.

# Appendix A

# Derivation of a Subspace Multinomial Model

Here, we will explain the derivation of the formula used in Chapter 4.5 for numerical optimization of the multinomial subspace model parameters estimation. For the sake of the simplicity, we consider the case of having one multinomial distribution. Assuming that all phoneme labels are drawn from a single multinomial distribution, we can write the likelihood of a sequence of phonemes $\mathbf{o}_n$ corresponding to the $n^{th}$ utterance according to the unigram model as:

$$P(o_n) = \prod_{e=1}^{E} \phi_{en}^{\nu_{en}}, \tag{A.1}$$

where $E$ is the total number of the multinomial probabilities (i.e. unigram probabilities in our case ), $\phi_{en}$ is an utterance-dependent probability of the unigram $e$, $e = 1...E$ and $\nu_{en}$ is the number of occurrences of the phoneme $e$ in the observation $\mathbf{o}_n$. Taking the logarithm of the (A.1), we get

$$\log P(o_n) = \sum_{e=1}^{E} \nu_{en} \log \phi_{en}. \tag{A.2}$$

Representing the logarithm of the $\phi_{ne}$ in the iVector form and normalize it in a softmax style, we can rewrite the Equation A.2 as:

$$\log P(o_n) = \sum_{e=1}^{E} \nu_{en} \log \phi_{en} = \sum_{e=1}^{E} \nu_{en} \log \frac{\exp(m_e + \mathbf{t}_i \mathbf{w}_n)}{\sum_{j=1}^{E} \exp(m_j + \mathbf{t}_j \mathbf{w}_n)} \tag{A.3}$$

where $m_e$ is a language-independent log probability of $e$ estimated over all the training data, $\mathbf{t}_e$ is the $e^{th}$ row of the low-rank subspace matrix $\mathbf{T}$ and $\mathbf{w}_n$ is the corresponding latent variable for the utterance $n$. The model parameter are $\mathbf{T}$ and $\mathbf{m}$ that should be estimated along with utterance-dependent latent variable $\mathbf{w}_n$ (one $\mathbf{w}$ for each utterance in the TRAIN set). We calculate the $\mathbf{m}$ in advance and keep it fixed during estimation of the other parameters as:

$$m_e = \log(\frac{\sum_{n=1}^{N} \nu_{ne}}{\sum_{j=1}^{E}\sum_{n=1}^{N} \nu_{nj}}), \tag{A.4}$$

First, we show how to estimate $\mathbf{w}$ with a fixed $\mathbf{T}$. To present our objective function for the parameters estimation, we modify the objective function in (A.2). In the following, we first explain the modification of (A.2) and corresponding derivatives for estimating $\mathbf{w}_n$ and then for $\mathbf{T}$. We can rewrite (A.2) as

$$logP(\mathbf{o}_n|\mathbf{T},\mathbf{w}) = k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \log\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)], \tag{A.5}$$

where $k$ is a constant part that depends neither on $\mathbf{T}$ nor on $\mathbf{w}_n$. Now, we can use the inequality $1 - x/\bar{x} \leq -\log(x/\bar{x})$ that holds for any $\bar{x}$ and is an equality at $x = \bar{x}$, to remove the logarithm from (A.5). As a consequence, we will be able to treat $\mathbf{w}_n$ independently. Considering $x$ to be $\sum_{hi} \exp(m_{hi} + \mathbf{t}_{hi}\mathbf{w}_n)$, we can write:

$$\begin{aligned}
&logP(\mathbf{o}_n|\mathbf{T},\mathbf{w}) = \\
&k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \log\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)] \\
&= k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \log\frac{\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)} - \log\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)] \\
&\geq k + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n + 1 - \frac{\sum_i \exp(m_i + \mathbf{t}_i\mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)} - \log\sum_i \exp(m_i + \mathbf{t}_i\bar{\mathbf{w}}_n)],
\end{aligned} \tag{A.6}$$

where $\bar{\mathbf{w}}_n$ can be any chosen vector. Including all the terms independent of $\mathbf{w}_n$ into $k'$, we introduce auxiliary function $Q$:

$$Q = k' + \sum_e \nu_e[\mathbf{t}_e\mathbf{w}_n - \frac{\sum_i \exp(m_e + \mathbf{t}_e\mathbf{w}_n)}{\sum_j \exp(m_j + \mathbf{t}_j\bar{\mathbf{w}}_n)}], \tag{A.7}$$

The auxiliary function $Q$ is a lower bound for the likelihood function in (A.5) that touches the objective function in (A.5) for $\mathbf{w}_n = \bar{\mathbf{w}}_n$. Therefor, setting $\bar{\mathbf{w}}_n$ to the current estimate of $\mathbf{w}_n$ and choosing a new $\mathbf{w}_n$ that improves the auxiliary function in (A.7) guarantees to improve the likelihood function (A.5) as well. We estimate $\mathbf{w}_n$ iteratively where in each iteration we set $\bar{\mathbf{w}}_n$ to the current value of $\mathbf{w}_n$ and we find a $\mathbf{w}_n$ that maximizes (A.7). Unfortunately, (A.7) can not be maximized analytically. Therefor, we resort to Newton-Raphson-like iterative estimation:

$$\mathbf{w}_n^{new} = \mathbf{w}_n^{old} + (\underbrace{\frac{\partial^2 Q}{\partial \mathbf{w}_n^2}}_{\mathbf{H}_n})^{-1}(\underbrace{\frac{\partial Q^T}{\partial \mathbf{w}_n}}_{\boldsymbol{\nabla}_n}) = \mathbf{w}_n^{old} + \mathbf{H}_n^{-1}\boldsymbol{\nabla}_n, \tag{A.8}$$

where, $\bigtriangledown_n$ is the first derivative of the objective function in (A.7) with respect to the $n^{th}$ utterance and $\mathbf{H}_n$ is the second derivative of the objective function in (A.7) with respect to $\mathbf{w}_n$. The $\boldsymbol{\nabla}_n$ is obtained as:

$$\begin{aligned}
\boldsymbol{\nabla} &= \frac{\partial Q^T}{\partial \mathbf{w}_n} \\
&= \sum_e \nu_e \left[ \mathbf{t}_e - \frac{\sum_i \mathbf{t}_i \exp(m_i + \mathbf{t}_i \mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i \bar{\mathbf{w}}_n)} \right] \\
&= \sum_e \nu_e \mathbf{t}_e - \sum_e \nu_e \frac{\sum_i \mathbf{t}_i \exp(m_i + \mathbf{t}_i \mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i \bar{\mathbf{w}}_n)} = \\
&= \sum_e \nu_e \mathbf{t}_e - \frac{\sum_i \mathbf{t}_e \exp(m_e + \mathbf{t}_e \mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i \bar{\mathbf{w}}_n)} \sum_i \nu_i \\
&= \sum_e \mathbf{t}_e \left[ \nu_e - \frac{\exp(m_e + \mathbf{t}_e \mathbf{w}_n)}{\sum_i \exp(m_i + \mathbf{t}_i \bar{\mathbf{w}}_n)} \right] \sum_i \nu_i \\
&= \sum_e \mathbf{t}_e \left[ \nu_e - \boldsymbol{\phi}_{en} \right] \sum_i \nu_i
\end{aligned} \tag{A.9}$$

Calculating the $\boldsymbol{\nabla}_n$ at $\bar{\mathbf{w}}_n$ results in:

$$\boldsymbol{\nabla}_n = \frac{\partial Q^T}{\partial \mathbf{w}_n|_{\bar{\mathbf{w}}_\mathbf{n}}} = \sum_e \mathbf{t}_e \left[ \nu_e - \boldsymbol{\phi}_{en}^{old} \right] \sum_i \nu_i \tag{A.10}$$

Taking the second derivative of the objective function in (4.19) gives us the equation for $\mathbf{H}$ as:

$$\mathbf{H}_n = \frac{\partial^2 Q}{\partial \mathbf{w}_n^2} = \sum_e \mathbf{t}_e \left[ - \frac{\mathbf{t}_e \exp(m_e + \mathbf{t}_e \mathbf{w}_n)}{\sum_i \mathbf{t}_i \exp(m_i + \mathbf{t}_i \bar{\mathbf{w}}_n)} \right] \sum_i \nu_i \tag{A.11}$$

Calculating the $\mathbf{H}_n$ at $\bar{\mathbf{w}}_n$ results in:

$$\mathbf{H}_n = \frac{\partial^2 Q}{\partial {\mathbf{w}_n}^2|_{\bar{\mathbf{w}}_n}} = - \sum_e \mathbf{t}_e \mathbf{t}_e^T \boldsymbol{\phi}_{en}^{old} \sum_i \nu_i \tag{A.12}$$

After each update in (A.8), we calculate the objective function in (A.7). In case it has increased compared to its value before the update, we accept the update otherwise we keep halving the update step until it results in a higher value of the objective function in (A.7). In this thesis, in case this procedure does not result in higher value in (A.7) after 10 halving, we retain the previous value of $\mathbf{w}_n$. Unfortunately this approach does not seem to be stable and it often becomes necessary to halve the step size many times. The same problem was mentioned in [60]. We used the same solution as was proposed in [60]. The following equation is used to obtain an approximate but more stable estimate of the Hessian $\mathbf{H}_n$:

$$\bar{\mathbf{H}}_n = \sum_e \mathbf{t}_e^T \mathbf{t}_e \max(\nu_{ne}, \phi_{ne}^{old} \sum_{i=1}^{E} \nu_{ni}). \tag{A.13}$$

Applying similar modification to the likelihood function in (A.2) to obtain (A.7), we can write the auxiliary function for estimation of the model parameter $\mathbf{T}$ as:

$$Q' = k' + \sum_n \sum_e \nu_e [\mathbf{t}_e \mathbf{w}_n - \frac{\sum_i \exp(m_i + \mathbf{t}_i \mathbf{w}_n)}{\sum_j \exp(m_j + \bar{\mathbf{t}}_j \mathbf{w}_n)}], \tag{A.14}$$

We use following Newton Raphson-like updates for estimation of the $\mathbf{T}$:

$$\mathbf{t}_e^{new} = \mathbf{t}_e^{old} + (\underbrace{\frac{\partial^2 Q'}{\partial \mathbf{t}_e^2}}_{\mathbf{H}_e})^{-1} (\underbrace{\frac{\partial Q'^T}{\partial \mathbf{t}_e}}_{\boldsymbol{\nabla}_e}) = \mathbf{t}_e^{old} + \mathbf{H}_e^{-1} \boldsymbol{\nabla}_e, \tag{A.15}$$

Taking the first derivative of (A.16) gives us the $\boldsymbol{\nabla}_e$ as:

$$
\begin{aligned}
\boldsymbol{\nabla}_e &= \frac{\partial Q'^T}{\partial \mathbf{t}_e} \\
&= \sum_n \left[ \nu_e \mathbf{w}_n - \frac{\exp(m_e + \mathbf{t}_e \mathbf{w}_n)\mathbf{w}_n}{\sum_i \exp(m_i + \bar{\mathbf{t}}_i \mathbf{w}_n)} \sum_e \nu_e \right] \\
&= \sum_n \left[ \nu_e - \frac{\exp(m_e + \mathbf{t}_e \mathbf{w}_n)}{\sum_i \exp(m_i + \bar{\mathbf{t}}_i \mathbf{w}_n)} \sum_e \nu_e \right] \mathbf{w}_n^T \\
&= \sum_n \left[ \nu_e - \boldsymbol{\phi}_{ne} \sum_e \nu_e \right] \mathbf{w}_n^T
\end{aligned}
\tag{A.16}
$$

Calculating the $\boldsymbol{\nabla}_e$ at $\bar{\mathbf{t}}_e$ results in:

$$
\boldsymbol{\nabla}_e = \frac{\partial Q^T}{\partial \mathbf{t}_e|_{\bar{\mathbf{t}}_e}} = \sum_n \left[ \nu_e - \boldsymbol{\phi}_{ne}^{old} \sum_e \nu_e \right] \mathbf{w}_n^T \tag{A.17}
$$

Taking the second derivative of (A.16) gives us the $\mathbf{H}_e$ as:

$$
\begin{aligned}
\mathbf{H}_e &= \frac{\partial^2 Q}{\partial \mathbf{t}_e^2} \\
&= \sum_n \left[ \nu_e - \frac{\exp(m_e + \mathbf{t}_e \mathbf{w}_n)}{\sum_i \exp(m_i + \bar{\mathbf{t}}_i \mathbf{w}_n)} \sum_e \nu_e \right] \mathbf{w}_n^T \\
&= -\sum_n \left[ \frac{\exp(m_e + \mathbf{t}_e \mathbf{w}_n)\mathbf{w}_n}{\sum_i \exp(m_i + \bar{\mathbf{t}}_i \mathbf{w}_n)} \sum_e \nu_e \right] \mathbf{w}_n^T
\end{aligned}
\tag{A.18}
$$

Calculating the $\mathbf{H}_e$ at $\bar{\mathbf{t}}_e$ results in:

$$
\mathbf{H}_e = \frac{\partial^2 Q}{\partial \mathbf{t}_e^2|_{\bar{\mathbf{t}}_e}} = -\sum_n \left[ \boldsymbol{\phi}_{nc}^{old} \sum_e \nu_e \right] \mathbf{w}\mathbf{w}_n^T \tag{A.19}
$$

Similar to (A.13), following equation is used in case of $\mathbf{H}_e$

$$
H_e = \sum_{n=1}^{N} \max(\nu_{nc}, \boldsymbol{\phi}_{nc}^{old} \sum_{e=1}^{E} \nu_{ne}) \mathbf{w}_n^T \mathbf{w}_n. \tag{A.20}
$$

# Appendix B

# HU mapping table

Table B.1 shows mapping of the long and short phonemes to the same phoneme for the BUT Hungarian phoneme recognizer.

Table B.1: HU mapping table from 61 phonemes to 33.

| | |
|---|---|
| :2 | :2 |
| A: | |
| b: | b |
| d: d:: | d |
| dz | |
| e: | |
| E | |
| f | |
| g | |
| h1 | h |
| i: | i |
| j: | j |
| J: | J |
| k: | k |
| l: | l |
| F m: | m |
| n: N | n |
| o: | o |
| O | |
| p | |
| r: | r |
| s: | s |
| </s> <s> | pau |
| S: | S |
| t: t1 t1: | t |
| ts: tS tS: | ts |
| u: | u |
| v | |
| x | |
| y: | y |
| z: | |
| z | |
| Z | |

# Bibliography


[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Courier Dover Publications, 2012.

[2] Alan Agresti. *An introduction to categorical data analysis*, volume 423. John Wiley & Sons, 2007.

[3] International Phonetic Association. *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, 1999.

[4] Mohammad Hassan Bahari, Najim Dehak, and Hugo Van hamme. Gaussian Mixture Model Weight Supervector Decomposition and Adaptation. Technical report, MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), Cambridge, MA, USA, 2013.

[5] K. Bartkova, D.L. Gac, D. Charlet, and D. Jouvet. Prosodic parameter for speaker identification. In *Proceedings of Interspeech*, 2002.

[6] Jerome R Bellegarda. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296, 2000.

[7] K.M. Berkling, T. Arai, and E. Barnard. Analysis of phoneme-based features for language identification. In *Proceedings of ICASSP*, pages 289–292, 1994.

[8] CM Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[9] N. Brümmer. The EM algorithm and minimum divergence. Technical report, Agnitio Labs Technical Report. Online: http://niko. brummer. googlepages. com/EMandMINDIV. pdf, 2009.

[10] N Brümmer, A Strasheim, V Hubeika, P Matejka, L Burget, and O Glembek. Discriminative Acoustic Language Recognition via Channel-Compensated GMM Statistics. *Proceedings of Interspeech*, 2009.

[11] Niko Brummer. *Measuring, refining and calibrating speaker and language information extracted from speech*. PhD thesis, Stellenbosch: University of Stellenbosch, 2010.

[12] Niko Brümmer, Sandro Cumani, Ondřej Glembek, Martin Karafiát, Pavel Matějka, Jan Pešán, Oldřich Plchot, Mehdi Soufifar, Edward Villiers de, and Jan Černocký. Description and analysis of the Brno276 system for LRE2011. In *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, pages 216–223, Singapur, SG, 2012.

[13] L Burget, P Matejka, P Schwarz, O Glembek, and J Cernocky. Analysis of Feature Extraction and Channel Compensation in a GMM Speaker Recognition System. *IEEE Transaction on Audio, Speech, and Language Processing*, 15(7):pp. 1979–1986, 2007.

[14] Lukáš Burget, Stephane Dupont, Harinath Garudadri, František Grézl, Hynek Heřmanský, Pratibha Jain, Sachin Kajarekar, and Nelson Morgan. QUALCOMM-ICSI-OGI Features for ASR. In *Proceedings of 7th International Conference on Spoken Language Processing*, page 4. International Speech Communication Association, 2002.

[15] Lukas Burget, Pavel Matejka, and Jan Cernocky. Discriminative training techniques for acoustic language identification. In *Proceedings of ICASSP, Toulous, France*, volume 1, pages I–I. IEEE, 2006.

[16] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech and Language*, 20:210–229, 2006.

[17] W.M. Campbell, F. Richardson, and D.A. Reynolds. Language Recognition with Word Lattices and Support Vector Machines. In *Proceedings of ICASSP*, 2007.

[18] Fabio Castaldo, Daniele Colibro, Emanuele Dalmasso, Pietro Laface, and Claudio Vair. Compensation of Nuisance Factors for Speaker and Language Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, VOL. 15(NO. 7):1969–1978, 2007.

[19] Steven Davis and Paul Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transaction on Acoustic, Speech and Signal Processing*, 28(pp. 1–4):pp. 357–366, Jul 1980.

[20] N Dehak, P Dumouchel, and P Kenny. Modeling Prosodic Features With Joint Factor Analysis for Speaker Verification. *IEEE Transaction on Audio, Speech, and Language Processing*, Jan 2007.

[21] Najim Dehak, Patrick Kenny, R eda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-End Factor Analysis for Speaker Verification. *IEEE Transaction on Audio, Speech and Language Processing*, pages pp. 1–23, Jul 2009.

[22] 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, 39(1):1–38.*, 39(1):1–38, 1977.

[23] L. Ferrer et al. *Statistical modeling of heterogeneous features for speech processing tasks*. PhD thesis, 2009.

[24] R Fletcher. *Practical Methods of Optimization*. Wiley, second edition, 2000.

[25] M.J.F. Gales. Cluster adaptive training of hidden Markov models. *IEEE Transaction on Speech and Audio Processing*, 8(4):417–428, july 2000.

[26] Daniel Garcia-Romero. *Robust Speaker Recognition based on Latent Variable Models*. PhD thesis, University of Maryland, 2012.

[27] Daniel Garcia-Romero and Carol Y Espy-Wilson. Analysis of i-vector Length Normalization in Speaker Recognition Systems. In *Proceedings of Interspeech*, pages 249–252, 2011.

[28] J.L. Gauvain, A. Messaoudi, and H. Schwenk. Language recognition using phone lattices. In *Proceedings of ICSLP*, 2004.

[29] Ondrej Glembek. *Optimization of Gaussian Mixture Subspace Model and Related Scoring Algorithm in Speaker Verification*. PhD thesis, BUT, 2012.

[30] Raymond G Gordon Jr. Ethnologue: Languages of the world, dallas, tex.: Sil international. *Online version: http://www. ethnologue. com*, 2005.

[31] A. Hatch, S. Kajarekar, and A. Stolcke. Within-class covariance normalization for SVM-based speaker recognition. In *Proceedings of Interspeech*, 2006.

[32] T. J. Hazen. *Automatic Language Identification Using a Segment-Based Approach*. MIT, 1993.

[33] Valiantsina Hubeika, Lukáš Burget, Pavel Matějka, and Petr Schwarz. Discriminative Training and Channel Compensation for Acoustic Language Recognition. In *Proceedings of Interspeech*, number 9, page 4. International Speech Communication Association, 2008.

[34] Zdeněk Jančík, Oldřich Plchot, Niko Brümmer, Lukáš Burget, Ondřej Glembek, Valiantsina Hubeika, Martin Karafiát, Pavel Matějka, Tomáš Mikolov, Albert Strasheim, and Jan Černocký. Data selection and calibration issues in automatic language recognition - investigation with BUT-AGNITIO NIST LRE 2009 system. In *Proceedings of Odyssey 2010 - The Speaker and Language Recognition Workshop*, 2010.

[35] P Kenny. Joint factor analysis of speaker and session variability: Theory and algorithms. *http://www.crim.ca/perso/patrick.kenny*, Jan 2006.

[36] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Joint Factor Analysis Versus Eigenchannels in Speaker Recognition. *IEEE Transaction on Audio, Speech, and Language Processing*, 15(4):1345–1447, May 2007.

[37] Patrick Kenny, Mohamed Mihoubi, and Pierre Dumouchel. New MAP estimators for speaker recognition. In *Proceedings of Interspeech*, 2003.

[38] A. Klapuri and M. Davy. *Signal processing methods for music transcription*. Springer, 2006.

[39] Marcel Kockmann. *Subspace modeling of prosodic features for speaker verification*. PhD thesis, Brno, CZ, 2012.

[40] Marcel Kockmann, Lukáš Burget, and Jan Cernocky. Investigation into prosodic syllable contour features for speaker recognition. *Proceedings of ICASSP, Dallas*, pages 1–4, Sep 2010.

[41] R. Kuhn, P. Nguyen, J. C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, K. Field, and M. Contolini. Eigenvoices for speaker adaptation. In *Proceedings of ICSLP*, 1998.

[42] Haizhou Li, Bin Ma, and Chin-Hui Lee. A Vector Space Modeling Approach to Spoken Language Identification. *IEEE Transaction on Audio, Speech, and Language Processing*, 15:271–284, 2007.

[43] C.Y. Lin and H.C. Wang. Language identification using pitch contour information. In *Proceedings of ICASSP*, 2005.

[44] A.F. Martin and M.A. Przybocki. NIST 2003 Language Recognition Evaluation. In *Proceedings of Eurospeech*, 2003.

[45] David González Martínez, Lukáš Burget, Luciana Ferrer, and Nicolas Scheffer. Ivector-Based Prosodic System For Language Identification. In *Proceedings of Interspeech*, pages 4861–4864, Kyoto, Japan, 2012.

[46] David González Martínez, Oldřich Plchot, Lukáš Burget, Ondřej Glembek, and Pavel Matějka. Language Recognition in iVectors Space. In *Proceedings of Interspeech 2011, Florence, Italy.*, pages 861–864, 2011.

[47] Pavel Matejka. Phonotactic and Acoustic Language Recognition. *Doctoral Thesis, Brno University of Technology*, pages 1–107, Aug 2008.

[48] Pavel Matějka, Oldřich Plchot, Mehdi Soufifar, Ondřej Glembek, Fernando Luis D'Haro, Karel Veselý, František Grézl, Jeff Ma, Spyros Matsoukas, and Najim Dehak. Patrol Team Language Identification System for DARPA RATS P1 Evaluation. In *Proceedings of Interspeech*, Portland, USA, 2012.

[49] Pavel Matějka, Petr Schwarz, Jan Černocký, and P. Chytil. Phonotactic language identification using high quality phoneme recognition. In *Proceedings of Eurospeech*, 2005.

[50] Tomáš Mikolov, Oldřich Plchot, Ondřej Glembek, Pavel Matějka, Lukáš Burget, and Jan Černocký. PCA-based feature extraction for phonotactic language recognition. In *Proceedings of Odyssey 2010 - The Speaker and Language Recognition Workshop*, pages 251–255.

[51] Tom Minka. Bayesian inference, entropy, and the multinomial distribution. *Technical Report.*, 2003.

[52] Y.K. Muthusamy, E. Barnard, and R.A. Cole. Reviewing automatic language identification. In *Signal Processing Magazine, IEEE*, 1994.

[53] Y.K. Muthusamy, N. Jain, and R.A. Cole. Perceptual benchmarks for automatic language identification. In *Proceedings of ICASSP*, 1994.

[54] T. Nagarajan and H.A. Murthy. Language identification using parallel syllable-like unit recognition. In *Proceedings of ICASSP*, 2004.

[55] J. Navratil. Recent advances in phonotactic language recognition using binary-decision trees. In *Proceedings of Interspeech*, 2006.

[56] R.W.M. Ng, Cheung-Chi Leung, Tan Lee, Bin Ma, and Haizhou Li. Prosodic attribute model for spoken language identification. In *Proceedings of ICASSP*, pages 5022–5025, 2010.

[57] Tim Ng, Bing Zhang, Long Nguyen, Spyros Matsoukas, Xinhui Zhou, Nima Mesgarani, Karel Veselý, and Pavel Matějka. Developing a Speech Activity Detection System for the DARPA RATS Program. In *Proceedings of Interspeech*, volume 2012, pages 1–4. International Speech Communication Association, 2012.

[58] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.

[59] M. Penagarikano, A. Varona, L.J. Rodriguez-Fuentes, and G. Bordel. A dynamic approach to the selection of high order n-grams in phonotactic language recognition. In *Proceedings of ICASSP*, 2011.

[60] Daniel Povey, Lukáš Burget, Mohit Agarwal, Pinar Akyazi, Arnab Ghoshal, Ondřej Glembek, K. Nagendra Goel, Martin Karafiát, Ariya Rastrow, Richard Rose, Petr Schwarz, and Samuel Thomas. The subspace Gaussian mixture model-A structured model for speech recognition. *Computer Speech & Language*, 25(2):404–439, April 2011.

[61] U.D. Reichel. Data-driven extraction of intonation contour classes. In *Proceedings of ISCA Workshop on Speech Synthesis*, pages 240–245, Bonn, 2007.

[62] DA Reynolds, TF Quatieri, and RB Dunn. Speaker verification using adapted Gaussian Mixture Models. *Digital Signal Processing*, 10(1-3):pp. 19–41, 2000.

[63] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2009.

[64] T. Schultz, I. Rogina, and A. Waibel. LVCSR-based language identification. In *Proceedings of ICASSP*, 1996.

[65] P. Schwarz, P. Matejka, and J. Černocky. Recognition of phoneme strings using TRAP technique. In *Proceedings of Eurospeech, International Speech Communication Association*, 2003.

[66] Petr Schwarz, Pavel Matějka, and Jan Černocký. Towards Lower Error Rates In Phoneme Recognition. *Lecture Notes in Computer Science*, 2004(3206):465–472, 2004.

[67] Petr Schwarz, Pavel Matejka, and Jan Cernocky. Hirarchical structures of neural networks for phoneme recognition. *Proceedings of ICASSP 2006, Toulouse*, pages pp. 325–328, Mar 2006.

[68] W. Shen and R. Reynolds. Improving phonotactic language recognition with acoustic adaptation. In *International Conferences on Spoken Language Processing*, pages 358–361, 2007.

[69] Elizabeth Shriberg, Luciana Ferrer, Anand Venkataraman, and Sachin S Kajarekar. SVM modeling of" SNERF-grams" for speaker recognition. In *Proceedings of Interspeech*, 2004.

[70] E. Singer, P.A. Torres-Carrasquillo, T.P. Gleason, W.M. Campbell, and Douglas A. Reynolds. Acoustic, phonetic and discriminative approaches to automatic language recognition. 2003.

[71] S. M. Siniscalchi, J. Reed, and C. H. Svendsen, T. andLee. Exploring universal attribute characterization of spoken languages for spoken language recognition. In *Proceedings of Interspeech*, pages 168–171, 2009.

[72] Mehdi Soufifar, Lukáš Burget, Oldřich Plchot, Sandro Cumani, and Jan Černocký. Regularized Subspace n-Gram Model for Phonotactic iVector Extraction. In *Proceedings of Interspeech*, number 8, pages 74–78. International Speech Communication Association, 2013.

[73] Mehdi Soufifar, Sandro Cumani, Lukáš Burget, and Jan Černocký. Discriminative Classifiers for Phonotactic Language Recognition with iVectors. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pages 4853–4856, Kyoto, JP, 2012.

[74] Mehdi Soufifar, Marcel Kockmann, Lukáš Burget, Oldřich Plchot, Ondřej Glembek, and Torbjorn Svendsen. iVector Approach to Phonotactic Language Recognition. In *Proceedings of Interspeech 2011*, Florence, IT, 2011.

[75] A. Stolcke, M. Akbacak, L. Ferrer, S. karayekar, C. Richey, N. scheffer, and E. Shriberg. Improving Language Recognition with Multilingual Phone Recognition and Speaker Adaptation Transforms. In *Proceedings of Odyssey The Speaker and Language Recognition Workshop*, pages 251–255, 2012.

[76] G. Strang. *Introduction to linear algebra*. Wellesley Cambridge Pr, 2003.

[77] D. Talkin. A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518, 1995.

[78] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. MIT Press, 2003.

[79] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[80] I.R. Titze and D.W. Martin. Principles of voice production. *The Journal of the Acoustical Society of America*, 104:1148, 1998.

[81] PA Torres-Carrasquillo, E Singer, MA Kohler, RJ Greene, DA Reynolds, and JR Deller Jr. Approaches to language identification using Gaussian Mixture Models and shifted delta cepstral features. *Proceedings of Interspeech*, 2002.

[82] R.C.F. Tucker, M.J. Carey, and E.S. Parris. Automatic language identification using sub-word models. In *Proceedings of ICASSP*, volume 1, pages 301–304, 1994.

[83] David A Van Leeuwen, Michaël De Boer, and Rosemary Orr. A human benchmark for the nist language recognition evaluation 2005. *Proceedings of Speaker and Language Odyssey*, 2008.

[84] Jesús A Villalba and Niko Brümmer. Towards Fully Bayesian Speaker Recognition: Integrating Out the Between-Speaker Covariance. In *Proceedings of Interspeech*, pages 505–508, 2011.

[85] Y. Yan. *Development of an approach to language identification based on language-dependent phone recognition*. PhD thesis, Oregon graduate institute of science and technology, 1995.

[86] SJ 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.

[87] MA Zissman. Overview of current techniques for automatic language identification of speech. In *Proceedings of the IEEE Automatic Speech Recognition Workshop*, 1995.

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