TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2015

81

David Martínez González

Subspace Gaussian Mixture Models for Language Identification and Dysarthric Speech Intelligibility Assessment

Departamento
Ingeniería Electrónica y Comunicaciones

Director/es
Lleida Solano, Eduardo
Miguel Artiaga, Antonio

http://zaguan.unizar.es/collection/Tesis

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

© Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

SUBSPACE GAUSSIAN MIXTURE MODELS FOR LANGUAGE IDENTIFICATION AND DYSARTHRIC SPEECH INTELLIGIBILITY ASSESSMENT

Autor

David Martínez González

Director/es

Lleida Solano, Eduardo
Miguel Artiaga, Antonio

UNIVERSIDAD DE ZARAGOZA

Ingeniería Electrónica y Comunicaciones

2015

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

Subspace Gaussian Mixture Models for Language Identification and Dysarthric Speech Intelligibility Assessment

David Martínez González

ViVoLab Speech Technology Group

Universidad de Zaragoza

PhD Thesis

Advisor: Eduardo Lleida Solano

Co-advisor: Antonio Miguel Artiaga

2015 September

Abstract

In this Thesis, we investigated how to efficiently apply subspace Gaussian mixture modeling techniques onto two speech technology problems, namely automatic spoken language identification (LID) and automatic intelligibility assessment of dysarthric speech. One of the most important of such techniques in this Thesis was joint factor analysis (JFA). JFA is essentially a Gaussian mixture model where the mean of the components is expressed as a sum of low-dimension factors that represent different contributions to the speech signal. This factorization makes it possible to compensate for undesired sources of variability, like the channel. JFA was investigated as final classifier and as feature extractor. In the latter approach, a single subspace including all sources of variability is trained, and points in this subspace are known as *i-Vectors*. Thus, one *i-Vector* is defined as a low-dimension representation of a single utterance, and they are a very powerful feature for different machine learning problems.

We have investigated two different LID systems according to the type of features extracted from speech. First, we extracted acoustic features representing short-time spectral information. In this case, we observed relative improvements with *i-Vectors* with respect to JFA of up to 50%. We realized that the channel subspace in a JFA model also contains language information whereas *i-Vectors* do not discard any language information, and moreover, they help to reduce mismatches between training and testing data. For classification, we modeled the *i-Vectors* of each language with a Gaussian distribution with covariance matrix shared among languages. This method is simple and fast, and it worked well without any *i-Vector* post-processing. Second, we introduced the use of prosodic and formant information with the *i-Vectors* system. The performance was below the acoustic system but

both were found to be complementary and we obtained up to a 20% relative improvement with the fusion with respect to the acoustic system alone.

Given the success in LID and the fact that *i-Vectors* capture all the information that is present in the data, we decided to use *i-Vectors* for other tasks, specifically, the assessment of speech intelligibility in speakers with different types of dysarthria. Speech therapists are very interested in this technology because it would allow them to objectively and consistently rate the intelligibility of their patients. In this case, the input features were extracted from short-term spectral information, and the intelligibility was assessed from the *i-Vectors* calculated from a set of words uttered by the tested speaker. We found that the performance was clearly much better if we had available data for training of the person that would use the application. We think that this limitation could be relaxed if we had larger databases for training. However, the recording process is not easy for people with disabilities, and it is difficult to obtain large datasets of dysarthric speakers open to the research community.

Finally, the same system architecture for intelligibility assessment based on *i-Vectors* was used for predicting the accuracy that an automatic speech recognizer (ASR) system would obtain with dysarthric speakers. The only difference between both was the ground truth label set used for training. Predicting the performance of an ASR system would increase the confidence of speech therapists in these systems and would diminish health related costs. The results were not as satisfactory as in the previous case, probably because the ASR acted as a filter introducing noise in the labels. Nonetheless, we think that we opened a door to an interesting research direction for the two problems.

Resumen

En esta Tesis se ha investigado la aplicación de técnicas de modelado de subespacios de mezclas de Gaussianas en dos problemas relacionados con las tecnologías del habla, como son la identificación automática de idioma (LID, por sus siglas en inglés) y la evaluación automática de inteligibilidad en el habla de personas con disartria. Una de las técnicas más importantes estudiadas es el análisis factorial conjunto (JFA, por sus siglas en inglés). JFA es, en esencia, un modelo de mezclas de Gaussianas en el que la media de cada componente se expresa como una suma de factores de dimensión reducida, y donde cada factor representa una contribución diferente a la señal de audio. Esta factorización nos permite compensar nuestros modelos frente a contribuciones indeseadas presentes en la señal, como la información de canal. JFA se ha investigado como clasificador y como extractor de parámetros. En esta última aproximación, se modela un solo factor que representa todas las contribuciones presentes en la señal. Los puntos en este subespacio se denominan *i-Vectors*. Así, un *i-Vector* es un vector de baja dimensión que representa una grabación de audio. Los *i-Vectors* han resultado ser muy útiles como vector de características para representar señales en diferentes problemas relacionados con el aprendizaje de máquinas.

En relación al problema de LID, se han investigado dos sistemas diferentes de acuerdo al tipo de información extraída de la señal. En el primero, la señal se parametriza en vectores acústicos con información espectral a corto plazo. En este caso, observamos mejoras de hasta un 50% con el sistema basado en *i-Vectors* respecto al sistema que utilizaba JFA como clasificador. Se comprobó que el subespacio de canal del modelo JFA también contiene información del idioma, mientras que con los *i-Vectors* no se descarta ninguna información del idioma, y además, son útiles para mitigar diferencias entre

los datos de entrenamiento y de evaluación. En la fase de clasificación, los *i-Vectors* de cada idioma se modelaron con una distribución Gaussiana en la que la matriz de covarianza era común para todos. Este método es simple y rápido, y no requiere de ningún post-procesado de los *i-Vectors*. En el segundo sistema, se introdujo el uso de información prosódica y formántica en un sistema de LID basado en *i-Vectors*. La precisión de éste estaba por debajo de la del sistema acústico. Sin embargo, los dos sistemas son complementarios, y se obtuvo hasta un 20% de mejora con la fusión de los dos respecto al sistema acústico solo.

Tras los buenos resultados obtenidos para LID, y dado que, teóricamente, los *i-Vectors* capturan toda la información presente en la señal, decidimos usarlos para otras tareas, en concreto, para la evaluación automática de inteligibilidad en el habla de personas con disartria. Los logopedas están muy interesados en esta tecnología porque permitiría evaluar a sus pacientes de una manera objetiva y consistente. En este caso, los *i-Vectors* se obtuvieron a partir de información espectral a corto plazo de la señal, y la inteligibilidad se calculó a partir de los *i-Vectors* obtenidos para un conjunto de palabras dichas por el locutor evaluado. Comprobamos que los resultados eran mucho mejores si en el entrenamiento del sistema se incorporaban datos de la persona que iba a ser evaluada. No obstante, esta limitación podría aliviarse utilizando una mayor cantidad de datos para entrenar el sistema.

Finalmente, la misma arquitectura del sistema de evaluación automática de inteligibilidad de habla se empleó para predecir la precisión que un reconocedor automático de habla (ASR, por sus siglas en inglés) obtendría cuando lo utilizasen personas con disartria. Predecir la precisión de un sistema de ASR permitiría aumentar la confianza que los logopedas tienen en dichos sistemas y reduciría costes relacionados con sanidad. Los resultados no fueron tan satisfactorios como para el sistema de evaluación automática de inteligibilidad, probablemente porque el sistema de ASR actuó como un filtro e introdujo ruido en las etiquetas a predecir. No obstante, el potencial mostrado por los *i-Vectors* para este tipo de aplicaciones ha abierto la puerta a caminos de investigación muy interesantes en los dos casos.

This work is dedicated to my family.

A special dedication to the pillars of my life: my little boy, Ezequiel; my wife, Mapi; and my parents, Pili and Manolo.

Acknowledgements

This manuscript is just the end of a trip that I was taking during 6 years, and where I could meet many very interesting travelers along the road. I want to thank all of those people because they are responsible for such an exciting time.

First, I want to thank Eduardo Lleida, my PhD advisor, for many reasons. He was the one who gave me the opportunity to start this PhD, back in 2009, and he always had the doors to conferences, workshops, and internships, open. Also, he was always looking for the best research directions, and his technical advices were always highly valuable.

I want to thank the rest of people of ViVoLab Speech Technologies group for their continuous support, good advices, and harmonic daily coexistence: Alfonso Ortega, Luis Vicente, Enrique Masgrau, Javier Simón, Adolfo Arguedas, Julia Olcoz, Jorge Llombart, Paola García, "fully Bayesian" Jesús Villalba, Antonio Miguel, who always had time for a passionate and enriching factor analyzed discussion, and Diego Castán, a good friend whose PhD road was parallel to mine, but both roads linked with a couple of coffee bridges per day. I can not forget those who left the group before me, but his work is still present in the lab: Óscar Saz, William Rodríguez, Carlos Vaquero, and Kike García.

I would also like to make a special mention for those who hosted me in their speech groups. In 2010, I had the opportunity to do a 4-months internship in Speech@FIT, at Brno University of Technology. No doubt, this is one of the best places in the world to learn about speech technology. I could meet superb researchers like Oldřich Plchot, Pavel Matějka, Ondřej Glembek, Mehdi Soufifar, Jan "Honza" Cernocký, and Lukáš Burget. I will say that the role played by Lukáš in this Thesis was crucial, and of course it would

not have been the same without him. If we join his brilliance to his passion for what he does and for explaining the things that he does, we obtain a global maximum in the plot of best researchers in speech technology.

Then, I had the opportunity to be an intern at SRI International for 4 months in 2011. Spectacular place. There, I was very lucky to meet Nicolas Scheffer, Luciana Ferrer, Yun Lei, and Lukáš Burget again!

It seems that I really enjoyed this of doing internships, and in 2012 I was hosted by the University of Sheffield for 4 months. Really good experience. I am tremendously grateful to Phil Green and Heidi Christensen for his collaboration.

Let me acknowledge also all the people that I met in workshops and conferences. And to conclude with, let me thank Agnitio for hiring me in January 2015. Although this probably delayed the submission of the Thesis a bit more than expected, such an exciting job was worth it. There I could work with two ex-ViVoLab members, Carlos Vaquero and Luis Buera.

In short, I want to thank all the very good researchers that were driving and crossing my road to PhD. I tried to learn and take something of all of you (without your permission, sorry for that), but now I can say that I grew on the shoulders of giants. Thanks, travelers!

Contents

List of Figures

LIST OF FIGURES

LIST OF FIGURES

List of Tables

Acronyms

AGB adaptive Gaussian backend

AHC agglomerative hierarchical clustering

ANN artificial neural network

ASGD asynchronous stochastic gradient descent

ASR automatic speech recognizer

BC between-class

BNBS broadcast narrowband speech

BUT Brno University of Technology

CBG composite background

CDHMM continuous density hidden Markov model

CLT central limit theorem

CMVN cepstral mean and variance normalization

CNN convolutional neural network

CNS central nervous system

CSS cosine similarity scoring

CTS conversational telephone speech

DCT discrete cosine transform

DEB Dysarthria Examination Battery

DNN deep neural network

DP dynamic programming

EER equal error rate

EM expectation-maximization

FA factor analysis

FDA Frenchay Dysarthria Assessment

FDLP frequency domain linear prediction

FFT fast Fourier transform

fLFA feature domain latent factor analysis

fNAP feature domain nuisance attribute projection

FM frequency modulation

GB Gaussian backend

GFCC gammatone frequency cepstral coefficient

GLDS generalized linear discriminant sequence

GMM Gaussian mixture model

HLDA heteroscedastic linear discriminant analysis

HMM hidden Markov model

ICA independent component analysis

ISCA International Speech Communication Association

JFA joint factor analysis

JHU Johns Hopkins University

KL Kullback-Leibler

LDA linear discriminant analysis

LDC Linguistic Data Consortium

LDCRF latent-dynamic conditional random field

LFCC linear frequency cepstral coefficient

LID language identification

LLR log-likelihood ratio

LM language model

LPC linear predictive coding

LR logistic regression

LRE language recognition evaluation

LSA latent semantic analysis

LSTM long-short term memory

LVCSR large vocabulary continuous speech recognition

MAP maximum a posteriori

MCE minimum classification error

MD minimum divergence

MDA multiple discriminant analysis

MFA mixture of factor analyzers

MFCC mel-frequency cepstrum coefficient

MITLL Massachusetts Institute of Technology Lincoln Laboratory

ML maximum likelihood

MLLR maximum likelihood linear regression

MLP multilayer perceptron

MLR multi-class logistic regression

MMI maximum mutual information

MPPCA mixture of probabilistic principal component analysis

NAP nuisance attribute projection

NCA neighborhood component analysis

NIST National Institute of Standards and Technology

OGI_TS OGI multi-language telephone speech corpus

PC principal component

PCA principal component analysis

PLDA probabilistic linear discriminant analysis

PLLR phone log-likelihood ratio

PLP perceptual linear prediction

PMVDR perceptual minimum variance distortionless response

PNCC power normalized cepstral coefficient

PPCA probabilistic principal component analysis

PPR parallel phone recognition

P-PRLM parallel phoneme recognition followed by language model

PRLM phoneme recognition followed by language model

PSWR parallel sub-word recognition

RASTA relative spectral

RATS robust automatic transcription of speech

RDLT region dependent linear transform

RHS right-hand side

RMSE root-mean-square error

RNN recurrent neural network

RTTH Red Temática en Tecnologías del Habla

SDC shifted delta cepstrum

SGD stochastic gradient descent

SGMM subspace Gaussian mixture model

SMM subspace multinomial model

SVC support vector classification

SVD singular value decomposition

SVM support vector machine

SVR support vector regression

SWCE sine-weighted cepstrum estimator

TALM target aware language model

TBI traumatic brain injury

TDNN time-delay neural network

TFA tied factor analysis

TFLLR term-frequency log-likelihood ratio

TMFA tied mixture of factor analyzers

TMPPCA tied mixture of probabilistic principal component analysis

TOPT target-oriented phone tokenizer

TPPCA tied probabilistic principal component analysis

TRAP temporal pattern

TUP test utterance parametrization

UBM universal background model

VAD voice activity detection

VOP vowel-onset point

VPF vector of phone frequencies

VSM vector space model

VTLN vocal tract length normalization

VQ vector quantization

WC within-class

WCCN within class covariance normalization

1

Introduction

Contents

1. INTRODUCTION

Gaussian models are extremely important in statistics. One of the reasons is the central limit theorem (CLT), that states that the mean of many random variables independently drawn from the same distribution is distributed approximately normally, irrespective of the form of the original distribution. However, there are cases where the CLT only applies locally. In such cases, we need a Gaussian distribution for each of such regions of the space where the CLT applies. A model built by a mixture of Gaussian models is a Gaussian mixture model (GMM). One important advantage of the Gaussian distribution is that it belongs to the family of exponential functions, and it has a simple mathematical definition, it is easy to differentiate, and it has a well-known integral. These properties make this function a good candidate for many tasks, because it allows deriving closed-form or tractable solutions in most cases.

In this Thesis, we investigate GMM-based models and their application in speech related tasks. Our main research focus is to find methods to compensate these models for undesired sources of variability found in the signal. For example, in a language identification (LID) task, the transmission channel introduces uncertainty in the speech signal that can make our system fail to recognize the spoken language. Nonetheless, channel compensation is one of our main goals. By introducing compensation techniques, we can build probabilistic models that better explain the generation of data, and as a result, we obtain improvements in classification rates. This defines an important property of machine learning: the better generative models fit the (test) data, the more useful they are for classification tasks. To obtain such adaptable GMMs, we will use the concept of subspace. Basically, a subspace is a vector space that is a subset of other vector space with higher dimensionality. Broadly speaking, we will look for meaningful subspaces, or in other words, subspaces that are responsible for specific components in the speech signal. Hence, we want to compress information contained in a high dimensional space into a low dimensional subspace. In our LID example commented above, we will be interested in finding a subspace that models the channel effect. A point in that subspace will be calculated for every audio file, which will describe the effect of the channel in that recording. This point will be the key factor that will allow us to compensate our GMM model. However, in the most recent experiments, it will be shown that points in a subspace modelling, not only channel, but all sources of variability, will allow us to obtain even better results. This subspace is known as the

total variability subspace, and points in this subspace are called *i-Vectors*. They play a central role in this Thesis.

The methods presented in this Thesis can be seen as variants of one underlying model known as linear Gaussian model [Roweis and Ghahramani, 1999], which has the form

$$\mathbf{o} = \mathbf{W}\mathbf{x} + \epsilon, \tag{1.1}$$

where $\mathbf{o}$ is the observed variable, $\mathbf{x}$ is a hidden state variable that summarizes the current observation, $\mathbf{W}$ is a matrix that translates $\mathbf{x}$ to the observation space, and ϵ is a Gaussian distributed observational noise with zero mean, and uncorrelated with $\mathbf{x}$ and $\mathbf{o}$. The idea is that $\mathbf{x}$ is an informative low-dimension variable that explains the sequence of N observations $\mathbf{O} = \mathbf{o}_1 ... \mathbf{o}_N$. Hence $\mathbf{W}$ will model a subspace and $\mathbf{x}$ will be the variable lying in such subspace.

The Thesis is divided in four parts where we study two different applications based on GMM subspaces. First, in Part I, Chapter 2, we describe in depth the most important linear Gaussian models needed for the understanding of the algorithms used in this Thesis. We start with the description of models that we have not used in this work, but we can say they are the *parents* of those finally used, like principal component analysis (PCA). We define GMM formally, and explain the idea of subspace in probabilistic models. Then, we see how probabilistic principal component analysis (PPCA) and factor analysis (FA) arise as a probabilistic formulation of PCA, and also how they are used for mixture modelling. Finally, we describe the tied versions of these algorithms, and we present *i-Vectors*. In Part II of the Thesis, we will face the problem of automatic spoken LID. The objective of this task is to determine the language spoken in a given audio file. Research in LID started in the 1970s. Nowadays, it can be considered as one of the most developed technologies in the field of speech processing, with great advances obtained in the last two decades. Thanks to the big amount of data available in numerous databases, recorded in different languages by the Linguistic Data Consortium (LDC)[1], and lately, to the easy and fast access to internet, where huge amount of data can be downloaded in most of languages of the world, this problem is an ideal candidate to test GMM approaches. Nonetheless, we will see that the results using these techniques are excellent, and they have become the state of the

[1] `https://www.ldc.upenn.edu`

art in LID. In Part III of the Thesis, we will use GMM subspaces in an automatic speech intelligibility assessment task for dysarthric speakers. In this case, the objective is to assess how intelligible the speech produced by dysarthric speakers is. This problem has not been investigated as much as the previous one, and possibly, one of the main reasons is the scarcity of data available for research purposes. Although limited by this inconvenient, we advance that the results obtained are very promising. The last part, Part IV, contains a chapter devoted to the conclusions of the Thesis and a chapter listing all the publications and different aspects that guarantee that this Thesis satisfies criteria of high quality.

In the next sections of this introductory chapter, we make a brief historical review of the main Gaussian subspace modelling techniques on which the methods used in this Thesis are based, we introduce the problem of LID, and we introduce the problem of intelligibility assessment of dysarthric speech.

1.1 Review of Subspace Modelling

The first ideas involving subspaces arose as dimensionality reduction techniques, such as PCA [Pearson, 1901; Hotelling, 1933; Jolliffe, 2002]. Later, subspaces were introduced in probabilistic models with the main objective of factorizing the observations into a set of unobserved variables that explain the variability in the signal, and at the same time, training a smaller number of parameters to avoid overfitting. This is what PPCA [Roweis, 1997; Tipping and Bishop, 1999b; Bishop, 2006] and FA [Spearman, 1904; Rubin and Thayer, 1982; Bartholomew et al., 2011] do. Most of the investigations with this models have been made for Gaussian distributions.

In 1992, GMM was introduced for the first time for a LID task [Nakagawa et al., 1992], and its use became very popular among speech technology scientists at mid nineties [Zhang et al., 1994; Reynolds and Rose, 1995]. In the most general GMM definition there is no subspace at all. However, the previous PPCA and FA were developed in a GMM context, and they gave rise to mixture of probabilistic principal component analysis (MPPCA) [Tipping and Bishop, 1999a] and mixture of factor analyzers (MFA) [Ghahramani and Hinton, 1996], respectively. This allowed benefiting of subspaces with multimodal data.

It was about year 2000, when some maximum a posteriori (MAP) [Gauvain and Lee, 1994] methods arose in the fields of speech and speaker recognition to adapt universal models to specific speakers. Techniques like eigenphones [Kenny et al., 2004], eigenvoices [Kuhn et al., 1998; Kenny et al., 2005], or eigenchannels [Kenny et al., 2003, 2007] were successfully developed and applied for adapting GMMs to speakers and channels. They were very closely related to the idea of GMM subspaces.

In 2004, a joint factor analysis (JFA) [Kenny and Dumouchel, 2004; Kenny, 2006; Kenny et al., 2007] model of speaker and channel was presented. JFA was seen as a model combining eigenvoices and eigenchannels, but with a much more powerful probabilistic formulation. JFA can be considered as a type of MFA model, where the speaker and channel subspaces are tied to a set of data points, and not only to one as in MFA. Actually, JFA can be considered to be a simplification of another modelling technique known as tied mixture of factor analyzers (TMFA) [Miguel et al., 2014]. TMFA is intractable in practice, and model approximations are needed to use this model with real data. Also, the version for a single Gaussian, tied factor analysis (TFA), was developed for a face recognition task [Prince et al., 2008].

In LID, JFA was adapted to be used only with one channel factor [Castaldo et al., 2007b; Hubeika et al., 2008; Campbell et al., 2008; Brümmer et al., 2009; Verdet et al., 2009; Jancik et al., 2010]. Hence, it is not a joint computation, but an eigenchannel formulation. We keep the JFA notation to make clear that it applies the same simplifications over TMFA than the speaker recognition formulation. LID results improved significantly by using JFA compared with GMMs. In current systems, JFA is used as feature extractor. Experiments in speaker recognition showed that the channel subspace still contained speaker information, and better results were obtained by using hidden factors lying in a single subspace including all sources of variability [Dehak, 2009] as input features to the classifier. These factors are the *i-Vectors*, and they were calculated in a per-file basis. In Dehak et al. [2011a], they indicate that the "i" comes from *identity*, because they are used to identify each file. In this Thesis, we show that the subspace modelling the channel in a LID JFA system also contains information about the language, and classification results improve by using *i-Vectors*. The versatility and flexibility of *i-Vectors*, and ease of adaptation to other problems, was the main reason that motivated us to use them in an intelligibility assessment task. Primarily,

the *i-Vector* approach is a dimensionality reduction technique, and thus the goal of using subspaces has returned to the initial one pursued by PCA.

In Figure 1.1 we show the relationship among the different linear Gaussian models presented before. For readers unfamiliar with these techniques, we recommend the reading of Part I and Appendix A of this Thesis, where we describe all of them in depth, with special attention to those that we have used along this work, namely GMM, JFA, and *i-Vectors*.

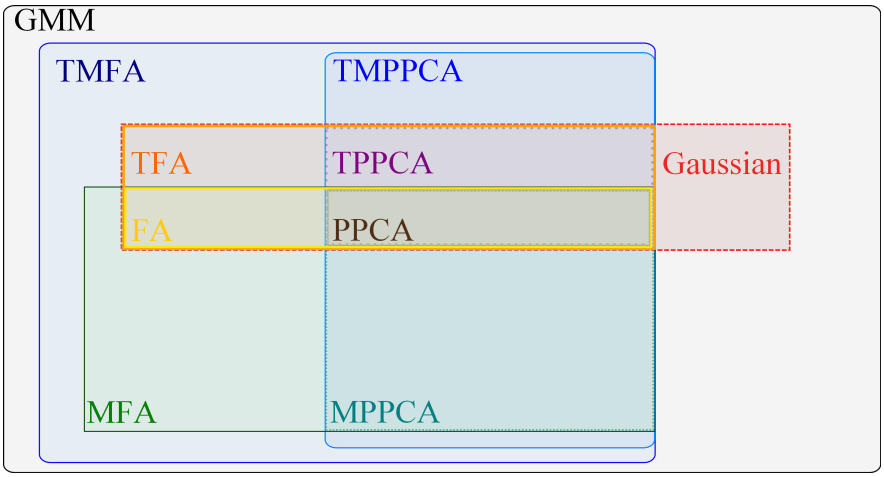


Figure 1.1: Diagram with linear Gaussian models relationship - The GMM is considered to be the most general model. The rest are different types of GMMs. Then, MFA is considered to be a type of TMFA model in which the number of tied data points is one. TFA is considered to be a TMFA model in which the number of Gaussian components is one. FA is considered to be at the same time a TFA and MFA model, because the number of tied data points is one and the number of Gaussian components is one. The family of PPCA models is considered to be a variant of FA models, where the noise covariance matrix, in addition to being diagonal as in FA, is also isotropic. Finally, the Gaussian distribution is considered to be a type of GMM where the number of Gaussian components is one. TFA, FA, TPPCA, and PPCA are considered to be Gaussian distributions with a different covariance structures, in the same way as TMFA, MFA, TMPPCA, and MPPCA are GMMs with a different covariance structures. PCA is out of this diagram because it has no associated Gaussian probabilistic distribution.

1.2 Introduction to Language Identification

Automatic spoken LID is the task of identifying the language being spoken in an audio utterance by computer devices. Nowadays, there are over 6.800.000.000 people in the world and 7.106 living languages, as per `http://www.ethnologue.com/world`. A distribution of languages according to the number of speakers can be seen in Figure 1.2. In turn, human-computer interaction is growing every day, and in many occasions it is required to know in advance the language of the user.

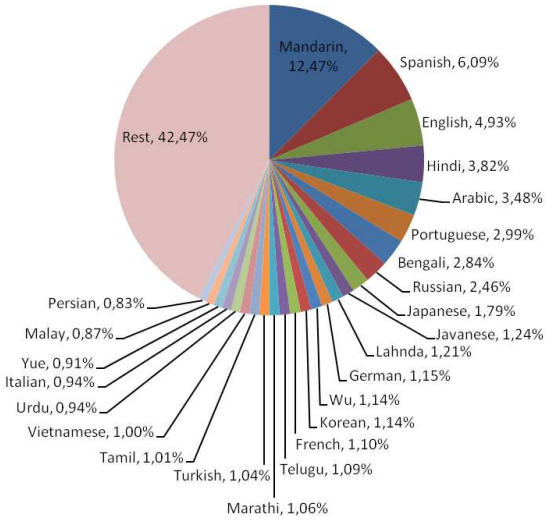


Figure 1.2: Most spoken languages in the world - Distribution of spoken languages according to the number of first-language speakers. Source: Ethnologue, `http://www.ethnologue.com`.

1.2.1 Need for Automatic Language Identification

The tasks where LID is needed are:

- Call center routing. Call centers receive thousands of calls every day from people speaking in different languages. These calls require an immediate answer of a person speaking the same language. Up to now, the routing was made manually, with an extra cost due to the time needed to find the right person. Today, LID systems can perform this task automatically with very high accuracy, and achieving great savings. In situations where emergency calls are attended, automatic

LID systems have become essential.

- Security. LID technology is considered to be a very important contribution to monitor and control communications by security agencies and armies.

- Preprocessing step in other speech systems. A LID module can be used in other multilingual speech systems. For example, you may want to use a speech recognizer where the speaker can speak in one of several languages. Then, you need a previous automatic step that switches the speech recognizer to the correct language.

- Multimedia classification. The huge amount of data generated by mobile devices and uploaded on internet must be classified. An LID system can help companies to label audio files with the correct language tag.

- Evaluation tool. LID systems can be used to evaluate how well a person speaks a foreign or nonnative language.

1.2.2 How languages differ from each other

The first thing one has to think to build a LID system is what make languages differ from each other. Languages have been historically classified according to different criteria. Probably, the most common classification is the classification into families, which groups languages by their roots. In this classification, the grouping is established by comparative linguistics, and all the languages into the same group are said to have a genealogical relationship, because all descent from the same common ancestor or *proto-language*. In Figure 1.3, taken from *Wikipedia* (`http://en.wikipedia.org/wiki/Language_family`), we can see the division of the world into language families, where each family is mainly predominant in the corresponding area.

Typology classification is based on the structural and functional features of languages. Three types of languages arise according to the word morphology: isolating, in which each word is built by a single morpheme; agglutinative, in which words are segmented into morphemes, each representing a single grammatical category; and inflecting, in which the words are segmented into morphemes, but there is no one-by-one correspondence between morphemes and grammatical categories, and one morpheme

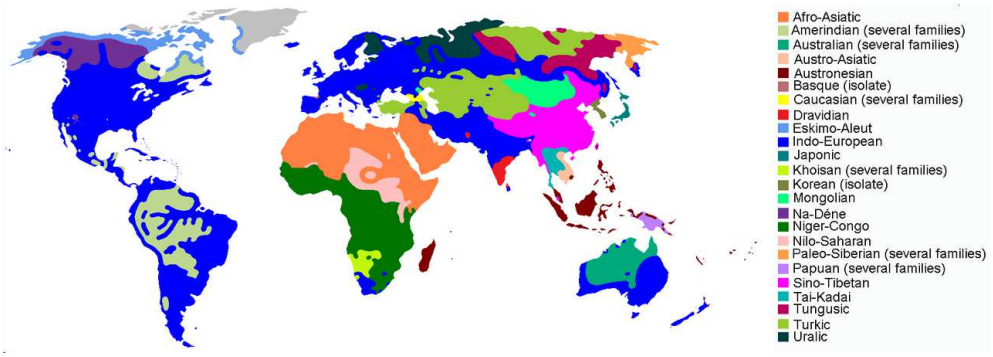


Figure 1.3: Language families in the world - There are 136 different language families in the world, 6 of them including 63.09% of all the living languages and 85.03% of speakers in the world

can represent more than one grammatical category. Languages can also be classified according to the subject-verb-object positioning, or to the phonological structure, because the frequency and relative position of phonemes is different for every language.

Another typology classification advocates that languages can be divided into rhythmic classes. Three classes are found: stress-timing, which postulates that the duration between two stressed syllables is equal, like in English; syllable-timing, which postulates that the duration of the syllables is the same, like in Spanish; or mora-timing, which postulates that the moras are placed in equal intervals of time, like in Japanese.

Based on these theoretical analysis, researchers have used the following cues to identify languages [Muthusamy et al., 1994]:

- Phonetics: sounds produced in different languages are different, and therefore the repertoire of phoneme is different from language to language. Also, the same phoneme can be acoustically different from language to language.

- Phonotactics: in one language, some phonemes can occur more frequently than in other language. Also, the combination of phonemes and the rules to combine them differ among languages.

- Morphology: those different combination of phonemes make different word roots, and finally, different words for each language. Vocabularies differ from language to language.

- Prosody: there are different sound patterns in different languages because they make different use of intonation, stress, and duration characteristics.
- Syntax: they way of combining words to make sentences differs among languages. Even if two languages share a word, this probably will have different meaning and will be combined differently.

1.2.3 The Generic Automatic Spoken LID Problem

The task of LID can be seen as a pattern recognition problem, in which machine learning techniques must be applied to teach computers languages. First, we have to extract features from the audio containing language information. At the same time, the features should have the property of being discriminative, that is, they should follow the same patterns within the same language, and different patterns among different languages. Then, a classifier must be trained with those features to recognize the language of new speech coming into the system. The basic architecture of an LID system is depicted in Figure 1.4. Given a sequence of vectors $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_N]$, the language problem can be mathematically defined as [Ambikairajah et al., 2011]

$$\hat{l} = \arg \max_{1 \leq l \leq L} P(\lambda_l | \mathbf{O}), \tag{1.2}$$

where λ_l refers to the model of language l, and L is the number of target languages. Applying Bayes' rule, the formula can be rewritten as

$$\hat{l} = \arg \max_{1 \leq l \leq L} \frac{P(\mathbf{O}|\lambda_l)P(\lambda_l)}{P(\mathbf{O})}. \tag{1.3}$$

Assuming equal probability for the models of all languages, and given that $P(\mathbf{O})$ is independent of the model, the LID task can be finally formulated as

$$\hat{l} = \arg \max_{1 \leq l \leq L} P(\mathbf{O}|\lambda_l). \tag{1.4}$$

In words, we say the the language, l, of a speech audio parameterized as $\mathbf{O}$, is the one that maximizes the probability that the observations, $\mathbf{O}$, have been generated by the model of language l, λ_l.

The task we have just defined is an identification task, and it is a multiclass problem. Given a set of L languages, it answers to the question: In which language is spoken our audio recording? And the decisor will select only one of the target languages.

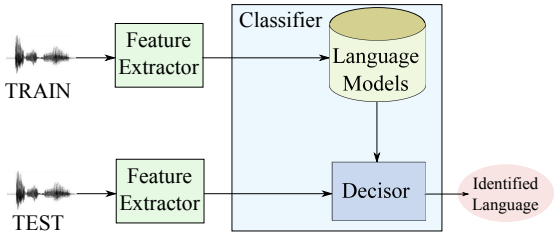


Figure 1.4: Generic LID system as a pattern recognition problem - Most general LID systems are divided into feature extraction and classification, with a training and a test phase.

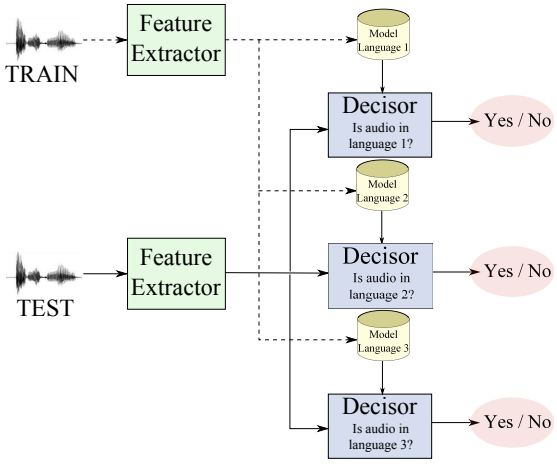


Figure 1.5: Language detection task scenario - Detection is a binary classification problem in which every test utterance is tested against all language models.

However, perhaps due to the influence of the National Institute of Standards and Technology (NIST) language recognition evaluations (LRE)[1], most systems in the literature perform the task of detection or verification. Detection is a binary classification problem performed for each target language. Thus, given a set of L target languages, the task is to test an audio recording against all target languages individually. Imagine our target languages are English, Mandarin, and Spanish. A detection task answers to three questions: Is the speech spoken in English? Is the speech spoken in Mandarin? Is the speech spoken in Spanish? Accordingly, we can say that the recording is spoken in zero, in one, or in more than one target language. In Figure 1.5, we depict the situation faced in a detection problem. Nonetheless, in this work, we will focus on the detection problem. Although the acronym LID means language identification, we will refer to

[1] `http://nist.gov/itl/iad/mig/spkr-lang.cfm`

our detection task also as LID in this Thesis.

We have indicated that we have to choose among L target languages. However, in real systems someone can speak in a different language not included in this set. Thus, we can distinguish two different situations. One, when we do not consider the possibility that someone can speak in a language not included in the set of L target languages, which is called *closed-set* problem. Two, when we do consider that someone can speak in a language not included in the set of L target languages, which is called *open-set* problem. In this work, we only focus on the closed-set problem.

1.2.4 Classification of LID Systems

Once we are aware of the main speech aspects that distinguish languages, the generic system presented above is developed to capture one or several of those aspects. LID systems can be classified into groups according to the type of information they use, which basically depends on the type of features extracted from speech. We can distinguish the following types of LID systems depending on the type of information that they use:

- Raw waveform: these systems use directly the speech signal without any transformation.
- Acoustic LID: it refers to the lowest level in the prosodic hierarchy of speech (presented below), and the goal is to represent the smallest stationary parts of the speech. Acoustic features are extracted in short windows and are lack of any linguistic meaning. They just model acoustic (commonly spectral) aspects of the speech.
- Acoustic-Phonetic LID: in these systems the basic features are different, such as articulatory features. Though, acoustic features are normally extracted first and modified later to obtain feature of this group. For example, we can model each phoneme or phonetic transcription individually by grouping acoustic features in a continuous interval, if we have the transcribed text, or simply perform an unsupervised modelling by grouping similar consecutive acoustic features.
- Phonotactic LID: usually in this approach, a phoneme recognizer is used to extract the phonemes spoken in the audio, and for each language, a statistical model

collecting how these phonemes are combined is built. During test, the phonemes extracted of each utterance are evaluated over each of the trained models. Sometimes, phonemes are grouped according to the articulatory features they belong to. These are more basic phonological units like manner or place of articulation. It is the most common token-based approach.

- Syllable-Based LID: systems where the basic unit for classification are syllables. The token-based version is the syllabotactic approach, which has the same idea as the phonotactic approach but with syllables instead of phonemes.
- Lexical LID: systems that extract lexical information like morphemes or words.
- Syntax LID: systems that extract syntactic information or are based on language rules.
- Prosodic LID: prosodic information capturing the rhythm, stress, and intonation of the speech is extracted and modeled for each target language. Prosodic information includes supra-segmental information, unlike the acoustic approach, which is focused on segmental information. Prosody can be modeled from the syllable level to the whole utterance.
- Large vocabulary continuous speech recognition (LVCSR) LID: this is the most simple idea but more difficult to implement. Ideally, if we had a perfect automatic speech recognizer (ASR) for each target language, we could recognize the audio with all of them, and observe which one gives the best output. The cost to build such a system is very high, because we would need many hours of labeled data for all the target languages, and very good ASR accuracies. Thus this alternative is infeasible in most cases today.
- Hybrid Systems: systems that mix some of the previous approaches.

The hybrid approach is a typical solution used successfully by many researchers because it combines different types of information which are complementary. The most used systems are the acoustic, the phonotactic, and the prosodic. The results are generally improved with respect to a single system alone. The fusion strategy can differ. Some authors fuse at the feature level, by concatenating features from different

sources, but the most extended strategy is to fuse at the score level, where two or more complete LID classifiers belonging to different categories are built, and the scores of each of those systems are combined in a controlled way.

Other ideas have arisen along the years that could be considered as belonging to other categories not mentioned above. We will make an extensive review of the literature in Chapter 3 and analyze the different approaches deeper.

In addition to the previous classification, LID systems can also be grouped in two categories, depending on the nature of extracted features. We find

- Vector-based methods: every utterance is represented by a set of continuous vectors. Those vectors are directly used for classification.

- Token-based methods: the speech signal is segmented into a set of discrete units, like phonemes or Gaussian component indices. The token stream is used to obtain a model of the frequency of occurrence of tokens and combinations of tokens, like n-grams counts, that will be used as features. Phonotactic and syllabotactic systems fall in this category, but they are not the only ones, since there are other levels of types of information in the speech that could be tokenized, like prosody.

The levels of the features that give rise to the different types of systems presented above are normally associated to the level of analysis of language [Ambikairajah et al., 2011], from deeper fields focused primarily on form, and surface fields focused primarily on meaning. In Figure 1.6, the different levels of linguistic structure and the corresponding LID systems into each category are shown. We have to say that, nowadays there are no known systems using semantic information or higher, mainly due to the complexity that such a system would require.

Although the linguistic levels and the LID system groups are well interconnected, we think that there are still some overlaps which make this association unclear. Actually, we have observed that the classification of LID systems matches better with the *prosodic hierarchy theory of prosodic phonology* [Selkirk, 1978]. This theory postulates that syntactic and phonological representations are not isomorphic and that there is a distinct level of representation called *prosodic structure* which contains a hierarchically organized set of prosodic constituents [Elordieta, 2008]. The prosodic hierarchy is the name for an ordered set of prosodic category types [Selkirk, 2011]. These types develop a syntactic structure that triggers the phonological rules [Hayes, 1989]. The levels in

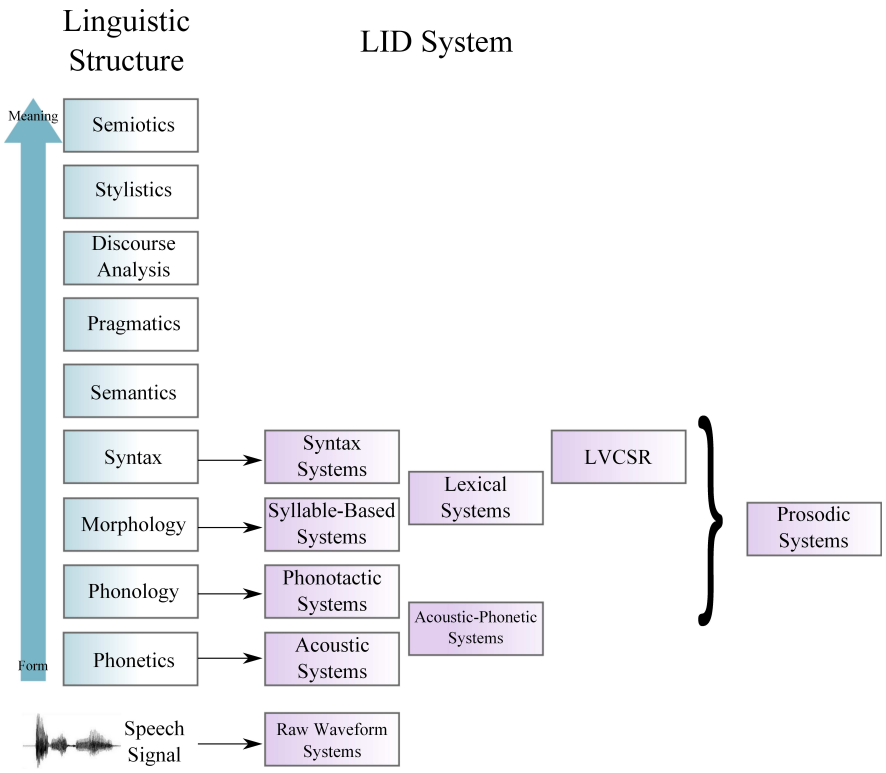


Figure 1.6: LID Systems classification according to linguistic structure - Classification of LID systems according to the linguistic level they are based on.

prosodic and syntactic hierarchies of speech are shown in Figure 1.7, together with the corresponding LID approach.

1.2.5 Objective of Part I of the Thesis

In this work, we focus on the acoustic and prosodic categories of LID. The main contribution of this work is the development and study of LID systems based on *i-Vectors* using spectral and prosodic features.

1.2.6 Organization of Part I

The Part I of the Thesis is organized as follows:

- Chapter 3: we make a historical review of the literature in LID.
- Chapter 4: in this chapter, we present the basic architecture of our LID system, we define the metrics used to report results, and we describe the database used

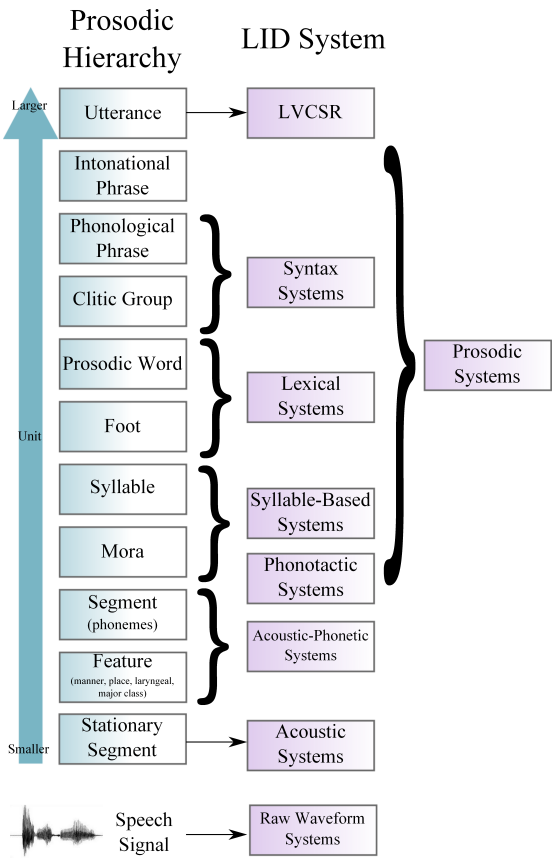


Figure 1.7: LID Systems classification according to the prosodic hierarchy of speech - The essence of this theory is that utterances are phrased and there is no isomorphic correspondence to syntactic units.

in our experiments.

- Chapter 5: this chapter is dedicated to the description of the evaluation methods used with the different linear Gaussian models.
- Chapter 6: we report results using acoustic features for LID.
- Chapter 7: we report results using prosodic and formant information for LID.
- Chapter 8: in this chapter, we study the complementarity of acoustic and prosodic information for LID.
- Chapter 9: we extrapolate our study to other databases by reporting results on a much larger and standard database like 2009 NIST LRE database; this allows

us to give general validity to our conclusions.

1.3 Introduction to Intelligibility Assessment of Dysarthric Speech

1.3.1 Dysarthria

The term dysarthria is used to refer to any of the speech disorders that are due to disturbances in neuromuscular control of the speech mechanism resulting from impairment of any of the basic motor processes involved in speech production [Darley et al., 1975]. This can affect respiration, phonation, resonance, articulation, and prosody, and can provoke abnormal characteristics in speech quality and reduced intelligibility. Six major types of dysarthria can be found depending on the affected area of the neuromotor system [Enderby, 2013]:

- flaccid associated with lower motor neurons
- spastic associated with upper motor neurons linked to the cerebral cortex
- ataxic associated with the cerebellum
- hyperkinetic and hypokinetic both associated with the extrapyramidal system
- mixed, which affects more than one of the previous areas.

The prevalence of dysarthria in society is difficult to determine and no exact numbers are given in any source of information. One of the most accepted rates indicates that dysarthria affects to 10 per 100000 people [Enderby and Emerson, 1995]. The causes of this affection can vary among affected people, and also among the different dysarthria types. According to the etiology, dysarthria can be originated by a vascular disorder, such as traumatic, infectious, neoplastic, metabolic, degenerative, psychogenic, ... Specifically, for each dysarthria type, the main causes are [González V. and Bevilacqua R., 2012]:

- Flaccid: stroke, traumatic brain injury (TBI), amyotrophic lateral sclerosis, central nervous system (CNS) tumors, neuritis, myasthenic syndromes, and different muscular dystrophy processes.

- Spastic: stroke, TBI, demyelinating disease, neoplasia, infections of the CNS, degenerative diseases, among others.
- Ataxic: stroke, TBI, cerebellar tumors, cerebellitis, among others.
- Hyperkinetic: it can be classified according to the speed of movements. If movements are fast, it is observed with choreas, ballism, Gilles de la Tourette syndrome, among others. If movements are slow, it is observed in athetosis, dystonia, and tardive dyskinesia.
- Hypokinetic: Parkinson's disease is one of the most frequent causes.
- Mixed: it can be caused by any of the causes enumerated in the previous types.

Each dysarthria type can affect to all or some of the main process affected in communication. The main symptoms in speech of each type are summarized next [González V. and Bevilacqua R., 2012]:

- Flaccid: breathy, weak voice, with hypernasality and consonant distortion.
- Spastic: voice perceived as strain, harsh, raspy, slow, showing consonant distortion and hypernasality.
- Ataxic: articulation and prosody heavily affected, hypotonia, voice with exaggerated stress, monointensity, and consonant distortion.
- Hyperkinetic: as per the previous classification, if movements are fast, the speech is monotone, with long intervals of sound, and consonant and vocalic distortion, with a harsh voice, inappropriate silences, excessive loudness variations, and hypernasality episodes. If movements are slow, we fond consonant distortion, strain and harsh voice, monointensity and monotone.
- Hypokinetic: monotone voice, monointensity, lack of stress, and hypophonia.
- Mixed: voice shows a combination of effects of previous types.

1.3.2 Intelligibility Assessment of Dysarthric Speech

Intelligibility is defined as the accuracy with which a message is conveyed in Yorkston and Beukelman [1980], and a measure of how comprehensible a speech message is according to Wikipedia [1]. In Kent et al. [1989], intelligibility is one aspect of dysarthric people's ability to communicate, but not the only one. Other aspects are naturalness, acceptability, and bizarreness. Nonetheless, intelligibility can be considered to be the one that has the major impact in communication. Speech distortions may exist but the message still be intelligible.

Our objective is to automatically measure speech intelligibility of dysarthric speakers. Clinical diagnoses of dysarthric speakers have been traditionally conducted by speech therapists, which means that there is a subjective contribution in the evaluations, resulting in disagreements among experts. Nonetheless, with a small training time, speech therapists tend to give close ratings. In Beech et al. [1993], the widest range of score ratings was 15%. In order to remove as much of this subjectivity as possible, standard methods to assess dysarthria diagnosis have been developed, like the *Dysarthria Profile* [Robertson, 1982], the *Frenchay Dysarthria Assessment* (FDA) [Enderby, 1983], or the *Dysarthria Examination Battery* (DEB) [Drummond, 1993]. All of these contain a section dedicated to rating intelligibility, because the level of intelligibility is an indication of the type of dysarthria, of the degree of the disorder, and of the relative contribution of the basic physiological mechanisms [Strand, 2004]. The benefits that speech technology brings to speech therapists are objectivity and replication of the results, and the assessment task would be easier to perform. Consequently, some implementations of these tests have introduced this type of technology. For example, in Carmichael [2007] an ASR system was used to rate intelligibility in a computerized version of the FDA.

One of the major novelties of this work is that the information contained in a whole utterance (in our case each utterance contains a single word) is compressed and represented with an *i-Vector*. As we will see, our *i-Vectors* are computed from the acoustic parametrization of the signal, and they capture all kinds of variability present in the speech. For this problem, we are only interested in intelligibility, and theoretically, *i-Vectors* should also contain this information. A similar methodological

[1] http:en.wikipedia.orgwikiIntelligibility_%28communication%29

approach can be found in Bocklet et al. [2012], but they used GMM supervectors instead. However, the great dimensionality reduction given by *i-Vectors* allows us to build much simpler predictors.

One of the main problems associated to research on dysarthric speech is the scarcity of data [Green et al., 2003]. Data recording requires several repetitions of words involving difficult movements of the speech articulators, which can be very exhausting for speakers with some dysarthric conditions. Also, it is of crucial importance the good quality of intelligibility labels, with high inter- and intra-speaker agreements, since those labels are the ground truth to be predicted by our automatic system. In this Thesis, we work on the Universal Access Speech (UASpeech) database [Kim et al., 2008], where different types of recordings belonging to 15 dysarthric speakers with different degrees of intelligibility are available. Given the limited number of speakers, the experiments conducted in previous studies on this database [Falk et al., 2011, 2012; Christensen et al., 2012; Paja and Falk, 2012; Martínez et al., 2013a] used data from the test speaker during training (naturally, data not seen in training). This would correspond to a scenario where the users of the assistive technology application were known in advance, and therefore, data of the final users could be pre-collected to build the system. However, although this is common in real life, it is not always the case, and there are occasions in which we do not know who will use the system. This opens a window to a more general situation in a clinical environment, where we would desire one single application valid for everyone. Our study gives a step forward in this direction and compares the cases where we have and where we do not have information available for training of the patient whose intelligibility will be assessed.

1.3.3 Automatic ASR Accuracy Prediction for Dysarthric Speakers

Additionally, the system is also trained to predict the *Accuracy* given by an ASR when it is used by dysarthric speakers. We realized that the same architecture designed to assess intelligibility could be directly used to predict the performance of such system, and that *i-Vectors* would also capture the information needed to make those predictions. ASR has a high potential to be used by clinicians as an assistive technology for dysarthric speakers, and if we were able to obtain a confidence measure of the recogniser, uptake rates would increase and health related costs would diminish [Mengistu et al., 2011]. People with dysarthria have, in many occasions, limited range of movements, and it

can be difficult for them to press the keys of a keyboard or move the mouse to use a computer. ASR is an ideal human-computer interaction solution to overcome these problems.

1.3.4 Organization of Part II

The Part II of the Thesis is organized as follows:

- Chapter 11: we review the literature about intelligibility assessment of dysarthric speakers, and we cite the most remarkable works on ASR accuracy prediction for dysarthric speech.
- Chapter 12: we present the experimental setup, including databases used in our experiments and application evaluation metrics.
- Chapter 13: the system architecture is described.
- Chapter 14: the experiments for intelligibility assessment and ASR accuracy prediction are detailed, including regression and classification approaches.
- Chapter 15: the goodness of the *iVectors* for intelligibility assessment is analyzed.
- Chapter 16: the conclusions of the work are drawn and discussed.

Part I

Linear Gaussian Models

2

Linear Gaussian Models

Contents

In this chapter, we explain the training process of the most relevant linear Gaussian models in our work, from PCA to *i-Vectors*. Additionally, Appendix A contains most of the formulas to compute the parameters of these models. We will refer to them along the text as necessary.

2.1 Principal Component Analysis

2.1.1 Introduction

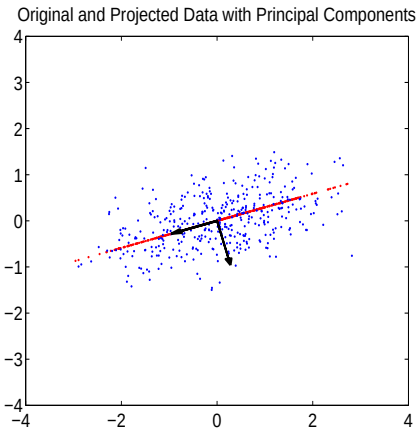
PCA [Pearson, 1901; Hotelling, 1933; Jolliffe, 2002] can be considered to be the first and most basic technique using the concept of subspace. PCA is a dimensionality reduction technique, which finds the orthogonal directions with highest variance in the data, or in other words, the most informative directions. These directions are named principal components (PCs). By projecting the data into the directions with highest variance, a compressed data representation with minimal loss of information is obtained. If most of the variance is concentrated in a reduced set of dimensions, we can achieve very good compression rates. However, PCA is only suited for Gaussian data, and it fails in other cases. In short, PCA looks for:

- Directions with maximum variance (maximum information of the data).
- Orthogonality (no redundant information in the components).

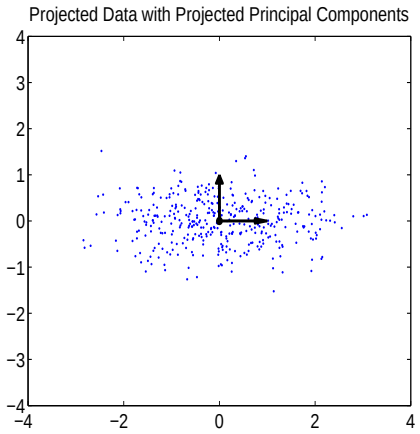
In Figure 2.1 we show a 2-dimension example with PCA applied over Gaussian data. In Figure 2.1a, we have the original 400 data points (blue points), with sample covariance matrix $\mathbf{C}_o = \left[\begin{smallmatrix} 1.27 & 0.30 \\ 0.30 & 0.33 \end{smallmatrix}\right]$, generated from a distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, with $\mathbf{\Sigma} = \left[\begin{smallmatrix} 1 & 0.5 \\ 0.5 & 0.6 \end{smallmatrix}\right]$. As we can see, the PCs (black arrows) point in the directions of maximum variance. We perform a lossy compression (red points) by projecting the data into the main PC. The loss depends on the variance in the removed dimension. We can also see that the covariance becomes diagonal if we project the data into the two PCs, what corresponds to a rotation. This is represented in Figure 2.1b, and $\mathbf{C}_x = \left[\begin{smallmatrix} 1.36 & 0 \\ 0 & 0.24 \end{smallmatrix}\right]$.

2.1.2 Limits of PCA

In Section A.1.1 of Appendix A, we present a mathematical formulation of PCA. As we can see, PCA is based on second order statistics of the data. If data are not completely



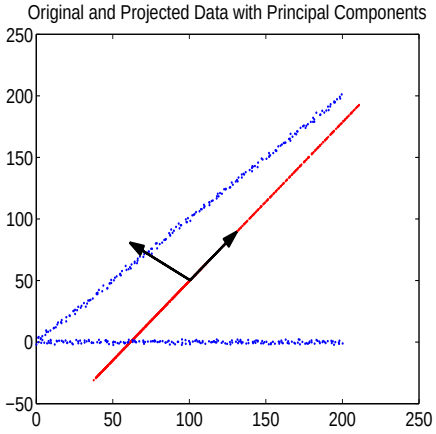
(a) PCA for data compression. PCs (black) point in the directions of maximum variance. Original data (blue) projected into the main PC (red).



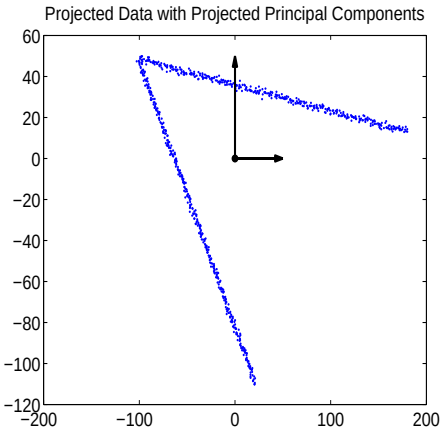
(b) PCA for data rotation. If we keep the 2 dimensions, the effect is a rotation, and the PCs are aligned with the coordinate axes.

Figure 2.1: PCA with Gaussian data.

described by second order statistics, PCA will fail. The data in the example of Figure 2.1 were Gaussian distributed, and Gaussian data are completely described by second order statistics. However, see the example of Figure 2.2, which shows a case with non-Gaussian data. In Figure 2.2a, we can see the original space (blue points) with the PCs (black arrows), and the projected points into the main PC (red points). It is obvious that the compression in this case is very bad, and there is a big loss of information. In Figure 2.2b we have the same data after a rotation using the two PCs. Clearly, we can see that the directions with the highest variance are not orthogonal, and the PCs do not point in those directions. The reason is that for non-Gaussian cases, we need higher order moments, not captured with the covariance matrix, to explain the data. If we use PCA to reduce dimensionality in these cases, a lot of information will be lost. There are other techniques to cope with such cases, as independent component analysis (ICA)[Comon, 1991], but this is out of the scope of this Thesis.



(a) PCA compression with non-Gaussian data. The compressed version (red) of the original data (blue) using the PC (black) with highest variance is very lossy.



(b) PCA rotation with non-Gaussian data. PCs do not point in directions of maximum variance, because such directions are not orthogonal.

Figure 2.2: PCA with non-Gaussian data.

2.2 Gaussian distribution

The Gaussian distribution is the essence of linear Gaussian models. The mathematical definition for a D-dimension variable $\mathbf{o}$ is

$$\mathcal{N}(\mathbf{o}|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{o}-\mu)^{\intercal}\Sigma^{-1}(\mathbf{o}-\mu)}. \tag{2.1}$$

, where μ is the mean of the distribution, and $\mathbf{\Sigma}$ is the covariance matrix.

2.3 Probabilistic Principal Component Analysis

PCA is not associated to any probabilistic model. However, a probabilistic formulation of the problem would bring several benefits [Bishop, 2006]. The most remarkable advantages are

- Enable comparison with other probabilistic techniques.
- Enable application of Bayesian methods.
- Facilitate statistical testing.

- Combination of the model into a mixture of models.
- Treatment of missing data.
- Application of maximum likelihood (ML) for efficient parameter computation.
- Use of class-dependent models for classification tasks.

The probabilistic formulation of PCA is PPCA [Roweis, 1997; Tipping and Bishop, 1999b; Bishop, 2006]. In PPCA, the transformed space is modeled by a hidden latent Gaussian variable, $\mathbf{x}$, of dimension M, with $M \leq D$, and D being the dimensionality of our observations,

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}). \tag{2.2}$$

The observations, $\mathbf{o}$, given the compressed representation of the observations, $\mathbf{x}$, are expressed as

$$\mathbf{o} = \mu + \mathbf{W}\mathbf{x} + \epsilon, \tag{2.3}$$

where μ is an offset in our observations with respect to the origin, $\mathbf{W}$ is a reduced rank DxM matrix that translates $\mathbf{x}$ from its M-dimension subspace to the D-dimension original space, and ϵ is an isotropic Gaussian noise, $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. The probability of the observations given the hidden variable is Gaussian,

$$p(\mathbf{o}|\mathbf{x}) = \mathcal{N}(\mu + \mathbf{W}\mathbf{x}, \sigma^2\mathbf{I}). \tag{2.4}$$

Unlike PCA, where we want to calculate a compressed representation of our observations, in PPCA the formulation is the other way round. From a generative viewpoint, we would first sample $\mathbf{x}$ to fix the mean of the conditional distribution of $\mathbf{o}$, and then the value of $\mathbf{o}$ would be determined by sampling the resulting conditional distribution. A good graphical representation of this is given in Figure 2.3 (Figure taken from Bishop [2006]). To imagine the generation of the density function of $\mathbf{o}$, we can think of a spray moving through the direction given by $\mathbf{W}$, painting with intensity given by $p(\mathbf{x})$.

To calculate the marginal probability of the observations, we have to apply the *sum rule* of probability and integrate out the latent variable, $\mathbf{x}$,

$$p(\mathbf{o}) = \int p(\mathbf{o}|\mathbf{x})p(\mathbf{x})d\mathbf{x}. \tag{2.5}$$

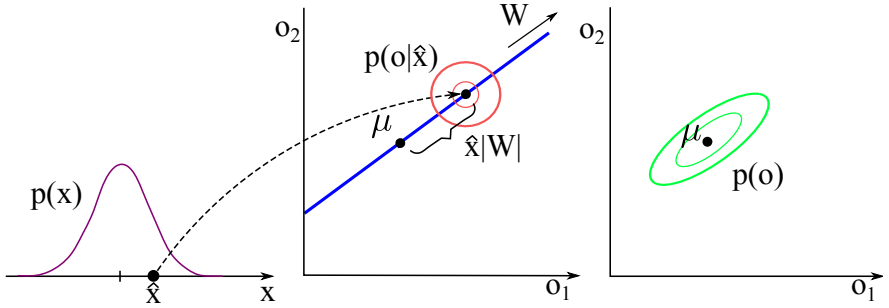


Figure 2.3: PPCA - Illustration of the the data generation process in PPCA, with a two-dimension data space and a one-dimension latent space. An observed data point, $\mathbf{o}$, is generated by first drawing a value of $\mathbf{x}$, $\hat{\mathbf{x}}$, from its prior distribution, $p(\mathbf{x})$, and then drawing a value of $\mathbf{o}$ from an isotropic Gaussian distribution (illustrated by the red circles) with mean $\mu + \mathbf{W}\hat{\mathbf{x}}$ and covariance $\sigma^2\mathbf{I}$. The green ellipses show the density contours for the marginal distribution $p(\mathbf{o})$. (Figure taken from Bishop [2006]).

The mean and covariance can be determined as

$$\mathbb{E}[\mathbf{o}] = \mathbb{E}[\mu + \mathbf{W}\mathbf{x} + \epsilon] = \mathbb{E}[\mu] + \mathbf{W}\mathbb{E}[\mathbf{x}] + \mathbb{E}[\epsilon] = \mu + 0 + 0 = \mu, \tag{2.6}$$

$$\begin{aligned} cov[\mathbf{o}] = \mathbb{E}[(\mathbf{o} - \mathbb{E}[\mathbf{o}])(\mathbf{o} - \mathbb{E}[\mathbf{o}])^\intercal] &= \mathbb{E}[(\mathbf{W}\mathbf{x} + \epsilon)(\mathbf{W}\mathbf{x} + \epsilon)^\intercal] \\ &= \mathbb{E}[\mathbf{W}\mathbf{x}(\mathbf{W}\mathbf{x})^\intercal + \mathbf{W}\mathbf{x}\epsilon^\intercal + \epsilon(\mathbf{W}\mathbf{x})^\intercal + \epsilon\epsilon^\intercal] \\ &= \mathbb{E}[\mathbf{W}\mathbf{x}(\mathbf{W}\mathbf{x})^\intercal] + \mathbb{E}[\mathbf{W}\mathbf{x}\epsilon^\intercal] + \mathbb{E}[\epsilon(\mathbf{W}\mathbf{x})^\intercal] + \mathbb{E}[\epsilon\epsilon^\intercal] \\ &= \mathbf{W}\mathbb{E}[\mathbf{x}\mathbf{x}^\intercal]\mathbf{W}^\intercal + \mathbf{W}\mathbb{E}[\mathbf{x}\epsilon^\intercal] + \mathbb{E}[\epsilon\mathbf{x}^\intercal]\mathbf{W}^\intercal + \mathbb{E}[\epsilon\epsilon^\intercal] \\ &= \mathbf{W}\mathbf{I}\mathbf{W}^\intercal + 0 + 0 + \sigma^2\mathbf{I} \\ &= \mathbf{W}\mathbf{W}^\intercal + \sigma^2\mathbf{I} = \mathbf{C} \end{aligned} \tag{2.7}$$

And then

$$p(\mathbf{o}) = \mathcal{N}(\mu, \mathbf{C}). \tag{2.8}$$

Note how the covariance of the data can be modeled by fewer parameters than a full covariance Gaussian model. Actually, an important utility of PPCA is to reduce the number of parameters to be trained, what will be very advantageous in case of reduced training datasets to avoid overtraining.

For a complete model definition we also need an expression for $p(\mathbf{x}|\mathbf{o})$. First, we know that our model is a linear Gaussian model, with $\mathbf{o}$ being a linear function of $\mathbf{x}$,

and the covariance matrix, $\mathbf{C}$, being independent of $\mathbf{x}$. Then, it can be shown that $p(\mathbf{x}|\mathbf{o})$ also follows a Gaussian distribution,

$$p(\mathbf{x}|\mathbf{o}) = \mathcal{N}(\mathbf{x}|\mathbf{B}^{-1}\mathbf{W}^{\intercal}(\mathbf{o}-\mu), \sigma^2\mathbf{B}^{-1}), \tag{2.9}$$

with $\mathbf{B} = \mathbf{W}^{\intercal}\mathbf{W} + \sigma^2\mathbf{I}$.

One important difference between PCA and PPCA is how the parameters are computed. In PCA, the matrix of basis, $\mathbf{W}$, was computed as the eigenvectors of the covariance matrix of the original data. In PPCA, the solution for $\mathbf{W}$ are also (a scaled version of) the eigenvectors of the covariance matrix, but we arrive via an ML formulation of the problem. There are two different approaches to reach the ML solution: direct differentiation of the log-likelihood function and through an iterative process like the expectation-maximization (EM) algorithm. We describe them in Sections A.2.1 and A.2.2 of Appendix A, respectively.

2.3.1 Comparison of PCA, ML PPCA, and EM PPCA

Interestingly, there can be multiple solutions in ML PPCA depending on the value of the orthogonal matrix $\mathbf{R}$, defined in eq. (A.7), which, for simplicity, is normally chosen to be the identity matrix, $\mathbf{R} = \mathbf{I}$. In this case, the solutions found by the two methods point in the same direction, but differ in a scaling factor given by $\sqrt{\lambda_i - \sigma^2}$ [Tipping and Bishop, 1999b]. Observe that in PCA we set the constraint that the basis must be orthonormal, whereas in ML PPCA, the orthogonal solution arises as a natural consequence of maximizing the log-likelihood of the data. Meanwhile, in EM PPCA the same solution is found iteratively, but also with a rotation ambiguity, which in this case, can not be controlled.

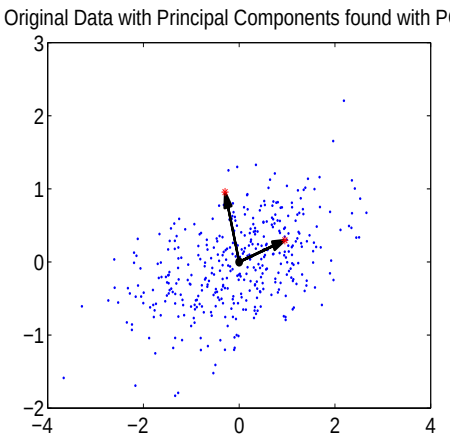
Another point to compare is the way that PCA and PPCA obtain the compressed representation of the observed data. Although the PCs found by PCA and PPCA point in the same direction, the compressed representation in PPCA is equal to the mean of $p(\mathbf{x}|\mathbf{o})$, given in eq. (2.9), whereas in PCA it is directly the projection of the data on the transpose matrix with the eigenvectors of the sample covariance matrix, as indicated in Section A.1.1. These values are, in general, not the same. The reason is that PPCA has a probabilistic nature, what involves the consideration of uncertainty, whereas PCA is a deterministic transform.

In Figure 2.4 we can see an example with 400 2-dimension points generated from a distribution $\mathcal{N}(0, \Sigma)$, with $\Sigma = \left[\begin{smallmatrix} 1 & 0.5 \\ 0.5 & 0.6 \end{smallmatrix}\right]$. The PCA solution is represented in Figure 2.4a. In Figure 2.4b, we have the solution obtained with ML PPCA with a 1-dimension subspace. Finally, in Figure 2.4c, we have the solution obtained with EM PPCA with 1-dimension subspace. We can check that the solution found by the three algorithms point in the same directions, and the PPCA solution is equal to the PCA solution multiplied by the scaling factor $\sqrt{\lambda_i - \sigma^2}$. In Figure 2.4d, we show how the log-likelihood increases in every iteration, for EM PPCA.

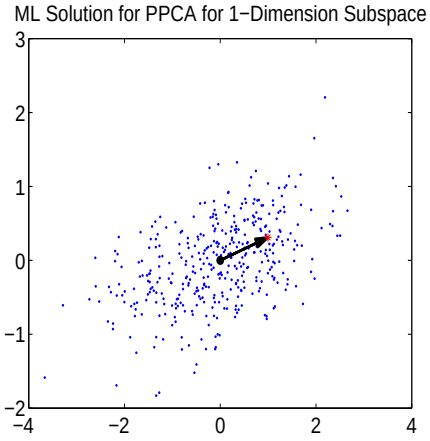
In the case of EM PPCA, we can obtain the same solution as in the ML case, up to a rotation matrix $\mathbf{R}$. This can be expressed as $\mathbf{W_{EM}} = \mathbf{W_{ML}}\mathbf{R}$. In the ML approach, we can explicitly choose $\mathbf{R} = \mathbf{I}$, but not in the EM approach. Thus, the solution found by the EM algorithm for PPCA may be non-orthogonal, due to the matrix $\mathbf{R}$. Let's illustrate this with an example. In Figure 2.5, we have an example with 400 3-data points generated from a Gaussian distribution with mean $\mathbf{0}$ and $\mathbf{\Sigma} = \left[\begin{smallmatrix} 1 & 0.5 & 0.2 \\ 0.5 & 0.6 & 0.3 \\ 0.2 & 0.3 & 0.4 \end{smallmatrix}\right]$, where we only plot the first and second dimensions. The solution given by PCA is $\mathbf{W_{PCA}} = \left[\begin{smallmatrix} 0.90 & 0.43 & 0.10 \\ 0.41 & -0.73 & -0.55 \\ 0.16 & -0.53 & 0.83 \end{smallmatrix}\right]$. The ML solution found by PPCA for a 2-dimension latent variable is $\mathbf{W_{ML}} = \left[\begin{smallmatrix} -1.04 & -0.23 \\ -0.47 & 0.40 \\ -0.19 & 0.29 \end{smallmatrix}\right]$, and $\sigma^2 = 0.14$. We can check that eq. (A.7) holds. And finally, the solution found by PPCA with the EM algorithm is $\mathbf{W_{EM}} = \left[\begin{smallmatrix} -0.68 & -0.82 \\ -0.62 & 0.03 \\ -0.33 & 0.12 \end{smallmatrix}\right]$, and $\sigma^2 = 0.14$. In Figure 2.5, we can see the basis found by the ML solution in black (they point in the same direction as the PCA solution), and the basis found by the EM solution in red. In order to know the rotation matrix, $\mathbf{R}$, responsible for the difference, we have to know that $\mathbf{R}$ is the eigenvector matrix of $\mathbf{W_{EM}}^\intercal\mathbf{W_{EM}}$, because

$$\mathbf{W_{EM}}^\intercal\mathbf{W_{EM}} = \mathbf{R}^\intercal\mathbf{W_{ML}}^\intercal\mathbf{W_{ML}}\mathbf{R} = \mathbf{R}^\intercal(\mathbf{L}_M - \sigma^2\mathbf{I})\mathbf{R} \tag{2.10}$$

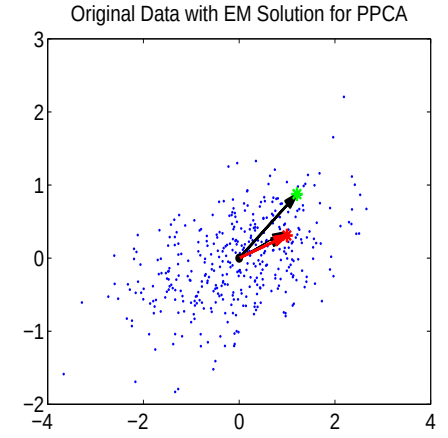
where we have used the relationship in eq. (A.7). In our example we obtain that the eigenvectors of $\mathbf{W_{EM}}^\intercal\mathbf{W_{EM}}$ are $\mathbf{R} = \mathbf{U}_M = \left[\begin{smallmatrix} 0.79 & 0.61 \\ -0.61 & 0.79 \end{smallmatrix}\right]$, which is our rotation matrix, as can be checked if we solve the problem $\mathbf{W_{EM}} = \mathbf{W_{ML}}\mathbf{R}$. It corresponds to a rotation of 37.7°.



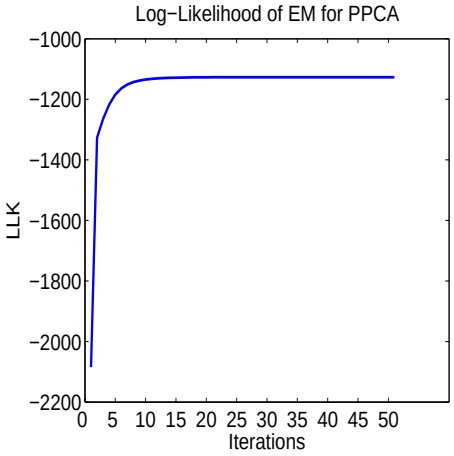
(a) PCA. PCs defined by $\mathbf{W} = \begin{bmatrix} 0.96 & 0.30 \\ -0.30 & 0.96 \end{bmatrix}$, with eigenvalues 1.32 and 0.24, for each column respectively.



(b) ML PPCA. The solution is $\mathbf{W} = \begin{bmatrix} 1.00 \\ 0.31 \end{bmatrix}$ and $\sigma^2 = 0.24$.



(c) EM PPCA. The green point indicates the initial value given to $\mathbf{W}$, the red arrow is the result after 50 iterations, and the black arrows are the intermediate iterations. The final subspace is defined by $\mathbf{W} = \begin{bmatrix} 1.00 \\ 0.31 \end{bmatrix}$ and $\sigma^2 = 0.24$.



(d) Log-likelihood over iterations in the EM PPCA. The log-likelihood increases in every iteration.

Figure 2.4: Comparison between PCA and PPCA.

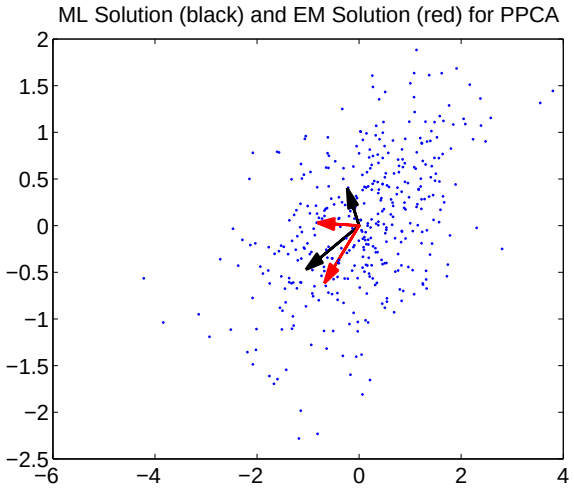


Figure 2.5: Solutions found by ML and EM PPCA - The original data has 3 dimensions, and the latent subspace has 2 dimensions. The $\mathbf{W}$ found by ML PPCA (black) is rotated by 37.7° with respect to the $\mathbf{W}$ found by EM PPCA (red).

2.4 Factor Analysis

FA [Spearman, 1904; Rubin and Thayer, 1982; Bartholomew et al., 2011] is also a linear Gaussian model, where the relationship between observed and latent variables is

$$\mathbf{o} = \mu + \mathbf{W}\mathbf{x} + \epsilon. \tag{2.11}$$

The prior distribution of the hidden variable is

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|0, \mathbf{I}). \tag{2.12}$$

The main difference between FA and PPCA is in the modeling distribution of the noise variable, ϵ. In FA, the marginal noise distribution of the noise variable is

$$p(\epsilon) = \mathcal{N}(\epsilon|0, \mathbf{\Psi}), \tag{2.13}$$

where $\mathbf{\Psi}$ is a diagonal matrix, but not isotropic Then, the conditional probability of the observations given the hidden variable is

$$p(\mathbf{o}|\mathbf{x}) = \mathcal{N}(\mathbf{o}|\mu + \mathbf{W}\mathbf{x}, \mathbf{\Psi}), \tag{2.14}$$

and the marginal distribution of the observations is

$$p(\mathbf{o}) = \mathcal{N}(\mathbf{o}|\mu, \mathbf{C}), \tag{2.15}$$

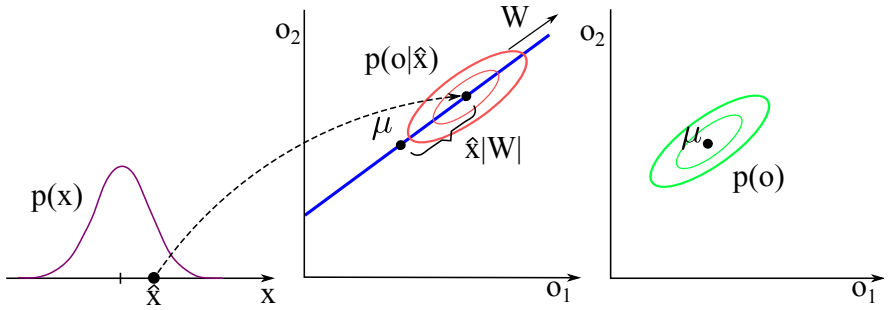


Figure 2.6: FA - Illustration of the the data generation process in FA, with a two-dimension data space and a one-dimension latent space. An observed data point, $\mathbf{o}$, is generated by first drawing a value of $\mathbf{x}$, $\hat{\mathbf{x}}$, from its prior distribution, $p(\mathbf{x})$, and then drawing a value of $\mathbf{o}$ from an diagonal Gaussian distribution (illustrated by the red ellipses) with mean $\mu + \mathbf{W}\hat{\mathbf{x}}$ and covariance $\mathbf{\Psi}$. The green ellipses show the density contours for the marginal distribution $p(\mathbf{o})$. (This Figure is an adaptation of Figure 2.3 to the FA case.)

with $\mathbf{C} = \mathbf{W}\mathbf{W}^\intercal + \mathbf{\Psi}$. The consequence is that FA makes distinction between variance and covariance, whereas PPCA, as we saw, does not. In FA, the first therm in the covariance, $\mathbf{W}\mathbf{W}^\intercal$, models the correlations among different dimensions, while the second term, $\mathbf{\Psi}$, models the variance of each individual dimension. The columns of $\mathbf{W}$ are called *factor loadings* and and the diagonal elements of $\mathbf{\Psi}$ are called *uniqueness*. For each data point, the role of the hidden variable is to capture the correlation among different dimensions.

In Figure 2.6 we have a graphical view of the generative process of FA. In this case and unlike PPCA, when we select a vector $\mathbf{x}$, the covariance of the resulting distribution is diagonal but not necessarily isotropic. We can consider PPCA as a particular case of FA, where this covariance has all diagonal elements equal.

The EM derivation of FA can be found in Section A.3.1 of Appendix A.

2.5 Gaussian Mixture Model

The previous models are useful when data are unimodal. However, there are many real cases where the distribution of the data is more complicated, and several modes appear. For such cases, we need mixtures of the previous models that capture multimodality in data. The most known multimodal distribution is the GMM. A GMM is a parametric density function represented as a weighted sum of Gaussian component

densities [Reynolds and Rose, 1995]. It is considered as a generalization of linear Gaussian models because it is a combination of linear Gaussian models. In this work, we assume that every observation is generated by one of the Gaussians of a mixture, thus every observation comes from a linear Gaussian model.

The mathematical formulation of a GMM distribution for a D-dimension continuous variable $\mathbf{o}$ is

$$p(\mathbf{o}|\mathbf{\Theta}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{o}|\mu_k, \mathbf{\Sigma}_k), \tag{2.16}$$

where K is the number of Gaussian components in the mixture, $\Theta = \{\omega, \mu, \mathbf{\Sigma}\}$ are the model parameters collectively represented, ω_k, μ_k and $\mathbf{\Sigma}_k$ are the weight, mean and covariance matrix associated with component k, respectively, and the weights satisfy the constraints $\omega_k \leq 0$ and $\sum_{k=1}^{K} \omega_k = 1$.

For maximization, it is usually more convenient to use the logarithm of the likelihood or log-likelihood instead of the likelihood. The reason is that the product of very low probabilities can underflow the numerical precision of the computer, and the computation of sums of log probabilities instead of product solves this problem. The logarithm function is monotonic, so the final result will be the same. The logarithm of function 2.16 for a dataset $\mathbf{O}$ of N points is given by

$$\ln p(\mathbf{O}|\mathbf{\Theta}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{o}_n|\mu_k, \mathbf{\Sigma}_k)\}, \tag{2.17}$$

The ML training of this model has no closed-form solution and it must be performed iteratively. Normally, the EM algorithm is used. As we know, the objective of a ML formulation is to find the parameters $\mathbf{\Theta}$ that maximize the likelihood function. The logarithm in formula 2.17 acts on the summation over k and not directly on the Gaussian. Setting the derivatives to zero will not obtain a closed-form solution for the calculation of the parameters. The full derivation of the EM algorithm for GMM can be consulted in Section A.4.1 of Appendix A.

2.6 Mixture of PPCAs

The nice properties of PPCA explained in previous sections can be applied to mixture models. Such a model is known as MPPCA [Tipping and Bishop, 1999a]. In MPPCA the components of a GMM are modeled according to the PPCA model defined in eq.

(2.3). The PPCAs of each component are independent one of each other. The same properties of PPCA hold locally for each PPCA in the mixture.

Now, the generative model requires the random choice of a Gaussian component according to the proportions given by ω, as defined for a GMM, and then sampling from the distributions of the hidden variable $\mathbf{x}$, and from the noise variable ϵ. However, the parameters μ_k, $\mathbf{W}_k$, and σ_k^2 are component-dependent. In short, there is a different subspace for each Gaussian component. So, the model is the same defined by eq. (2.16), but with the covariance structure given by

$$\mathbf{\Sigma}_k = \mathbf{W}_k \mathbf{W}_k' + \sigma_k^2 \mathbf{I}. \tag{2.18}$$

Unlike PPCA, the training of MPPCA can only be done with an iterative algorithm like the EM algorithm. We have to maximize the objective function of a GMM but with the covariance structure presented before. In this case, we have to include two hidden variables into the EM objective function, $\mathbf{z}$, that accounts for the Gaussian component that generates the data, and $\mathbf{x}$, that indicates a point in the subspace. The result is that for each Gaussian component, we will obtain a different PPCA model. In Section A.5.1 of Appendix A, we summarize the EM algorithm for MPPCA.

2.7 Mixture of FAs

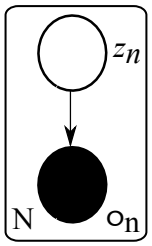
The MFA [Ghahramani and Hinton, 1996] is a GMM where each Gaussian component is modeled with FA. It is similar to MPPCA, but instead of PPCA, FA is used within each component, and therefore, the covariance matrix of the noise variable is diagonal instead of isotropic. The covariance matrix in Gaussian component k is

$$\mathbf{\Sigma}_k = \mathbf{W}_k \mathbf{W}_k' + \mathbf{\Psi}_k. \tag{2.19}$$

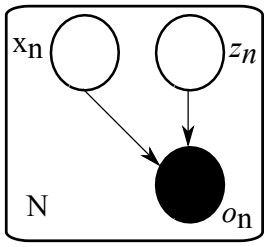
The model parameters are estimated with the EM algorithm. We summarize the E and M steps in Section A.6.1 of Appendix A.

2.7.1 Comparison of Mixture Models

In this section, we see how the different methods presented above differ in practice. First, we show the graphical models of GMM and the associated subspace methods.



(a) Graphical model of a GMM. The unobserved variable $\mathbf{z_n}$ determines the Gaussian component that originates the observed variable $\mathbf{o_n}$ at time n.



(b) Graphical model of MPPCA or MFA. We have two unobserved variables. $\mathbf{z_n}$ determines the Gaussian component that originates the observed variable $\mathbf{o_n}$ at time n. $\mathbf{x_n}$ is the low dimension representation of $\mathbf{o_n}$.

Figure 2.7: Graphical models of GMM without or with subspace modeling.

As we can see in Figure 2.7, the only difference introduced in the subspace methods compared to GMM is the addition of an unobserved continuous latent variable.

In Figure 2.9, we present a simple example with data coming from 8 clusters with different covariance matrix around the origin of coordinates. Each cluster is represented with an ellipse of a different color. From each of them, a total of 35 2-dimensional points were generated. We trained 5 mixture models: an isotropic GMM, a diagonal GMM, a full covariance GMM, an MPPCA, and an MFA. The latent variables in MPPCA and MFA were 1-dimensional. The training was done by running 50 iterations of EM algorithm. First, we can see that the simplest models with fewer number of parameters, the isotropic and diagonal GMMs, clearly do not capture the correlations in the data. The other three do capture correlations. In Figure 2.8, it is shown the log-likelihood of each model at each iteration of training. Each model was initialized with the parameters estimated for the previous model. The maximum log-likelihood on training data was obtained with the full covariance GMM. The gap at the beginning of the MPPCA model is due to the initialization of the subspace. Then, we generated another set of test data not seen during training, from the same models as the training data. The total log-likelihood obtained by each of the models on the test data is given in Table 2.1. We can see that the maximum log-likelihoods were obtained with MPPCA and MFA, although their log-likelihoods on training data were lower than the log-likelihood obtained by the full covariance GMM. The reason is that subspace models contain fewer

Isotropic	Diagonal	Full Covariance	MPPCA	MFA
-6270.9	-6259.3	-6199.3	-6197.3	-6199.0

Table 2.1: Log-likelihood for different mixture models - Log-likelihood over test data obtained with the isotropic GMM, diagonal GMM, full covariance GMM, MPPCA, and MFA.

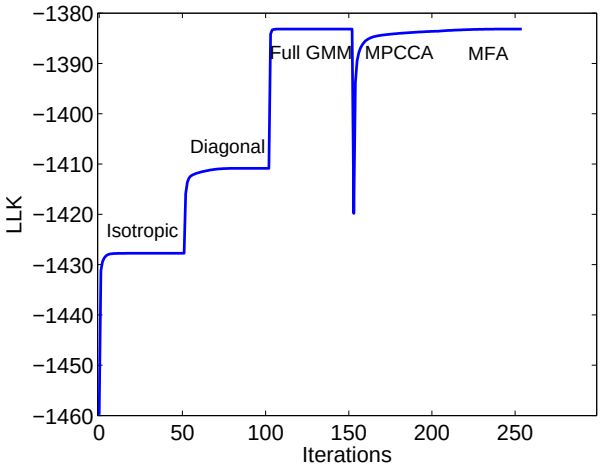


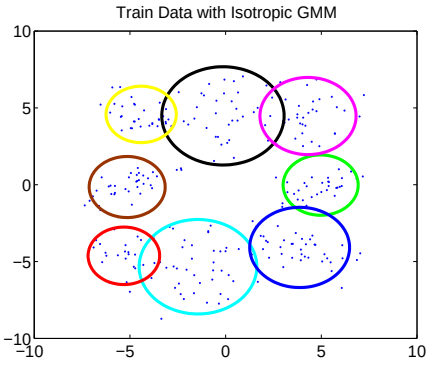
Figure 2.8: Log-likelihood on training data obtained by different mixture models - The log-likelihood is shown for each mixture model: isotropic, diagonal, full covariance GMM, MPPCA, and MFA. 50 iterations were run for training each model. Each model was initialized with the one on the left. The gap between full covariance GMM and MPPCA is due to the MPPCA subspace initialization.

number of parameters, and avoid overtraining, especially when the training dataset is small.

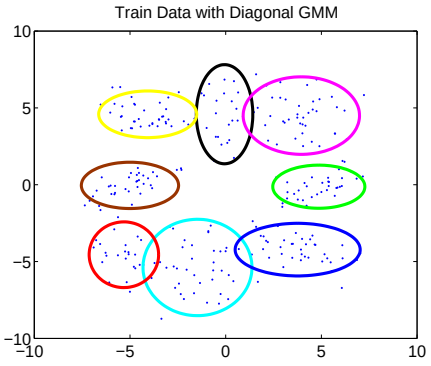
2.8 Maximum A Posteriori Adaptation

Before Patrick Kenny presented JFA in 2004 [Kenny and Dumouchel, 2004], he investigated other MAP [Gauvain and Lee, 1994] methods for speaker adaptation. We will review the main ideas behind those methods for a better understanding of JFA. In fact, JFA can be considered to be the union in a single algorithm of up to three MAP adaptations. The main difference among the different MAPs is that they model different aspects of the speech signal, and they capture different correlations among data.

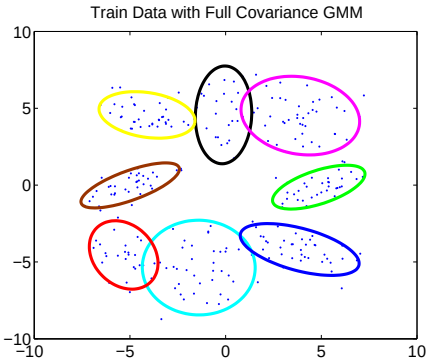
The first method he investigated was eigenphone MAP [Kenny et al., 2000, 2004].



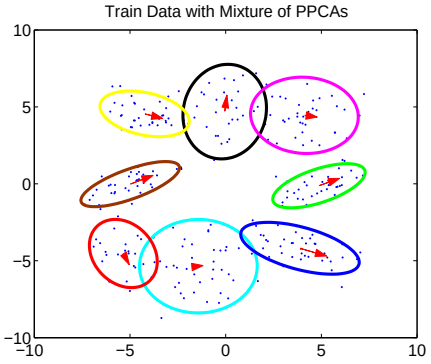
(a) Isotropic covariance GMM. Final isotropic GMM obtained in training.



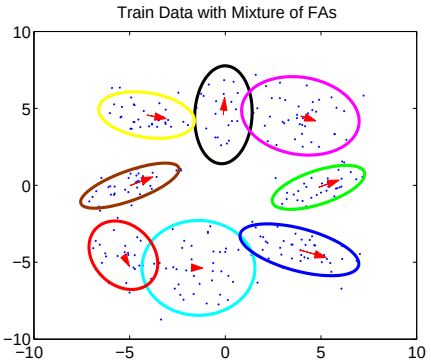
(b) Diagonal covariance GMM. Final diagonal GMM obtained in training.



(c) Full covariance GMM. Final full covariance GMM obtained in training.



(d) MPPCA. Final MPPCA obtained in training. Arrows point in the direction of the subspaces.



(e) MFA. Final MFA obtained in training. Arrows point in the direction of the subspaces.

Figure 2.9: Gaussian mixture models trained with different covariance structures: isotropic, diagonal, full, MPPCA, and MFA.

The key assumption of this method is that the speakers of the database are correlated whereas the components of the mixture are statistically independent. If D is the dimension of the features and S is the number of speakers, there exists a symmetric matrix, $\mathbf{K}$, of size SDxSD that collects the correlations among speakers, and will allow combining data from all of them to create speaker adapted models. This matrix is called the *speaker correlation matrix*. The rationale behind this method is that, if we have a GMM universal background model (UBM) that models all the speakers, with mean μ_k and covariance $\mathbf{\Sigma}_k$ for Gaussian component k, an observation $\mathbf{o}$ generated by component k and speaker s can be expressed as

$$\mathbf{o} = \mu_k + \mathbf{M}_{sk} + \epsilon, \tag{2.20}$$

where $\mathbf{M}_{sk}$ is Gaussian distributed with mean $\tilde{\mathbf{M}}_{sk}$ and covariance $\tilde{\mathbf{K}}_{ss}$, and ϵ is Gaussian with mean $\mathbf{0}$ and covariance $\mathbf{\Sigma}_k$. These mean and covariance are the parameters of the posterior distributions for speaker s given the data of all the speakers, and if $\mathbf{S_{ok}}$ is the concatenation of the first order statistics of the observations around μ_k of all the speakers, the posterior parameters, concatenated for all the speakers, can be calculated as

$$\tilde{\mathbf{M}}_k = \mathbf{K}_k \mathbf{L}_k^{-1} \mathbf{\Sigma}_k^{-1} \mathbf{S_{ok}} \tag{2.21}$$

$$\tilde{\mathbf{K}}_k = \mathbf{K}_k \mathbf{L}_k^{-1}, \tag{2.22}$$

with $\mathbf{L}_k = \mathbf{\Sigma}_k^{-1} \mathbf{N}_k \mathbf{K}_k + \mathbf{I}$, and $\mathbf{N}_k$ is the concatenation of the zeroth order statistics of all the speakers in the database for component k in matrix form. As we can see, these parameters depend on all the speakers of the database.

The dual of this model is called eigenvoice MAP [Zavaliagkos et al., 1995; Kuhn et al., 1998; Kenny et al., 2000, 2002, 2005]. In this method, speakers are statistically independent, and Gaussian components are correlated. If $\mathbf{M}_0$ is the concatenation of the means of all the components of the UBM, the model of speaker s can be expressed as

$$\mathbf{M}_s = \mathbf{M}_0 + \mathbf{V}\mathbf{y}_s, \tag{2.23}$$

where $\mathbf{y}_s$ is a hidden variable with standard normal distribution. The premise is that speaker information is low-dimensional.

The posterior distribution of $\mathbf{M}_s$ given the data can be expressed as

$$\tilde{\mathbf{M}}_s = \mathbf{M}_0 + \mathbf{V}\mathbf{l}_s^{-1}\mathbf{V}^\intercal\boldsymbol{\Sigma}^{-1}\mathbf{S}_s, \tag{2.24}$$

$$\tilde{\mathbf{B}}_s = \mathbf{V}\mathbf{l}_s^{-1}\mathbf{V}^\intercal, \tag{2.25}$$

where $\mathbf{l}_s = \mathbf{I} + \mathbf{V}^\intercal\boldsymbol{\Sigma}^{-1}\mathbf{N}_s\mathbf{V}$. $\mathbf{N}_s$ is the concatenation of the zeroth order statistics of all the Gaussian components for speaker s, and $\mathbf{S}_s$ is the concatenation of the first order statistics of all the Gaussian components for speaker s. As we can see, for each speaker, the posterior distribution parameters depend on the data of that speaker in all components.

Then, suppose that $\mathbf{o}$ follows a GMM distribution, where each component k has mean $\mathbf{M}_{k0}$ and covariance Σ_k. If we know that an observation has been generated by speaker s and component k, we can express

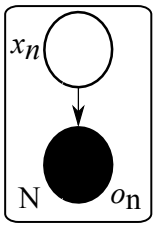
$$\mathbf{o} = \mathbf{M}_{ks} + \epsilon \tag{2.26}$$

with ϵ following a normal distribution with mean $\mathbf{0}$ and covariance $\boldsymbol{\Sigma}$. Then, the posterior probability of $\mathbf{o}$ given the speaker s, has a mean equal to $\tilde{\mathbf{M}}_{sk}$, and a covariance equal to $\tilde{\mathbf{B}}_{ks} + \boldsymbol{\Sigma}_k$.

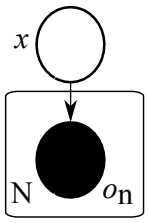
He investigated a third MAP model to adapt the channel. It is analogous to the eigenvoice model, but with the channel playing the role of the speaker. It is known as eigenchannel MAP [Kenny et al., 2003, 2007]. The premise is that channel information is also low-dimensional. In eigenchannel MAP, once you have a model for speaker s, as the once given by eq. (2.26), you proceed to adapt the channel. If $\tilde{\mathbf{M}}_s$ is the concatenation of the means of all the Gaussian components of the posterior distribution for speaker s, the model that considers the channel effect can be expressed as

$$\tilde{\mathbf{M}}_{sc} = \tilde{\mathbf{M}}_s + \mathbf{U}\mathbf{x}, \tag{2.27}$$

where $\mathbf{x}$ is a hidden latent variable with standard normal distribution, and $\mathbf{U}$ is a low-rank matrix spanning the channel subspace. The mathematical formulation is similar to eigenvoice MAP, but now the initial model can be the speaker adapted model (note that being adapted to a speaker is not a requirement), and we look for the channel adapted model for speaker s. As in eigenvoice MAP, it exploits correlations among different Gaussian components, but now assuming that channels are independent.



(a) Graphical model of FA. The unobserved variable $\mathbf{x_n}$ is the reduced version of $\mathbf{o_n}$ at time n.



(b) Graphical model of TFA. The unobserved variable $\mathbf{x}$ is tied to the N samples of the observed variable $\mathbf{O} = \mathbf{o_1}...\mathbf{o_N}$.

Figure 2.10: Graphical models of FA and TFA.

2.9 Tied Factor Analysis

Tied factor analysis (TFA) was first introduced for a face recognition task [Prince et al., 2008]. It follows the same foundations than FA but with a stronger constraint over the hidden variable. In TFA, the hidden variable $\mathbf{x}$ is forced to be the same over a set of data points. The physical meaning is that this set of points has one commonality underlying in their generation process. For example, we may want all the samples in a file to share the hidden variable because they were transmitted over the same channel, or even we may want that all the samples belonging to a set of files generated by the same speaker share the hidden variable. In fact, we can think of as many meanings for the hidden variable as aspects labeled in our database are available. In short, the hidden variable is used to model a specific aspect of the data: the speaker, the channel, the face, etc. Now, our linear Gaussian model formula for FA seen in eq. (2.11), and consequently, the probability of our observations given the hidden variable seen in eq. (2.14), acquire a new utility. TFA can be used as a MAP technique to adapt a universal model with mean μ to a specific aspect of the speech captured by the hidden variable $\mathbf{x}$. For clarity, we present the graphical models of FA and TFA in Figure 2.10.

In Figure 2.11, we see an example of TFA with 2-dimension Gaussian data and a 1-dimension hidden subspace. The points in different colors belong to different files. The hidden variable models the channel and therefore it is tied to all the data points within a file. The subspace points in the direction where the files are aligned, although the sign of the subspace is not important (a valid solution would be to multiply by -1 the obtained solution). A specific value of the hidden variable, $\mathbf{x}$, will fix the mean of

the model that generated the data along the subspace pointed by the red arrow, and the points of the corresponding file would be around such mean.

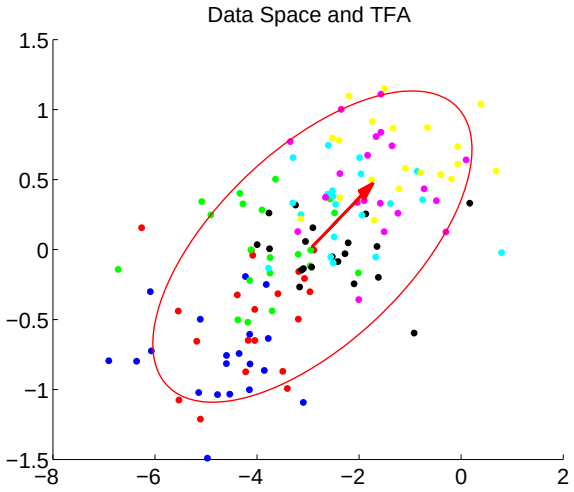


Figure 2.11: TFA - 2-dimension example of TFA with a 1-dimension subspace. The hidden variable models the channel and it is tied to all data points within a file. All the points in the same color belong to the same file.

The mathematical formulation is relegated to the next section, where we explain TMFA, the mixture approach of TFA. As it happens in all the the relationships between single Gaussian and mixture of Gaussians models, TFA can be considered as a special case of TMFA. This concept could be also applied for PPCA, but we do not give the details for it in this Thesis.

2.10 Tied Mixture of Factor Analyzers

TMFA [Kenny et al., 2004; Miguel et al., 2014] is the multimodal version of TFA. It is related to MFA but the hidden variable is tied to a set of data points. Tying the hidden variable allows us to capture the correlations among different Gaussian components. Thus we learn how the points generated by a given Gaussian component are distributed, if we know their distribution in the rest of Gaussian components. As in TFA, the hidden variable can learn one underlying aspect of the speech, like the speaker or the channel, and be used to adapt a universal model to a specific scenario, but in addition, multimodality is considered. Unlike MFA, the sign of each subspace is

important, because it is related with the signs of the rest of subspaces. In MFA, we can change the sign of any subspace and obtain the same model, because they only have a local meaning. However, in TMFA, if we change the sign of any subspace we will not obtain the same solution. A valid solution would be obtained by changing the sign of all the subspaces simultaneously.

In this section we present the exact formulation of TMFA. However, the obtained solution is infeasible for real cases due to the required computational load. This model was first introduced with some approximations to make it tractable with the name of JFA [Kenny and Dumouchel, 2004; Kenny, 2006; Kenny et al., 2007]. JFA was an evolution of the MAP techniques presented in Section 2.8. Initially, it contained up to three hidden variables, two of them to model the speaker, and one to model the channel. Later, it has been used for many other problems, like LID, with different number of hidden variables [Castaldo et al., 2007b; Hubeika et al., 2008; Campbell et al., 2008; Brümmer et al., 2009; Verdet et al., 2009; Jancik et al., 2010]. Recently, other approximations, more accurate than JFA, have been presented with success [Miguel et al., 2014]. Given the importance of JFA in this Thesis, it will be presented separately in Section 2.11.

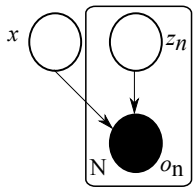
In TMFA, an observation at time n generated by component k can be expressed as

$$\mathbf{o}_n = \mu_k + \mathbf{W}_k\mathbf{x} + \epsilon_k. \tag{2.28}$$

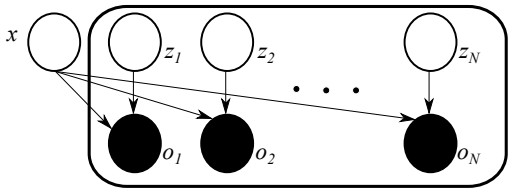
We can see the graphical model in Figure 2.12a. In Figure 2.12b we can also see its expanded version. Unlike MFA, whose graphical model is shown in Figure 2.7b, where each data point has an associated hidden variable, TMFA has a hidden variable which is common to a set of N data points.

2.10.1 Exact Calculation

Observe the difference between the graphical models of TMFA in Figure 2.12, and the graphical model of MFA in Figure 2.7b. In TMFA, the whole sequence of N observed variables, $\mathbf{O} = \mathbf{o}_1...\mathbf{o}_N$, depends on the same hidden variable, $\mathbf{x}$, and hence, they are not independent of each other unless $\mathbf{x}$ is known. Thus we have to model the whole sequence of observations together and capture the correlations of the observed variables at different times n. This is achieved by modeling the concatenation of all the observed



(a) Graphical model of TMFA. The hidden variable, $\mathbf{x}$, is common to a set of N data points.



(b) Expanded graphical model of TMFA. We see that N observations depend on the same hidden variable $\mathbf{x}$.

Figure 2.12: Graphical model of TMFA in short and expanded notation.

variables as a single vector $\bar{\mathbf{O}} = [\mathbf{o}_1; ...; \mathbf{o}_N]$ [Miguel et al., 2014]. From now on, we will call *supervectors* to the concatenation of vectors. To compute the marginal distribution of $\mathbf{O}$, we have to integrate out the hidden variable, which follows a standard normal distribution, and the sequence of indicator variables $\mathbf{Z} = \mathbf{z}_1 ... \mathbf{z}_N$. Then

$$\begin{aligned} P(\bar{\mathbf{O}}) &= \sum_{s_1=1}^{K} ... \sum_{s_N=1}^{K} \int p(\bar{\mathbf{O}}, \mathbf{x}, \mathbf{z}_1 = z_{s_1}, ..., \mathbf{z}_N = z_{s_N}) d\mathbf{x} \\ &= \sum_{s_1=1}^{K} ... \sum_{s_N=1}^{K} \int \prod_n p(\mathbf{o}_n | \mathbf{x}, \mathbf{z}_n = z_{s_n}) p(\mathbf{z}_n = z_{s_n}) p(\mathbf{x}) d\mathbf{x} \\ &= \sum_{s_1=1}^{K} ... \sum_{s_N=1}^{K} \omega_{s_1} ... \omega_{s_N} \mathcal{N}(\bar{\mathbf{O}} | \bar{\mu}_s, \bar{\mathbf{\Sigma}}_s), \end{aligned} \tag{2.29}$$

where s is an integer from 1 to K^N that identifies the current sequence of Gaussian indices $s_1, ...s_N$, s_n indicates the active Gaussian component of sequence s at time n, K is the number of Gaussian components, $p(\mathbf{z}_n = z_{s_n})$ indicates the probability that Gaussian component s_n is active at time n, also indicated as $p(z_{s_n} = 1)$, $\bar{\mu}_s$ is the concatenation of the means of Gaussian components indicated by s, $\bar{\mathbf{\Sigma}}_s$ is a DNxDN matrix with the following structure

$$\bar{\mathbf{\Sigma}}_s = \bar{\mathbf{W}}_s \bar{\mathbf{W}}_s^{\intercal} + \bar{\mathbf{\Psi}}_s, \tag{2.30}$$

where $\bar{\mathbf{W}}_s = \mathbf{W}_{s_1}^{\intercal} ... \mathbf{W}_{s_N}^{\intercal}$ is the concatenation of matrix subspaces corresponding to the index combination s, and $\bar{\mathbf{\Psi}}_s$ is a very large diagonal matrix whose nth diagonal block is $\mathbf{\Psi}_{s_n}$, and the off-diagonal blocks are set to $\mathbf{0}$. In order to keep notation uncluttered, we express it as

$$P(\bar{\mathbf{O}}) = \sum_{s=1}^{K^N} \bar{\omega}_s \mathcal{N}(\bar{\mathbf{O}}; \bar{\mu}_s, \bar{\mathbf{\Sigma}}_s), \tag{2.31}$$

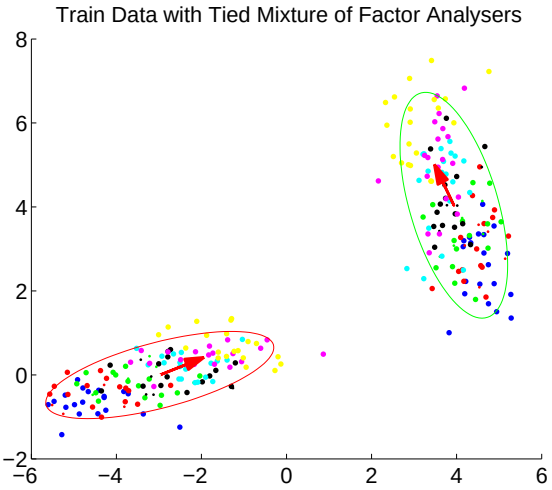


Figure 2.13: Example of TMFA - TMFA modeling two clusters of data. This models is able to learn correlations among different Gaussians. See the subspaces spanned by the arrows. In the lower part of the arrows we find dark blue, red, green and black points in the two components, while on the top part we find yellow, cyan, pink, and another set of black points in the two components.

where now s goes through all possible K^N permutations of Gaussian indices, and $\bar{\omega}_s$ is the product of Gaussian weights corresponding to the combination s. As per eq. (2.31), the model is equivalent to a mixture of factor analyzers with K^N components.

The model parameters must be computed with an iterative process, like the EM algorithm presented in Section A.7.1 of Appendix A.

2.10.2 Example

In Figure 2.13, we have an example of TMFA, with $J = 5$, $N_j = 10$, $K = 2$, and we have run 10 iterations of the EM algorithm. We have artificially created 10 data points for each file j, where the points of each file are printed in a different color. The data are 2 dimensional, whereas the dimension of the hidden variable, $\mathbf{x}$, is 1. Unlike MFA, where we obtained a subspace for each component pointing in the directions of maximum variability, and the model only captures correlations among points generated by the same Gaussian component, in TMFA, the model learns correlations among points generated by different Gaussian components. See for example the blue points in the Figure. They are on the bottom of the arrow in both Gaussians. Or the yellow

points, which are on top of the arrow. The hidden variable in the blue points is always low, while in the yellow points is always high. The reason is that the model has learnt the correlations among points generated by different Gaussian components. The correlation among points of Gaussian component $k = 1$ will be given by $\mathbf{W}_1\mathbf{W}_1^\intercal$, whereas the correlations among points of Gaussian components $k = 1$ and $k = 2$ will be given by $\mathbf{W}_1\mathbf{W}_2^\intercal$. Now, the subspaces can not be considered only locally, and they must be considered globally. We can see that, unlike MFA, the sign of the subspace is important. As we said at the beginning of this section, only changing the sign of all the subspaces simultaneously would give an equivalent solution.

2.11 Joint Factor Analysis

One interesting approximation to avoid the full computation of the K^N Gaussian combinations over a file was introduced in Kenny and Dumouchel [2004] with the name of JFA. It was designed as a MAP adaptation that included two hidden variables to model the speaker and one hidden variable to model the channel, but addressed from a FA perspective.

The main simplification made by JFA is that the Gaussian posteriors are known in advance. They are given by a previously trained GMM, the UBM. This means that we know beforehand which Gaussian component generated each data point. So, we reduce the K^N possible permutations in the likelihood function of TMFA in eq. 2.31 to only one. However, the assignments are soft. Hence it is assumed that all the Gaussian components partially contribute to the generation of each data point, and not only one.

Suppose that our D-dimension vector of N observations, $\mathbf{O} = \mathbf{o}_1...\mathbf{o}_N$ can be modeled by a GMM distribution with K components, with weights $\omega_1, ..., \omega_K$, means $\mu_1, ..., \mu_K$, and covariance matrices $\mathbf{\Sigma}_1, ..., \mathbf{\Sigma}_K$. This distribution is the UBM. The means can be concatenated to build a single supervector, $\mu = [\mu_1^\intercal, ..., \mu_K^\intercal]^\intercal$, with dimension KD. The analogous covariance supermatrix would be block-diagonal with $\mathbf{0}s$ in the off-diagonal entries,

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & & \cdots & \\ & \mathbf{\Sigma}_2 & & \vdots \\ \vdots & & \ddots & \\ & \cdots & & \mathbf{\Sigma}_K \end{bmatrix}. \tag{2.32}$$

In JFA, we suppose that the concatenated vector of means, $\mathbf{M}$, can be factorized as follows, if we know the hidden variables $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{z}$,

$$\mathbf{M} = \mu + \mathbf{V}\mathbf{y} + \mathbf{U}\mathbf{x} + \mathbf{d}\mathbf{z}, \tag{2.33}$$

where $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{z}$, are standard normal distributed of dimension M_y, M_x and KD, respectively, $\mathbf{V}$ and $\mathbf{U}$ are low rank matrices of dimensions $KD\text{x}M_y$ and $KD\text{x}M_x$, respectively, and $\mathbf{d}$ is a diagonal matrix of size $KD\text{x}KD$. The term $\mathbf{Vy}$ was introduced to model speaker information keeping the properties of eigenvoice MAP, which allows fast adaptation to speakers; the term $\mathbf{dz}$ was introduced to model speaker information not modeled with the term $\mathbf{Vy}$, and it conserves the properties of classical MAP [Gauvain and Lee, 1994]; and the term $\mathbf{Ux}$ was introduced to model channel information keeping the properties of eigenchannel MAP. The matrices $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{d}$ are also known as *factor loading matrices*, while the hidden variables are also known simply as *factors*, in this case *speaker* and *channel factors*.

One important difference between eigenchannel MAP and JFA is that, in the first, the utterances of a given speaker are treated as statistically independent, whereas in the second, they are not independent. JFA computes the speaker and channel subspaces jointly, whereas in eigenchannel MAP, the speaker model (including speaker subspace if it exists), and the channel subspace was calculated over the speaker model. In this Thesis, we will use JFA for LID, with only one subspace to model the channel. Hence, it can also be considered to be an eigenchannel model, and in the first investigations the eigenchannel nomenclature was adopted [Castaldo et al., 2007c; Hubeika et al., 2008; Matejka et al., 2008], but the name of JFA has eventually been preferred by the community, probably due to the influence of speaker recognition investigations. Thus, from now on we will only focus on the case with one hidden factor where the mean supervector given $\mathbf{x}$ is expressed as

$$\mathbf{M} = \mu + \mathbf{U}\mathbf{x}. \tag{2.34}$$

As $\mathbf{x}$ will be different for each utterance, we can say in practice that each utterance is generated by a different GMM.

In the case of LID (it could be applied straight forward to any other classification problem by changing the labels of the languages by the labels of the classes of the

corresponding problem), each language is modeled by a different JFA model. The supervector of means for language l, $\mathbf{M}_l(j)$, is

$$\mathbf{M}_l = \mu + \frac{\mathbf{F}_l - \mu\mathbf{N}_l}{(r + \mathbf{N}_l)} + \mathbf{U}\mathbf{x} = \mathbf{t}_l + \mathbf{U}\mathbf{x}, \tag{2.35}$$

where r is the relevance factor which controls the weight that the UBM and the new data have on the model, and

$$\mathbf{t}_l = \mu + \frac{\mathbf{F}_l - \mu\mathbf{N}_l}{(r + \mathbf{N}_l)}. \tag{2.36}$$

In practice, $\mathbf{x}$ will be different for each utterance j, and is equal to the mean of its posterior distribution given the observed data, as given by eq. (A.71) of Appendix A. Therefore, $\mathbf{M}_l$ is also different for each utterance, as stated above. We introduce now some statistics calculated over the UBM. First, the vector of zeroth order statistics for the N_j observations of utterance j, $\mathbf{O}(j) = \mathbf{o}_1(j)...\mathbf{o}_{N_j}(j)$, for component k, is defined as

$$N_k(j) = \sum_{n=1}^{N_j} \gamma_n^j(z_k), \tag{2.37}$$

where $\gamma_n^j(z_k)$ was defined in A.22. The vector of first order statistics for component k is defined as

$$\mathbf{F}_k(j) = \sum_{n=1}^{N_j} \gamma_n^j(z_k)\mathbf{o}_n(j). \tag{2.38}$$

Finally, the vector of second order statistics for component k is defined as

$$\mathbf{S}_k(j) = \sum_{n=1}^{N_j} \gamma_n^j(z_k)\mathbf{o}_n(j)\mathbf{o}_n(j)^\intercal. \tag{2.39}$$

These statistics can be expressed in the form of supervectors to include all Gaussian components as

$$\mathbf{N}(j) = [N_1(j) \cdot \overbrace{[11...11]'}^{D}; N_2(j) \cdot \overbrace{[11...11]'}^{D}; ...; N_K(j) \cdot \overbrace{[11...11]'}^{D}], \tag{2.40}$$

$$\mathbf{F}(j) = [\mathbf{F}_1(j); \mathbf{F}_2(j); ...; \mathbf{F}_K(j)]; \tag{2.41}$$

$$\mathbf{S}(j) = \begin{bmatrix} \mathbf{S}_1(j) & & \cdots & \\ & S_2(j) & & \\ \vdots & & \ddots & \vdots \\ & \cdots & & \mathbf{S}_K(j) \end{bmatrix}. \tag{2.42}$$

Then, the global zeroth, first, and second order statistics for each language, for component k, are

$$N_{lk} = \sum_{j \in l} N_k(j), \tag{2.43}$$

$$\mathbf{F}_{lk} = \sum_{j \in l} \mathbf{F}_k(j), \tag{2.44}$$

$$\mathbf{S}_{lk} = \sum_{j \in l} \mathbf{S}_k(j), \tag{2.45}$$

and to indicate the global supervector of zeroth, first, and second order statistics for language l including all Gaussian components, we use the following notation,

$$\mathbf{N}_l = [N_{l1} \cdot \overbrace{[11...11]'}^{D}; N_{l2} \cdot \overbrace{[11...11]'}^{D}; ...; N_{lK} \cdot \overbrace{[11...11]'}^{D}], \tag{2.46}$$

$$\mathbf{F}_l = [\mathbf{F}_{l1}; \mathbf{F}_{l2}; ...; \mathbf{F}_{lK}]; \tag{2.47}$$

$$\mathbf{S}_l = \begin{bmatrix} \mathbf{S}_{l1} & & \cdots & \\ & \mathbf{S}_{l2} & & \vdots \\ \vdots & & \ddots & \\ & \cdots & & \mathbf{S}_{lK} \end{bmatrix}. \tag{2.48}$$

Also, we define the zeroth, first, and second statistics of the whole database as

$$N_k = \sum_l N_{lk}, \quad \mathbf{F}_k = \sum_l \mathbf{F}_{lk}, \quad \mathbf{S}_k = \sum_l \mathbf{S}_{lk}. \tag{2.49}$$

It is important to remark that the fixed alignments given by γ are previously computed over the UBM, and we use these alignments to compute the statistics. Then, we can see the UBM as a block that transforms features into sufficient statistics, and these statistics are the input to the JFA training process. This flow can be schematically seen in Figure 2.14.

In Figure 2.15, we present a schematic view of the model used in LID. For simplicity we only show one component of the mixture. The non-compensated mean of the class l is obtained via relevance MAP from the UBM, $\mathbf{t}_l$. To compensate for the channel, the vector $\mathbf{Ux}(j)$ is added to obtain the final mean of the class, $\mathbf{M}_l(j)$, for utterance j. In Figure 2.16, an artificial example of 3 classes is shown. In this case, we also

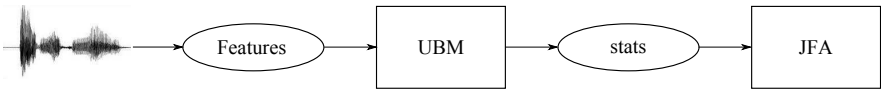


Figure 2.14: JFA training process flow - Features are used to calculate sufficient statistics over the UBM. JFA model parameters are computed with supervectors of sufficient statistics.

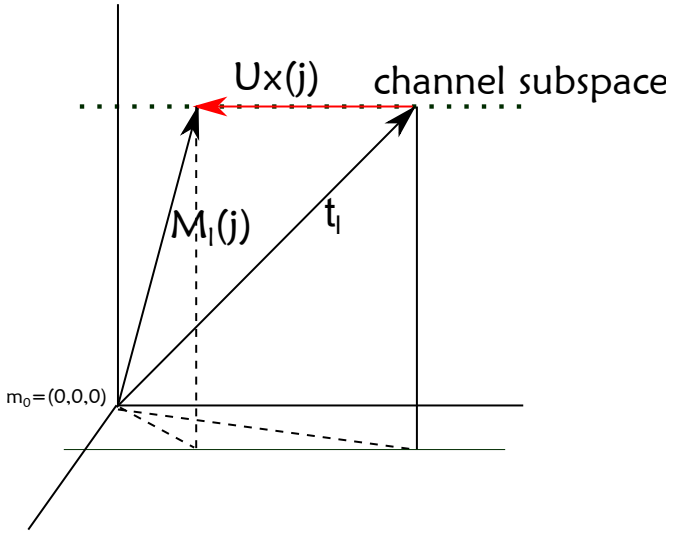


Figure 2.15: JFA Model - The mean of the model, $M_l(j)$, is a sum of the UBM mean, the MAP adaptation for the class given by $\mathbf{t}_l$, and the channel compensation factor, $\mathbf{Ux}(j)$.

have for simplicity a single component of 3 dimensions. The black, blue, and cyan points are the samples drawn from 3 different classes for $\mathbf{x} = 0$. The yellow, red, and magenta points are the same samples for different values of $\mathbf{x}$. The green bars show the direction of the channel subspace. In our example, the channel variability is confined to a 1-dimension subspace. In Figure 2.17, 2 classes of a 2-dimension JFA model with 1 Gaussian component can be seen, with a 1-dimension channel variability subspace aligned with the vertical dimension. The example shows that the model is adapted differently in different utterances. In Section A.8 of Appendix A, we derive the EM algorithm for JFA, which includes the traditional model estimation by differentiation of the objective function, and also a minimum divergence (MD) step, an alternative method to maximize the lower bound of the log-likelihood function, which requires a change of variable to avoid an overparametrization of the model.

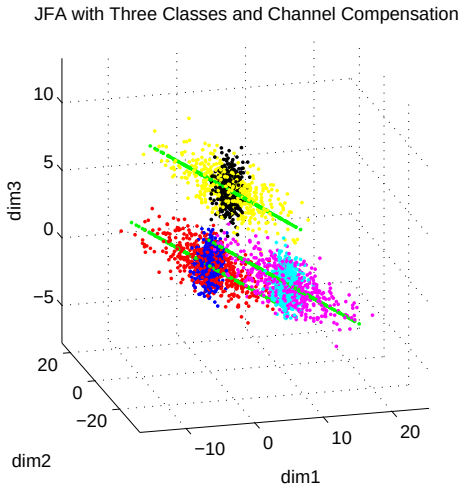


Figure 2.16: Artificial example of JFA - The blue points are drawn from the model of class 1 with mean $(0, 0, 0)$, the black points are drawn from the model of class 2 with mean $(2, -4, 6)$, the cyan points are drawn from the model of class 3 with mean $(10, -7, -1)$. For the 3 cases, the models are Gaussian distributions with covariance matrix equal to the identity matrix. The red, yellow and magenta clouds are the same distributions affected by channel variability $\mathbf{x} \neq 0$. The green bar is the channel subspace, and it indicates where the channel factor can lie.

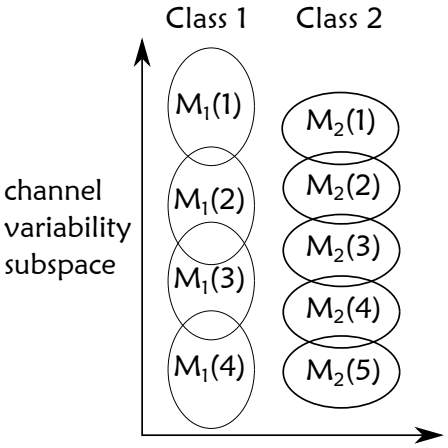


Figure 2.17: JFA in 2D - 2 classes of a 2-dimension JFA model with 1-dimension channel variability restricted to the vertical dimension.

2.12 Total Variability Subspace: *i-Vectors*

An *i-Vector* is a fixed-length low-dimension representation of a speech utterance [Dehak et al., 2011a]. The first application of *i-Vectors* was in the field of speaker recognition

Figure 2.18: *i-Vector* training process flow - Features are used to calculate sufficient statistics over the UBM. JFA model parameters are computed with supervectors of sufficient statistics. Unlike the chart in Figure 2.14, where JFA was our final goal, now JFA is used as a front-end to extract *i-Vectors*.

[Dehak et al., 2011a]. The first use in LID was presented by us [Martínez et al., 2011b], and simultaneously by Dehak et al. [2011b], in Interspeech 2011. The first use in the processing of pathological voices was also introduced by us in Martínez et al. [2013b,c]. In this section, we present the maths behind the *i-Vector* approach.

In Dehak [2009], it was shown that speaker information was not completely removed from the channel subspace in a JFA model with speaker and channel factors. Motivated by this finding, they redesigned a JFA model with a single factor, which did not collect a specific aspect of speech, but all possible sources of variability in the data. This was the reason to name $\mathbf{T}$ as the *total variability subspace*. It is the post-process of *i-Vectors* with the appropriate database and file labeling what makes them useful for one task or another, such as LID or intelligibility assessment. They are versatile features that can be used for many speech related tasks. The mean supervector, $\mathbf{M}$ of this model is expressed as

$$\mathbf{M} = \mu + \mathbf{T}\mathbf{i}, \tag{2.50}$$

where μ is the mean supervector of the UBM, $\mathbf{i}$ is a standard normal distributed hidden variable tied to all the data points within a file, and $\mathbf{T}$ is a low-rank matrix spanning the subspace where the hidden variable, $\mathbf{i}$, lies. The expected value of the posterior distribution of the hidden variable, $\mathbf{i}$, is the so-called *i-Vector*. We want to remark that one *i-Vector* represents a single file, has low dimensionality (typically in the order of hundreds), and it contains all the variability in the data. Hence, JFA can be seen here as a dimensionality reduction technique that receives as input all features within a file and returns a single low-dimension vector. In Figure 2.18, we show the flowchart from features to *i-Vectors*. It is interesting to see how we have returned to the initial objective pursued by PCA, but with a much more sophisticated method. The EM training of the model for *i-Vector* extraction can be seen in Section A.9 of Appendix A.

Part II

Language Identification

3

State of the Art

Contents

We revise the most important literature in the field of LID. The structure of this chapter is chronological. We have divided the time line in periods in which a particular technology was predominant. We have devised the following periods:

- Period 1973-1992: the Initial Attempts.
- Period 1992-2001: the Phonotactic Era.
- Period 2002-2006: the GMM and SVM Establishment.
- Period 2007-2010: the Factor Analysis Era.
- Period 2011-2014: the *i-Vector* Era.
- Period 2014-: the Deep Era.

Additionally, we have reflected all the state of the art, from the initial days of LID to date, in a diagram 3.1[1]. In this diagram, each box represents a single work, in most cases from one paper or one journal, but sometimes a box can include more than one paper or journal. The works appear in chronological order, from up to down. Some works extend for two years, and then the box is longer. There is also a color code. Blue is used for acoustic or acoustic-phonetic systems, red is used for phonotactic or syllabotactic systems, yellow is used for prosodic systems, orange is used for lexical or LVCSR systems, gray is used for raw waveform systems, and white is reserved for the calibration work of Niko Brümmer. Systems that capture the rhythm in a syllable-based basis are considered to be prosodic. A syllabic system would be that that extracts nonprosodic information at the syllable level. Boxes from hybrid works including two or three systems are painted in the two or three corresponding colors. In the boxes, the list of authors is complete if there is one or two authors. If there are more than two authors, only the name of the first appears. Finally, the arrows indicate influence between two works, and consequently, always go from older to newer investigations.

[1]We have tried to reflect the most important and relevant works in the history of LID until June 2014, but given the overwhelming amount of publications from the 1970s, specially in the last 15 years, if some reader notes that some important work is missing, we will appreciate that he/she contacts the author. We would like to keep this historical review and the diagram of Figure 3.1 as complete as possible.

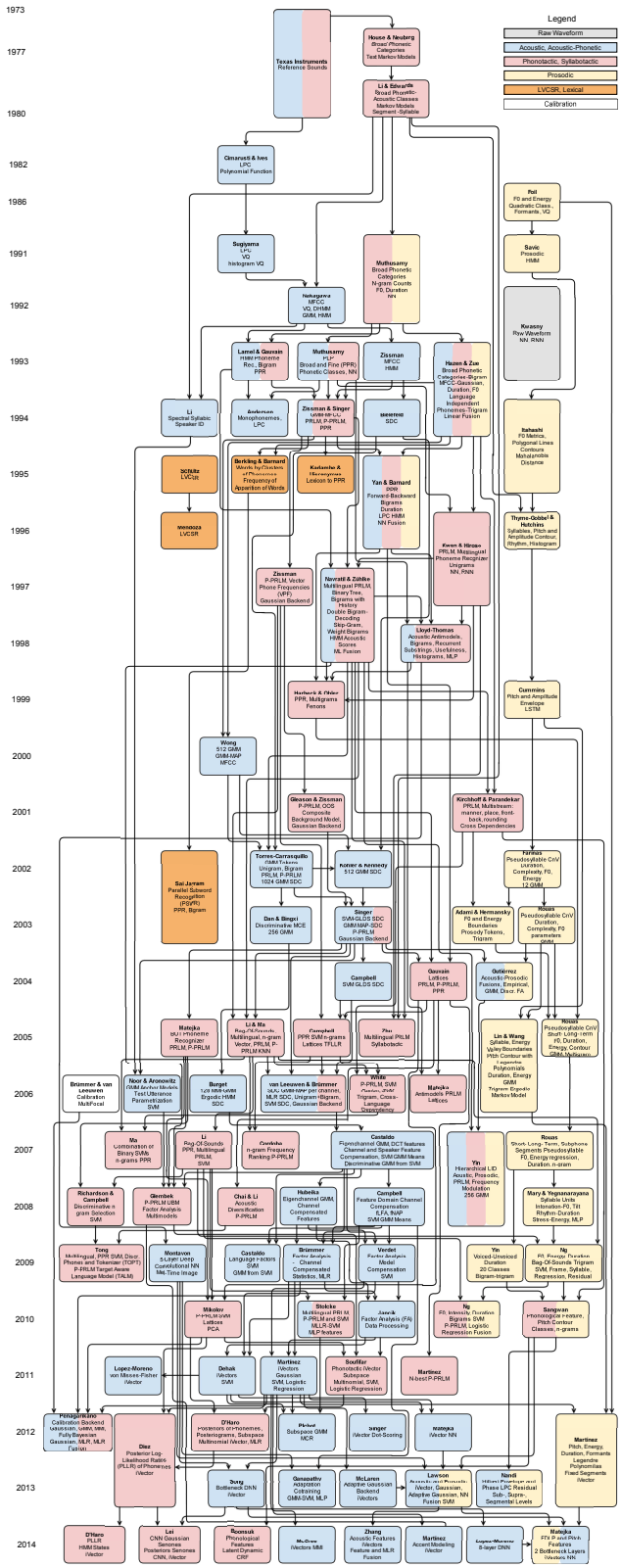


Figure 3.1: State of the art diagram - In this diagram we collect the most relevant works in the history of LID in chronological order from top to bottom, where each box represents a work. The color coding shows the category to which the major contribution or contributions of that work belong to.

3.1 Period 1973-1992: the Initial Attempts

Several features and methods were proposed in this initial experiments in the field of LID. The main exploited idea was that the speech can be segmented in broad phonetic units that vary among languages. Thus, approaches using "reference" sounds, syllable and segment-based information like broad phonetic categories or phones, pitch, formant, spectral, or raw waveform information, were investigated. Also, different linear and quadratic classifiers were tried, built with different techniques like polynomials, vector quantization (VQ), Markov model, hidden Markov model (HMM), GMM, or artificial neural network (ANN). These systems were the first attempts of acoustic, prosodic, and phonotactic LID. The main drawbacks at this time were that some of the experiments were confidential and no deeply explained, and the lack of uniformity in the databases, which made impossible to do fair comparisons. Another review of this period can be found in Muthusamy [1992].

The first written reports in the history of LID were carried out by Texas Instruments from 1974 to 1980 [Leonard and Doddington, 1974, 1975; Leonard, 1980; Muthusamy, 1992]. Their motivation was that languages differ by the frequency of occurrence of certain sounds or sound sequences. They extracted reference sounds, and the likelihood of new data over these references was computed. This can be considered as the beginning of the acoustic and phonotactic approaches, because they exploited the similarity of different spectral references, and the order in which they were found.

In 1977, House and Neuburg made an experiment with text transcriptions [House and Neuburg, 1977]. They segmented the speech into broad phonetic categories, and assumed that the sequence of segments obey a Markov model, whose parameters can be estimated for a given language with sufficient amount of data. No speech was involved in this experiment, but it was encouraging for the speech community as a motivation for further research in this direction.

In 1980, Li and Edwards made one of the earliest efforts to develop statistical inference techniques to discriminate among languages [Li and Edwards, 1980]. Following the suggestions of House and Neuburg [House and Neuburg, 1977], they employed zeroth, first, and second order Markov models to classify 6 broad segmental acoustic-phonetic classes. They defined two systems, one based on segments, and another based on syl-

lables. The 6 classes were obtained based on acoustic information, and with Markov models phonotactic information was also considered in the classification.

An acoustic approach was proposed in 1982 by Cimarusti and Ives [Cimarusti and Ives, 1982], who developed a polynomial decision function with 100 features extracted from linear predictive coding (LPC) analysis.

Research on prosodic features was initiated by Foil in 1986 [Foil, 1986]. He introduced rhythm and intonation through pitch and energy contours. He used a quadratic classifier. He also wanted to capture the frequency of apparition of sounds, and he used formant frequency values and locations to represent them. As classifier, he used a VQ distortion measure. The audio was recorded from radio in noisy conditions.

Sugiyama in 1991 proposed two algorithms based on VQ using acoustic features obtained from LPC [Sugiyama, 1991]. In the first algorithm, he created a codebook for each target language, and during test, he accumulated the distances of the test features to each codebook. The language with the lowest accumulated distance was selected. In the second approach, the difference was that he used a histogram VQ with a universal codebook.

Neural networks were used for the first time in the works of Muthusamy in 1991 and 1992 [Muthusamy et al., 1991; Muthusamy and Cole, 1991; Muthusamy, 1992]. His works were based on the assumption that each language has a unique acoustic signature that can be characterized by the waveform, phonetic and prosodic information of speech. First, he used a neural network to segment the speech into 7 broad phonetic categories. Second, he computed prosodic and frequency of occurrence features within those categories for each utterance. Third, the previous features were the input features to a second neural network that classified the spoken language. This can be considered the first work including acoustic, phonetic, and prosodic information. However, the three sources of information were used to segment the data, but for the language classification step, only phonetic, phonotactic, and prosodic information were considered.

Nakagawa made an interesting comparison of acoustic classifiers in 1992 [Nakagawa et al., 1992]. He compared a VQ and a discrete HMM, as discrete classifiers, and continuous HMM, and GMM, as continuous classifiers. It is notable that he was the first researcher that used a GMM for LID. GMM was considered to be the output of an HMM with one state. As features, he used 10 mel-frequency cepstrum coefficients

(MFCCs) [Davis and Mermelstein, 1980]. The use of HMMs was also proposed by Savic in 1991 [Savic et al., 1991] for prosodic features, but he did not report results.

The last study in this period was the one of Kwasny in 1992 [Kwasny et al., 1992]. The raw input signal filtered at four bands was the input to a neural network. In 1993, Wu [Wu et al., 1993], and also in 1993, Kwasny [Kwasny et al., 1993], made some further work based on this. They applied 5 band-pass filters and the outputs were passed to a recurrent neural network (RNN).

3.2 Period 1992-2001: The Phonotactic Era

This period began with the first attempt to create a common database to ease comparison among different works, the OGI multi-language telephone speech corpus (OGI_TS) [Muthusamy and Cole, 1992]. Initially, ten languages composed the corpus: English, Farsi, French, German, Japanese, Korean, Mandarin, Spanish, Tamil, and Vietnamese. A few years later, Hindi was added into the corpus. In 1996, NIST organized an LRE to compare LID system performance under the same conditions at different research groups [National Institute of Standards and Technology (NIST), 1996].

An this time we can see the birth of very successful ideas used still today, like parallel phone recognition (PPR), phoneme recognition followed by language model (PRLM), or parallel phoneme recognition followed by language model (P-PRLM). Most approaches were based on HMMs with GMMs to tokenize data. Usually, acoustic scores were given by HMMs, and phonotactics were captured by n-gram models or directly through the state transitions of HMMs. For acoustic systems, it was observed that GMMs performed similar than HMMs, so the dynamics of HMM were useless to model acoustics. Important contributions like binary tree modeling of phonotactics in PRLM systems or multilingual phoneme recognizers to cover a wide range of phonemes were from this period. Using n-grams as feature vectors instead of building a language model (LM) was also proposed in these years, and this approach was integrated in most phonotactic systems. Also LVCSR was introduced for LID. In this period, the phonotactic systems were, in general, dominant in LID.

The prosodic works at this time were focused on modeling the pitch and amplitude contour to capture intonation and stress, and on modeling duration to capture rhythm. Usually, these features were combined with simple classifiers.

The first results with OGI_TS database were published still in 1992 by Muthusamy [Muthusamy and Cole, 1992]. He extended his previous work in [Muthusamy et al., 1992] to a 10 language task.

In 1993, Muthusamy compared three approaches based on acoustic features, broad category segmentation, and fine phonetic classification [Muthusamy et al., 1993]. For the acoustic approach, he extracted 8 perceptual linear prediction (PLP) coefficients [Hermansky, 1990] and averaged them with 7 contiguous regions spanning 171 ms. In the second approach, 7 broad phonetic categories were created from PLP for each language, and bigram statistics were created. In the fine phonetic classification, he first recognized phonemes for each language and then obtained unigram and bigram statistics. For classification he used a fully-connected feed-forward neural network in the three cases. The idea resembles PPR but with a final neural network to classify the n-gram features.

Zissman performed another interesting work in 1993 [Zissman, 1993]. He used ergodic, continuous-observation, HMMs fed with cepstrum vectors. The output observation distributions were GMMs. The main conclusion was that the 8 states and 1 state HMM performed very similar, what means that the sequential modeling capabilities of HMMs were not exploited. He also applied channel equalization including spectral norm removal and relative spectral (RASTA) [Hermansky and Morgan, 1994].

Hazen and Zue in 1993 proposed for the first time the idea of PRLM, and performed the first fusion at score level [Hazen and Zue, 1993]. They also based his work on the segmentation ideas of House and Neuburg [House and Neuburg, 1977]. First they extracted meaningful segments, and as Muthusamy, these were based on phonemes. However, as it was difficult to label data for each language, they used an English phoneme recognizer trained on TIMIT database to segment all the languages, assuming that different languages share sound categories. Then, they clustered the phones into 23 broad phonetic classes in a hierarchical manner. Finally, they built three classifiers: acoustic, prosodic, and phonotactic. For the phonotactic, a bigram model obtained the best performance. The prosodic information was included by modeling the number of frames within each of the broad phonetic categories, and quantizing and modeling the values of the fundamental frequency. The acoustic information was represented by 14 MFCCs and its derivatives, and each category was modeled with a full covariance Gaussian distribution. Combining the scores of the three systems, they obtained better

results. In 1994, they added channel normalization, a GMM to model duration, and they used a language independent phoneme recognizer trained on new labeled target data instead of the one trained on TIMIT [Hazen and Zue, 1994]. They worked directly with phonemes instead of grouping them into broad phonetic classes.

The work of Lamel and Gauvain in 1993 also performed phoneme-based segmentation, and also followed a PPR approach [Lamel and Gauvain, 1993]. The evaluation was made by the acoustic scores of the ergodic 3-state continuous density hidden Markov model (CDHMM) with GMM output distributions, exploiting the phonotactics with a bigram model. Probably, this can be considered to be the first PPR system.

Li in 1994 [Li, 1994] presented a system based on spectral features taken at the syllabic rate [Li, 1994]. The selection of the language was made based on a speaker identification system with the nearest neighbor algorithm. The language of the training speaker closest to the test one was hypothesized as the language of the test utterance.

The work of Zissman and Singer in 1994 compared 4 very successful approaches that can be considered to be the roots of the systems used today [Zissman and Singer, 1994]. The first was a GMM with spectral information (12 MFCCs, first derivatives, and energy) for every target language. Actually, two different GMMs were trained for each language, one for the cepstra and one for the derivatives. The other three were variations of the phoneme segmentation followed by n-gram modeling. These techniques are still used today with the original names given by these authors. The first of these three was PRLM, where we only need one language with labels, and n-grams models are trained for each target language. In the second approach, several languages, not necessarily target languages, are used to train different phoneme recognizers. Then, they are set in parallel to build a P-PRLM. The idea is to join results of several PRLM systems. The third, also applied in Lamel and Gauvain [1993], was called PPR, and requires labeled data of all the target languages to build a phoneme recognizer for each one. The n-gram is normally integrated in the recognizer. This is the most difficult to implement for the demanding data requirements.

Andersen introduced monophones in 1994 [Andersen et al., 1994], sets of phones which are unique to a language because they are very distinguishable from phones of other languages.

Bielefeld introduced the shifted delta cepstrum (SDC) in 1994 [Bielefeld, 1994]. At that time, they did not succeed too much, but some years later SDC would become the

state of the art features for LID until today.

Itahashi in 1994 modeled F0 contours with polygonal lines, and derived pitch-related measures [Itahashi et al., 1994]. PCA was applied to keep only important variability, and the classification was performed with a discriminant analysis. In 1995, he made decisions based on minimal Mahalanobis distance [Itahashi and Du, 1995].

Yan and Barnard in 1995 and 1996 investigated a PPR system deeper [Yan and Barnard, 1995a,b; Yan et al., 1996]. Their major contribution was the use of forward and backward bigrams, which contain information about past and future phonemes. The output scores of each LM contained acoustic, language, and prosodic information, and these were merged with a neural network.

In 1995, there were three attempts to incorporate lexical information to LID systems. Kadambe and Hieronymus investigated the addition of a lexical module to PPR system [Kadambe and Hieronymus, 1995]. Sequences of phonemes recognized by the phone recognizer were mapped to words in an automatic way. They created language-dependent lexicons with thousands of words. Berkling and Barnard obtained categories from phonemes recognized by a phoneme recognizer and created words by concatenating several categories [Berkling and Barnard, 1995]. The frequency of apparition of these sequences were the input to a neural network. Schultz incorporated higher level of knowledge to an LID system [Schultz et al., 1995] . She compared 5 systems and introduced LVCSR for the first time in LID. The first was based on the likelihood obtained by a phone recognizer for each language; the second was based on phone recognition followed by a bigram model; the third included a word recognizer; the fourth included a word recognizer with a bigram model integrated; and the fifth was a word recognizer with bigram model integrated followed by a bigram to model the resulting sentence. Later in 1996, Mendoza built an LID system with an LVCSR system for each language [Mendoza et al., 1996]. The decision was based on the probability of the output word sequence of the recognizer of each language.

In 1996, Thyme-Gobbel and Hutchins presented a prosodic system using running averages, deltas, standard deviations, and autocorrelations, in a syllable-by-syllable basis [Thymé-Gobbel and Hutchins, 1996]. In total, 224 features divided into eight descriptive classes: pitch contour, differential pitch, size, differential size, amplitude, differential amplitude, rhythm, and phrase location. The scores were calculated by histogram modeling, what allowed them to compute log-likelihood ratios.

Kwan and Hirose used an RNN and a multilayer perceptron (MLP) to classify unigrams obtained with a multilingual phoneme recognizer [Kwan and Hirose, 1997].

Navratil and Zühlke made important contributions to the field in two works in 1997 [Navratil and Zühlke, 1997a,b]. In the first, they proposed to use an HMM to decode phonemes, but with several bigrams during the Viterbi decoding, as if several phoneme recognizers were run in parallel, similarly to P-PRLM. Once the phonemes were decoded for each grammar, another bigram was run for each target language. This was called double bigram-decoding. The phoneme recognizer was multilingual. Also, they introduced the concept of "skip-gram", a bigram model trained on the current phoneme and the one before the previous, to overcome the problem of scarcity of data for trigram training. Another contribution of this first work was in the classifier, and they proposed to weight more the bigrams that differ most among languages. In the second work, they modeled bigrams taking into account information about the sequence of phonemes of the two past phonemes instead of just one, but reducing the inventory of possible pairs based on the frequency of apparition. A second and very important proposal of this work was to model phonotactics with a binary tree.

In 1997, Zismman introduced some modifications to his baseline system, based on P-PRLM with interpolated n-grams and phones including short or long duration labels [Zissman and Street, 1997]. First, he presented the concept of Gaussian backend, a Gaussian distribution for each language over the output scores given by the n-grams coming from each phoneme recognizer. Usually, that Gaussian distribution had diagonal covariance shared among languages. This backend allowed adding languages to the system for which only development data were available, without the need of training data. Also, he proposed to treat an n-gram count as a vector of phone frequencies (VPF), and to model them with a Gaussian distribution, which is characterized by a mean and a diagonal covariance. This approach made possible to add an extra parameter with respect to the multinomial distribution normally used with n-grams, and to treat each phone independently. Both approaches were very important in future works.

One interesting idea in the work of Lloyd-Thomas in 1998 was the use of anti-models for score normalization [Lloyd-Thomas et al., 1998]. They are acoustic scores from models trained on languages different from the target language. He combined them with other phonotactic approaches using MLPs.

The work of Navratil in 1998 proposed to combine the scores obtained by a multilingual phone recognizer including a bigram model, a selection matrix to consider phone triples, and a binary decision tree to constrain up to three preceding phone, with language-dependent pronunciation models for each phone, which models phone acoustics with Gaussian probability density models [Navrátil and Zühlke, 1998]. For the fusion and classification, an MLP was used.

In 1999, Harbeck and Ohler experimented with multigrams, n-grams of variable length and at most length n, and fenons, segments automatically found, not necessarily equal to phonemes [Harbeck and Ohler, 1999]. Fenons helped in longer sentences, but multigrams did not improve results.

In 1999, Cummins investigated prosody for LID [Cummins et al., 1999]. He used pitch and amplitude envelope information. For classification, he used a long-short term memory (LSTM) model with modest results.

Wong in 2000 compared the performance of a GMM system with the performance of a GMM adapted from a UBM [Wong et al., 2000]. The input were 12 MFCCs and their first delta parameters. He used 512 components.

In 2001, Kirchhoff and Parandekar proposed to use multi-stream of features instead of only the phone sequence, where the new features were combination of phones to form meaningful groups like manner of articulation, consonantal place, vowel place, front-back, and rounding [Kirchhoff and Parandekar, 2001]. In this way, more information was extracted from the same training database, and the number of classes was reduced compared with phones. After the tokenization, n-gram models were trained like in a PRLM system. Additionally, they modeled cross-stream dependencies, by adding the current token of another class into the n-gram of a feature class as part of the history.

In 2001, Gleason and Zissman presented a way to model out-of-set languages, that is, non target languages that are in the evaluation test [Gleason and Zissman, 2001]. They worked with a P-PRLM system. If there were not enough data to train a model for such languages, they proposed to build a universal model with all unlabeled data and treat this model as another target language. They called it composite background (CBG) model. However, they showed that just the addition of a Gaussian backend over the scores of target languages was a better solution than the CBG.

3.3 Period 2002-2006: The GMM and SVM Establishment

This period established the basis for modern LID. First, acoustic systems improved with the success of GMMs, and the gap between phonotactic and acoustic systems was drastically reduced. Second, the use of support vector machines (SVM) was extensive for both acoustic and phonotactic systems for discriminative training. In the case of acoustic systems, the generalized linear discriminant sequence (GLDS) kernel was used, and in the case of phonotactic systems, Zissman's idea of 1997 about VPF was continued. The introduction of lattices instead of the best-scoring phoneme sequence was a very important contribution to phonotactic systems. The successful Brno University of Technology (BUT) phoneme recognizer is from this period [Schwarz, 2008], and the importance of calibration was revealed by Brümmer and van Leeuwen in 2006 [Brümmer and van Leeuwen, 2006]. This work would change the way LID scores are managed by researchers of this field.

In prosodic systems, new methods to extract syllables were investigated, GMMs were extensive to model pitch and energy contours, and token-based approaches were also explored.

NIST standardized LREs and two more took place during this period, in 2003 [National Institute of Standards and Technology (NIST), 2003] and 2005 [National Institute of Standards and Technology (NIST), 2005]. LID was considered to be a consolidated research field.

Farinas presented a system with rhythm and stress features in 2002 [Farinas et al., 2002]. As rhythmic units, he used the pattern of C^nV, where n indicates the number of consonants that precede an indefinite number of vowels. He called this pattern pseudosyllable. The features were the duration of the consonant part, the duration of the vowel part, and the number of consonants or complexity. Also, he added F0 and energy. He used GMM with 12 components for classification.

In 2002, Sai Jayram proposed to substitute the phone recognizer in PPR by a sub-word modeling block named parallel sub-word recognition (PSWR) [Sai Jayram et al., 2002]. In this idea was, first 12 MFCCs were extracted and segmented along each file with a dynamic programming (DP) procedure. Once the segmentation was done, the next step was to build L clusters with similar acoustic content. The clustering was

done with a K-means algorithm. These units were treated as the phones of a PPR system, and for each one an HMM was trained for every target language. In this work only the acoustic part was used, but they suggested that an LM after the HMMs could improve the results. In 2003, he introduced two bigram language models, one during the decoding of the states with Viterbi algorithm, and one after the sub-word recognition module [Sai Jayram et al., 2003].

In 2002, Torres-Carrasquillo presented a new tokenization approach [Torres-Carrasquillo et al., 2002a]. Instead of using the outputs of a phone recognizer, he proposed to use the Gaussian component index of a GMM with highest score at each frame. He trained unigram and bigram interpolated LMs with the index sequence. He compared PRLM and P-PRLM configurations.

One important work in the history of LID was presented by Torres-Carrasquillo in 2002, where he obtained similar results with an acoustic based system and with a PRLM system [Torres-Carrasquillo et al., 2002b]. He trained GMMs of up to 1024 components with cepstrum data and some special derivatives. Instead of using derivatives as they were commonly used, he used SDCs. SDCs are the derivatives of future frames added to the current MFCCs. They were already tried for LID by Bielefield in 1994 [Bielefeld, 1994], but at that time they did not succeed so much. In this way, long-term information was considered. He used a 10-1-3-3 configuration for SDCs.

In 2002, Kohler and Kennedy made experiments to find the optimal configuration of SDC parameters in a GMM LID approach with 512 components [Kohler and Kennedy, 2002]. He found that the configuration 9-1-3-3 was optimal.

Singer in 2003 [Singer et al., 2003] continued the work of Torres-Carrasquillo presented in Torres-Carrasquillo et al. [2002b]. He worked with three systems. First, he added duration-dependent Gaussian backends to the output of the P-PRLM system, he included representations for silence and closures to the dictionary of phones, and he also added trigram LMs. Second, he used a 7-1-3-7 configuration for the SDC in the GMM acoustic system, he removed non-speech frames, he performed channel normalization with the RASTA technique, he trained a single background model first that was later adapted to the individual languages, and he also used a duration-dependent Gaussian backend. Also, he performed a feature warping technique proposed in [Reynolds, 2003] to map each feature vector into a channel-independent feature space, with the goal of providing additional robustness to variations in recording conditions and channels.

Additionally, he changed the gender-independent GMMs by gender-dependent ones. He presented a third system based on SVM using GLDS with an expansion into feature space using a monomial basis. An average feature space expansion vector with all languages was found, and then a standard SVM was built. LM scores for the test utterances were calculated as the inner product between the language model and the average expansion of the utterance. Also, duration-dependent Gaussian backends were used with this system. The fusion of the three subsystems was performed with another duration-dependent Gaussian backend. This was the first time that the GMM approach performed better than phonotactics.

Adami and Hermansky in 2003 created boundaries based on the changes of F0 and energy, in a similar way to the use of phones in a PRLM system [Adami and Hermansky, 2003]. He obtained 10 to 60 classes and modeled them with trigrams. This can be considered to be a "prosotactic" approach.

Dan and Bingxi in 2003 trained a GMM discriminatively based on the minimum classification error (MCE) [Dan and Bingxi, 2003]. He used 256 Gaussians.

Rouas in 2003 [Rouas et al., 2003] continued the experiment with prosodic features started by Farinas in 2002 and presented in Farinas et al. [2002]. He distinguished between rhythmic features and intonation features. First, he automatically segmented the speech into pseudosyllables in a language independent manner. Within a pseudosyllable, he extracted three parameters to capture rhythm, namely the duration of the consonant group, the duration of the vowel group, and the number of consonants, and also 6 intonation features, namely mean, standard deviation, skewness, and kurtosis of F0, F0 location, and normalized F0 bandwidth. As classifier, he used a GMM.

In 2004, Campbell published his system based on SVM with GLDS kernel [Campbell et al., 2004]. He used SVM in a one-vs-all strategy. The input features were SDC with configuration 7-1-3-7.

Gauvain introduced the use of lattices for phonotactic systems in 2004 [Gauvain et al., 2004]. The idea of a phonotactic system was to decode the most likely sequence of phonemes. However, this is a hard decision. The use of lattices relaxes this assumption and treats probabilistically all possible sequences of phonemes. The final score for each phone recognizer is calculated by summing the likelihoods over all paths in the phone lattice.

In 2004, Gutiérrez presented different fusion techniques for an LID system [Gutiérrez et al., 2004]. The first method, called empirical, was just to sum and multiply score values. As statistical techniques, he presented a GMM and a discriminant factor analysis fusion. Separately, he presented the theory of evidence to perform fusion.

In 2005, Zhu presented an LID system with multilingual phone inventory, and he showed that it performed better with a small number of phonemes, specially for longer file durations [Zhu et al., 2005]. Also, he created a syllabotactic approach, where the acoustic decoder produced syllable streams instead of phones, and n-gram models were trained on the syllable sequences. He showed that context-dependent models performed better than context-independent ones, but the differences disappeared for segments 20 s long or longer. The syllabotactic and phonotactic approaches were complementary.

In 2005, Li and Ma proposed to create bag-of-sounds, a vector of counts of n-grams [Li and Ma, 2005]. This work can be considered to be a development of the works of Zissman in 1997 [Zissman and Street, 1997]. The dimension of a bag-of-sounds could be reduced with latent semantic analysis (LSA). The classifier was a K nearest neighbors with centroids found after applying VQ for partitioning the space.

Matejka in 2005 compared the performance of several PRLM systems and P-PRLM [Matejka et al., 2005]. He used a hybrid phoneme recognizer, the BUT phoneme recognizer [Schwarz, 2008], which was made open to the community and has been very successful until today. The recognizer is already trained on Czech, Hungarian, Russian, and English. It is based on neural networks and Viterbi decoding, it does not use any LM, and it uses a temporal context known as temporal pattern (TRAP). Matejka also tuned a phoneme insertion penalty embedded in the recognizer. The temporal context was left and right and included 31 frames. In that experiment, he trained 4 phoneme recognizers in Czech, Hungarian, Polish, and Russian, using the SpeechDat-E database, and 6 phoneme recognizers in English, German, Hindi, Japanese, Mandarin, and Spanish, using OGI_TS database. He used trigram LMs.

In 2005, Rouas introduced his system to model long- and short-term prosody for LID [Rouas et al., 2005]. Based on the rhythm (isochrony) and intonation's theories of language, and on perceptual experiments, he introduced the notion of pseudo-syllable as a common segment to capture prosody in intervals of equal rhythm in all the languages. A pseudo-syllable normally gathered only a vowel and preceding consonants. Static modeling was reflected by F0 variations and durations, and as classifier, he used a GMM

for each language. Dynamic modeling was captured by approximating the difference between the F0 contour and a baseline curve obtained by linear regression in each pseudosyllable. The energy was modeled in the same way. Duration was also included. To model the prosodic variation, he used an n-multigram, which are patterns with variable length. In another work of 2005, following the C^nV segmentation of Farinas et al. [2002], he used as features duration of the consonant cluster, duration of vowel cluster, and number of consonants [Rouas, 2005]. He modeled them with 16 component GMMs.

Lin and Wang in 2005 proposed to model pitch contour to build an LID system based on prosody [Lin and Wang, 2005]. First, they segmented the pitch according to syllable or word boundaries, and if the segment was too long, they used energy valleys as markers. In each segment, the pitch contour was modeled with the first three Legendre polynomials. The features were composed of the coefficients of the polynomial approximation, together with the duration of the segment, and the difference in energy calculated within the same segment. They used only the second and third coefficients, which modeled slope and second order curves. The features were modeled with a GMM. In a posterior work in 2006, the authors substituted the GMM by a dynamic model in ergodic topology, a Markov model, where the transition probabilities were calculated by a trigram model approximated as mixture of two bigram models [Lin and Wang, 2006].

Campbell and his group at MITLL participated in the 2005 NIST LRE evaluation and presented several systems [Campbell et al., 2006]. One of them was a PPR with lattices, where the counts of phones were derived on a per segment basis and summed across all segments for each utterance. Then, they were normalized to produce probabilities on a per utterance basis, which were at the same time term-frequency log-likelihood ratio (TFLLR) weighted. The final vector of each utterance was classified with SVM.

In 2006, van Leeuwen and Brümmer presented his systems submitted to the 2005 NIST LRE [van Leeuwen and Brümmer, 2006]. First, they included a GMM system with 5 PLP coefficients and log-energy, with derivatives. In order to perform channel compensation, they trained a different GMM for each channel and sex, starting from a common UBM and applying MAP adaptation. Second, they also included a discriminative multi-class logistic regression (MLR) classifier, whose input was a supervector

obtained from a third order multinomial expansion of SDC features. Finally, one vector per utterance was obtained. Third, also a phonotactic MLR was built. For this system, a phone recognizer was trained which produced 44 phone probabilities every 15 ms. The vector stream was segmented based on where the maximum vector probability change was, and then averaged over those phone-like segments. MLR supervectors were formed by the 44 phone probabilities averaged over the segment and by the 44 x 44 bi-phone probabilities. Fourth and last, an SVM system was also included, using SDC from PLP expanded in third order monomials. A Gaussian backend was included to perform fusion and calibration of the scores.

In 2006, Burget presented results of his discriminative acoustic system based on maximum mutual information (MMI) [Burget et al., 2006]. He used SDC in 7-1-3-7 configuration. Additionally, he also added heteroscedastic linear discriminant analysis (HLDA) and ergodic HMM.

In 2006, Matejka presented antimodels for PRLM approaches [Matejka et al., 2006]. For each target language, an antimodel was a LM trained with all data from the rest of languages. In test, he subtracted the score given by the antimodel to the score of the target LM.

White in 2006 made an effort to include more discriminative parts in standard P-PRLM systems [White et al., 2006]. First, he proposed to substitute the model combination commonly done by averaging the contributions of each PRLM subsystem by a model combination based on SVM. Second, he proposed to substitute LMs after each phone recognizer by SVMs too. Also, he proposed to merge sounds from the phone recognizers that do not match in the target languages to obtain less patterns or tokens to be recognized but with higher frequency of apparition. This allowed creating more robust models. Finally, he also proposed to create cross-language models or cross-stream dependencies, similarly to Kirchhoff and Parandekar [2001].

In 2006, Noor and Aronowitz presented a system based on anchor models, where a GMM is trained for a set of speakers called anchors [Noor and Aronowitz, 2006]. A technique named test utterance parametrization (TUP) was used, where a GMM is trained for the test speaker and another GMM is trained for the target speaker. The Gaussians were compared to determine if they belonged to the same speaker or not by using multiple discriminant analysis (MDA) and SVM. They trained the models for each language with the scores of all anchor models.

Brümmer and van Leeuwen in 2006 precisely defined the concept of calibration of LID scores [Brümmer and van Leeuwen, 2006]. This work had important consequences on the LID community, since many people became aware of the importance of calibration. He introduced the metric C_{llr}, an empirical measure of information expressed in terms of bits of Shannon entropy. He developed the software FoCal Multi-class Toolkit [Brümmer, 2007], very useful to fuse and calibrate LID scores, and exploited by a large number of researchers of the community.

3.4 Period 2007-2010: the Factor Analysis Era

The main contribution to LID in this period was the application of JFA to compensate for channel mismatches. This technique was introduced by Kenny for speaker recognition in 2004 [Kenny and Dumouchel, 2004; Kenny, 2006; Kenny et al., 2007]. The main idea is that the means of the GMM can be factorized, and each factor models a different aspect of speech. Usually in LID, one factor models the language, and one factor models the channel. Also, other approaches to reduce noise mismatches in the feature domain, like nuisance attribute projection (NAP), were investigated. SVMs were used in the acoustic systems mainly with GMM means as input. In the phonotactic approach, the use of the n-gram counts as feature vector was the main used technique. In this case, SVM was the preferred technique for classification.

For prosody, new features to model rhythm, stress, and intonation were investigated, and techniques used in the acoustic and phonotactic systems were also tried, like n-gram counts with SVM.

NIST continued with LREs in 2007 [National Institute of Standards and Technology (NIST), 2007] and 2009 [National Institute of Standards and Technology (NIST), 2009], and very competitive systems were presented. LID technology was very robust. In Spain in 2010, the Red Temática en Tecnologías del Habla (RTTH) organized the Albayzin LRE to evaluate LID systems where the target languages were the languages spoken in the Iberian peninsula [Red Temática en Tecnologías del Habla, 2010].

In 2007, Li [Li et al., 2007] expanded his work about bag-of-sounds of 2005 [Li and Ma, 2005]. He proposed to use segments universal to all languages in two ways: either with several phoneme recognizers to build a single vector of n-grams by concatenating their individual vectors, in a similar approach to PPR, or with universal phones

extracted from a single universal phone recognizer that build a single n-gram vector. Specifically, he used trigrams. This vector was the document to be classified, and the classifier could be a traditional LM or a classifier in the vector space model (VSM) like an SVM or a neural network. In his experiments, he used an SVM. The phonemes in the universal multilingual phoneme recognizer were obtained by an automatic segment model. He used a coverage of 258 phonemes of 6 languages, and they were modeled with HMMs with GMM output distributions of 32 components.

The work of Castaldo in 2007 can be considered to be the first in using the FA techniques [Castaldo et al., 2007c]. Actually, he proposed two new techniques for an LID system. The FA one, with high impact on posterior researches, consisted in a MAP channel adaptation in the form of a subspace of a GMM, also known as eigenchannels. This compensated interspeaker and intersession variability that could be found within a language. The other, time-frequency or discrete cosine transform (DCT) features, which were an alternative to SDC, more flexible, and also allow controlling the time interval spanned by the features. They showed similar performance to SDC and the combination of both was complementary. In a second work during this year, Castaldo investigated the performance of a 512 component GMM LID system with channel compensation in the feature space [Castaldo et al., 2007b]. He proposed to subtract the contribution of the intersession compensation values obtained with the first 5 top-scoring Gaussians to the features. The intersession compensation values were obtained with the intersession factor loading matrix. He performed a similar idea to compensate for speaker variability within each language. He also used a GMM-SVM system, a technique borrowed from speaker recognition that creates supervectors for training and testing by adapting the means of the UBM to the corresponding train or test utterances by MAP. The supervectors were classified with an SVM. He remarked that the factor loading matrix trained for channel compensation used the same data used for training the channel compensation matrix in a speaker recognition task, and then, the matrix was task-independent and reusable, and that the compensated features could be used for another classifiers like SVM, and not only to the GMM framework. In a third work in 2007, he proposed to train discriminative GMMs from the separating hyperplanes of a GMM-SVM system [Castaldo et al., 2007a]. When moving the Gaussian components of the GMM in this way, the total log-likelihood of the training data decreased, because the parameters were optimized for maximum log-likelihood. However, the equal

error rate (EER) also decreased, because the models of different languages were more separated. When he compared the GMM approach with the GMM-SVM, the former worked worse for long test utterances and better for shorter utterances.

In 2007, Cordoba proposed to substitute the classical LM by an n-gram frequency ranking in P-PRLM, where the scores were calculated as the distance between the ranking of the n-grams sorted according to the number of counts obtained from the training database, and the ranking of the n-grams sorted according to the number of counts obtained from the test utterance [Cordoba et al., 2007]. Better performance was obtained using only the most discriminative n-grams.

Ma in 2007 [Ma et al., 2007] built an LID system based on phoneme recognizers followed by a series of 1-vs-all SVMs [Ma et al., 2007]. The output of the SVMs were 0 or 1 depending on if the target language was present in the test utterance or not. Combining this binary output of all the recognizers he reduced the dimension to the number of target languages. This binary vector was used for classification. He relaxed the binary output to obtain continuous values. The classifier for each language was a GMM.

Rouas in 2007 worked on prosodic LID [Rouas, 2007]. He distinguished between sub-phonemic segments and pseudo-syllables to segment speech. First, he modeled F0 by linking the minima of the values, and calculating the linear regression curve of those points. Features were discrete, and they indicated if the curves were ascending, descending, or unvoiced. Another alternative was to subtract the regression curve to the raw values and performing the same discrete analysis into ascending, descending, or silence classes. The result of the difference was called residue. The energy was modeled in the same way, but only with ascending and descending classes. Duration was coded as short or long, if the segment was above or below the average segment duration. He built two systems. One based on short-term modeling, including the sub-phonemic segmentation, and using the residue, energy and duration labels. Other based on long-term modeling used pseudo-syllables, and F0 and energy labels. The labels were modeled by LMs with trigrams.

Yin proposed in 2007 an innovative hierarchical LID system [Yin et al., 2007]. First, he defined a distance metric between clusters to measure similarity. Then, he used a hierarchical agglomerative approach, in which he started with all the clusters independently and continued joining them into groups according to the similarity obtained

at each step. The distance was based on the accuracy obtained by the classifier. In this way he built a tree with the clusters. During the evaluation, he followed the tree to reach a leaf and decide a language. He investigated different types of features to perform the clustering, like MFCC, pitch and intensity to capture prosody, and MFCC with the prosodic features together. He used a GMM with 256 components for classification. In 2008, he improved his system by adding different distance metrics, namely a PRLM system, and a fusion at each of the hierarchy step instead of a single LID system [Yin et al., 2008a]. The same year, Yin presented frequency modulation (FM) features for LID [Yin et al., 2008b]. Traditionally, MFCC-like features only carry magnitude information, while phase information is discarded. FM features try to capture phase information. The features were also modeled with a GMM.

Richardson and Campbell in 2008 suggested a method to select discriminative n-grams in a phonotactic SVM LID system [Richardson and Campbell, 2008]. The idea was to start with n-grams, add another phone to have n+1-grams, and remove the least discriminative ones from the resulting list.

Mary and Yegnanarayana in 2008 presented an LID system based on prosody [Mary and Yegnanarayana, 2008]. First, they extracted syllable-like units without any ASR by automatically searching for the vowel-onset point (VOP). Then, they extracted parameters to represent intonation, rhythm, and stress from three consecutive syllables, and they concatenated the parameters to feed a multilayer feedforward neural network. The parameters to represent intonation were the change in F0, the distance from F0 peak to VOP, the amplitude tilt, and the duration tilt. Syllable duration and duration of the voiced region within a syllable were used to represent rhythm. They used the change in energy within a voiced region as stress feature.

The work of Sim and Li in 2008 proposed the use of acoustic diversification in addition to the traditional phonetic diversification found in P-PRLM systems [Sim and Li, 2008]. The standard approach in P-PRLM was to use several phoneme recognizers trained on different languages. The idea of acoustic diversification was to use a single phoneme recognizer trained with phonemes of just one language, but trained several times, each time with a different techniques. The techniques that the author proposed were ML and MMI with diagonal covariance, semi-tied covariances, and subspace for precision and mean.

In 2008, Hubeika proposed channel compensation in a GMM system using eigenchannels in feature and model domain [Hubeika et al., 2008]. The approach was similar to the works of Castaldo in 2007 Castaldo et al. [2007b,c]. She also combined it with MMI using 2048 Gaussian components. The improvement of channel compensation techniques were very impressive. Channel compensation in model domain performed better than in feature domain. Additionally, she showed the complementarity with a phonotactic system. Moreover, the system was successfully used for theNIST LRE 2007 Matejka et al. [2008].

Campbell in 2008 compared two subspace methods for channel compensation in the feature space for LID [Campbell et al., 2008]. These were feature domain latent factor analysis (fLFA) and his proposal, feature domain nuisance attribute projection (fNAP). For classification, he used SVM where the inputs were a kernel with GMM means and covariances of a UBM adapted to each utterance.

Glembek presented some advances to phonotactic systems in 2008 [Glembek et al., 2008]. He proposed the adaptation of binary trees and LMs from an UBM, a novel smoothing technique applied to LMs, FA applied to the phonotactic approach, and multi-models, a different model for each language depending on the accent or on the dialect, with supervised labeling. He showed that binary trees and LMs performed similar with these improvements.

Montavon in 2009 proposed a deep convolutional neural network (CNN) implemented as a time-delay neural network (TDNN) [Montavon, 2009]. This was the first work on deep architectures, although the big effort on these techniques for LID would arrive some years later. He wanted to better model short utterances that other approaches did not model properly. The input to his network were matrices of 39x600 with 39 MFCCs between 0 and 5 kHz, and a temporal context of 600 frames representing 8.33 ms. The deep architecture contained 5 layers and a total of $2.8 \cdot 10^7$ neuron connections, and it was trained with stochastic gradient descent (SGD).

Ng in 2009 made an experiment to select the best combination of prosodic features [Ng et al., 2009]. The classifier was a bag-of-sounds with trigrams and SVM. The features were divided in F0, energy, and duration groups, and there were four types: frame-based, syllable-based, regression, and residual. The selection was made with a mutual information criterion.

In 2009, Tong presented some proposals to improve a phonotactic PPR LID system [Tong et al., 2009a]. The system was configured as the phone recognition followed by concatenation of n-grams classified with an SVM. The first proposal was to restrict the number of phones of each PPR branch to the most discriminative ones for the target language corresponding to that branch. This allowed training models with higher n-grams. The criteria to select the most discriminative phones were the separation margin of the SVM given by the phone and the mutual information between the phone and the language. The second proposal was a tokenizer selection. That means, using only those phoneme recognizers that obtain better results. The phoneme recognizers selected were called target-oriented phone tokenizers (TOPTs). Finally they proposed to use universal phone tokenizers, with phones from as many languages as possible. In another work of 2009, the authors presented target aware language model (TALM) for P-PRLM [Tong et al., 2009b]. In this case, several LMs were built from the same phone recognizer, each for a target language, only with the most discriminative phones for that target language.

Castaldo in 2009 proposed an approach for LID similar to the eigenvoice approach for speaker recognition [Castaldo et al., 2009]. He proposed to create language factors by performing PCA on the differences between the set of supervectors of a language and the average supervector of every other language. The language factors had low dimensionality and could be the input to an SVM classifier. He also applied nuisance compensation. He used language factors to adapt GMM means and then the resulting model was evaluated, and also to directly classify them with an SVM. Since SVMs were computed from the means of a GMM, the support vectors could be used to build a model for positive samples, associated to positive Lagrange multipliers, and an antimodel with the support vectors associated to negative Lagrange multipliers. The positive and negative SVM boundaries could be weighted and combined to create a GMM, and then obtaining log-likelihood ratios from them. This model could also be trained discriminatively.

Yin in 2009, on the premise that phone durations are different in different languages, proposed to build an LID classifier based on the duration patterns of the sounds [Yin et al., 2009]. He took as unit the pattern formed by an unvoiced-voiced group, what created 2-dimension features. He also experimented with individual unvoiced and voiced units. He normalized and quantized the durations to obtain discrete values that could

be modeled with an LM. Thus, he modeled the temporal sequence with n-grams, bigrams and trigrams. He used 20 discrete values.

The FA approach applied to LID continued in the work of Brümmer in 2009 [Brümmer et al., 2009]. He presented a 2048 component GMM system for LID with channel compensated statistics, using the ideas of JFA. First, the language models were adapted from a UBM via MAP. The channel subspace was modeled with FA for each of the languages, but with a shared subspace. The channel loading matrix was obtained with the EM algorithm, and channel factors were obtained as a MAP point-estimate of its posterior distribution given the sufficient statistics. The score of the baseline system was computed by dot scoring of the language locations and the sufficient statistics. The scores with channel compensated statistics were obtained in a similar way, but the language locations were also obtained with channel compensated statistics. The language locations could be also obtained discriminatively with MLR, either in a pair-statistic or per-segment fashion.

In 2009, Verdet applied a similar idea to Brummer's work, with FA over a GMM [Verdet et al., 2009]. He proposed to first adapt each language by MAP, and then, to obtain the channel subspace over the remaining model, with a single channel matrix for all languages. The channel compensation was added to the model of each language at test, and the unnormalized feature was scored over it. He stated that the channel factors could be computed individually for each language, or a single one for all languages from the UBM. He followed the latter strategy for the experiments. For evaluation, he stated that one could first compensate the features and apply the LM, or one could use the non compensated statistics over the compensated model.

Stolcke in 2010 investigated the performance of multilingual recognizers on the 30 s task of 2005 NIST LRE [Stolcke et al., 2010]. He used PRLM and P-PRLM systems. First, he showed that a multilingual phoneme recognizer with phones of individual languages mapped to a multilingual phone set performed better than language-dependent phoneme recognizers. The system was fed with PLP features. By adding MLP features results were improved. Also, he experimented with maximum likelihood linear regression (MLLR) models for speaker adaptation, and NAP for noise compensation. He obtained the best results by fusing MLLR, cepstral GMM, PRLM, and phonotactic SVM.

In 2010, Mikolov presented a phonotactic P-PRLM system with important computational reductions [Mikolov et al., 2010]. First, he proposed to obtain the square roots of the expected n-gram counts of lattices. Then, he selected the most frequent n-grams in the training dataset. Next, he proposed to compute a PCA transformation to reduce the high-dimension vectors to a dimension more tractable by an SVM. Thus, for example, by using trigrams of the BUT Hungarian phoneme recognizer with a reduced phoneme repertoire of 33 phonemes, the initial vector was of dimension 35937, and he reduced to 4000 and 100 dimensions with PCA while keeping the classification accuracy.

The importance of data processing in 2009 NIST LRE was reflected in the work of Jancik in 2010 [Jancik et al., 2010]. The authors presented the system submitted by BUT and Agnitio to the 2009 NIST LRE. Their JFA model adapted to LID was built by adapting a GMM via MAP to each target language, and the channel compensation subspace was modeled with ML. They used 2048 components, and SDC features in 7-1-3-7 configuration. This system was one of the best performing ones in the 2009 NIST LRE. They also presented region dependent linear transform (RDLT) features for LID, but these did not perform very well.

In 2010, Ng made an experiment to select prosodic features for LID [Ng et al., 2010]. He discretized F0, intensity, and duration, and he used 20 and 67 different attributes formed with this information. For classification, he built bigrams and he used an SVM.

Sangwan in 2010 proposed a production model based on phonological features, specifically hybrid features including height-of-tongue, frontness-of-tongue, rounding, nasality, voicing, place-of-articulation, and manner-of-articulation [Sangwan et al., 2010]. Combinations of these features were grouped according to if they belonged to consonant, vowel, or consonant-vowel clusters. Also, pitch contours were identified and approximated by cubic polynomials, whose coefficients were used to cluster the contours with K-means. n-gram combinations were built with these groups and a maximum entropy model was trained for classification.

3.5 Period 2011-2014: The *i-Vector* Era

JFA was used as a front-end to extract *i-Vectors*. *i-Vector* became the state-of-the-art technique for acoustic, phonotactic, and prosodic systems. Most of the works focused

on the study of backends to classify *i-Vectors*, and different acoustic, phonotactic, and prosodic features to be modeled with the *i-Vector* approach.

In 2011, NIST conducted the last LRE to date [National Institute of Standards and Technology (NIST), 2011]. The goal was to discriminate between pairs of languages, instead of identifying or detecting one among a set of L. The technology was mature, and data management was one of the keys to obtain good results. In Spain in 2012, RTTH organized a second Albayzin LRE [Red Temática en Tecnologías del Habla, 2012].

i-Vectors were used for the first time for LID in 2011. Our work presented in Martínez et al. [2011b], and the work of Dehak presented in Dehak et al. [2011a] applied this technique with great success in the acoustic space. In our work, we compared generative and discriminative classifiers, and the generative ones outperformed the discriminative ones. Specifically, the best results were obtained with 600 dimension *i-Vectors* classified with a Gaussian model with shared covariance among languages. In addition, the fusion with a JFA system helped. In the work of Dehak, *i-Vectors* were used with SVMs and different dimensionality reduction techniques like linear discriminant analysis (LDA) and neighborhood component analysis (NCA). They found 400 to be the optimal *i-Vector* dimension. However, he obtained similar or better results with MMI. Fusing *i-Vectors* and MMI only helped in the 3 s task case.

Soufifar in 2011 presented a novel method to represent n-gram counts in phonotactic LID using *i-Vectors* [Soufifar et al., 2011]. He modeled the counts with a multinomial model, and created a subspace spanning the variability in the counts, where *i-Vectors* lied. The classification was made with SVM and logistic regression (LR).

In 2011, we presented the results we obtained in the 2010 Albayzin LRE [Martínez et al., 2011a]. In our phonotactic systems, N-best hypothesis were taken instead of lattices to reduce complexity.

In 2011, other approach to score *i-Vectors* was proposed by Lopez-Moreno [Lopez-Moreno et al., 2011]. He proposed the von Misses-Fisher distribution, which models directional data.

Plchot in 2012 proposed to use features derived from a subspace Gaussian mixture model (SGMM) for LID [Plchot et al., 2012]. The idea of SGMM is that the GMM mean supervectors modeling output distributions in HMMs of an LVCSR system can be represented in a low-dimension subspace, which is common to all the states. He

added a term to adapt also the speaker, and the speaker-specific vectors were used for classification. He used a phoneme recognizer instead of LVCSR. For classification he used MLR. Note that this SGMM is a specific technique, while we have used the name subspace Gaussian mixture models in the title of the Thesis to refer to all methods belonging to the family of linear Gaussian models described along the work.

In 2012, Singer presented the MITLL system submitted to the 2011 NIST LRE [Singer et al., 2012]. One of them proposed to score *i-Vectors* with dot-scoring for LID.

Matejka in 2012 presented some state of the art systems on the robust automatic transcription of speech (RATS) database [Matejka et al., 2012]. RATS contains recordings from existing databases and new collections of data retransmitted by 8 degraded channels, and it presents very challenging scenarios. They used 600 dimension *i-Vectors* with a neural network classifier with 3 hidden layers.

Penagarikano in 2012 analyzed the performance of different backends in an LID system [Penagarikano et al., 2012]. He explored: a generative Gaussian backend with a Gaussian trained for each target language; a discriminative Gaussian backend, where the Gaussian was refined with the MMI criterion; a fully-Bayesian Gaussian backend, where priors were set to the model parameters and integrated to evaluate the likelihood; the generative Gaussian mixture backend, where more than 1 component was added to the Gaussian model; and an LR backend, where the scores were transformed according to MLR. Also, the influence of Z-norm and ZT-norm were studied.

D'Haro in 2012 introduced a new approach to cope with the outputs of a phoneme recognizer in a phonotactic LID system [D'Haro et al., 2012]. They proposed to work with the posterior probabilities of each of the phonemes. First, they segmented the speech into phonemes, they took all the frames belonging to each phoneme, and they averaged the posterior probabilities within each segment, to obtain a single vector of posterior probabilities per phoneme. They built a joint posteriogram, i.e., a matrix built by doing the outer product of the $n-1$ previous vectors with the current one. In this way, a sequence of nxn matrices was obtained. Next, they summed all the joint posteriogram matrices to obtain a single matrix per utterance. Then, they concatenated all the columns to build a row vector with all possible n-grams. This vector was the input to a subspace multinomial model, from which *i-Vectors* were extracted in a similar way to Soufifar et al. [2011]. In this work, the authors used trigrams. For *i-Vector* classification, MLR was used.

Diez in 2012 [Diez et al., 2012], proposed to use phone log-likelihood ratios (PLLRs) instead of the posterior probabilities of the phonemes. The likelihood of a phone was the sum of the likelihoods of each state within an HMM modeling that phone. They modeled these features with the *i-Vector* framework, with a 1024-component GMM. Only the first PLLR derivative helped. In 2013, they introduced dimensionality reduction techniques like PCA to boost performance of PLLR features [Diez et al., 2013].

In 2012, we presented an *i-Vector*-based system with prosodic features [Martínez et al., 2012a]. The features were pitch, energy, and duration of the segments where these features were computed. Different segments were computed, like the ones defined by energy valleys, or fixed segments of 200 ms shifted every 50 ms. The fixed segments solution was the one that performed the best. In the segments, the contour of the pitch and energy was calculated by means of the Legendre polynomials. The polynomial coefficients and the duration were the input to the *i-Vector* system. The UBM had 2048 components and *i-Vectors* had 400 dimensions. *i-Vectors* were classified with a Gaussian classifier. In 2013, we enriched the prosodic system presented previously with formant information [Martínez et al., 2013b]. Formants were also modeled with Legendre polynomials. It was checked an increase in performance by using regions shifted 10 ms instead of 50 ms. In this experiment, the UBM had 2048 components, and the *i-Vectors* had 600 dimensions. The best performance was obtained by adding F1 and F2 frequencies to the prosodic system.

Ganapathy in 2012 presented a system to adapt models in a noisy task, in a supervised and in an unsupervised manner [Ganapathy et al., 2012]. The baseline system classified GMM means with an SVM. The unsupervised adaptation was made with an MLP classifier in a cotraining framework.

McLaren in 2013 proposed a backend classifier trained only on the most similar data to the test utterance [McLaren et al., 2013]. He called it adaptive Gaussian backend (AGB). He compared a situation with a Gaussian trained only with the closest *i-Vectors* in terms of Euclidean distance, and the situation where the support vectors of a previously trained SVM were used to build a Gaussian for each target language. In this case, the impostor class only contained the test utterance, even if this was of the target language. The support vectors were weighted by their Lagrange multipliers.

In 2013, Lawson studied several features with different *i-Vector* classifiers [Lawson et al., 2013]. The features were combined with different classifiers, namely a Gaussian

backend, an AGB, and neural networks. Then, the following combinations of features and classifiers were explored: subband autocorrelation classification with Gaussian backend, mean Hilbert envelope coefficient with AGB, medium duration modulation cepstral with neural networks, power normalized cepstral coefficient (PNCC) with AGB, MFCC with neural networks, and PLP with a Gaussian model. Finally, some system fusions were investigated with SVM.

Song in 2013 presented bottleneck features calculated from a deep neural network (DNN) of 5 layers for LID [Song et al., 2013]. In the input, he used 39 MFCCs and 4 pitch parameters. He used 10 frames, so in total, he used 430 visible units. He extracted 43 bottleneck features. Then, he built an *i-Vector* system of 600 dimensions on top of the bottleneck features. He applied LDA and within class covariance normalization (WCCN).

Nandi in 2013 made an experiment for LID based on the Hilbert envelope and phase information of the LPC residual [Nandi et al., 2013]. Phase information obtained better results than Hilbert envelope joining subsegmental, suprasegmental, and segmental levels of information.

3.6 Period 2014- : The Deep Era

In this period, very good results were presented with DNNs, specially for shorter durations, where the *i-Vector* modeling capabilities were less accurate. The motivation of using DNNs came from the speech recognition field, where they had been shown to be very powerful for acoustic modeling. In the previous works Montavon [2009] and Song et al. [2013], the authors experimented with deep CNNs and bottleneck features for LID, respectively, with good results. The first attempt of directly applying DNNs for LID was made at the 2013 Johns Hopkins University (JHU) Summer Workshop [The Center for Speech and Language Processing at Johns Hopkins University, 2013], where we also contributed [Lopez-Moreno et al., 2014]. In this experiment, we had a very large database of short segments available provided by Google. The DNN had 8 layers with 21 input frames (10 past, current, and 10 future frames), and 2560 hidden nodes. The input features were 39 PLPs. The training was made with asynchronous stochastic gradient descent (ASGD). Soon later, other works using bottleneck features and other DNN configurations were presented. We think that this technology can still

improve state of the art performance, specially for short segments. Additionally, DNNs are being investigated for other purposes, like noise compensation. Nevertheless, the *i-Vector* approach, either with SDC, or with bottleneck features, was still offering better classification rates for segments longer than 10 s than DNNs alone..

In 2014, NIST announced that there will be another NIST LRE at the beginning of 2015 [Martin et al., 2014], that will be focused on language pairs discrimination.

In 2014, D'Haro presented some modifications to PLLR features [D'Haro et al., 2014]. The first one was to use the log likelihood ratio of each individual state instead of summing up the posteriors corresponding to phone states in an HMM. The motivation was to take advantage of the transitions between phones as well as between states. The log likelihoods of each state were concatenated, resulting in higher dimensional features. This was compensated with a PCA dimensionality reduction. The posterior modeling was done with *i-Vectors*.

The work of Lei in 2014 presented two different approaches of CNNs adapted to LID [Lei et al., 2014]. First of all, a CNN was trained for an ASR system. The first of the two approaches proposed to use this CNN to obtain the zeroth order statistic of the senones (tied states of phones) of the ASR. That is, it was used to align the frames instead of the traditional GMM-UBM. Once they were aligned, the normal procedure with GMM-UBM was followed, and single Gaussians were computed for each senone and first order statistics could be obtained from them, even with different features. The second approach was to use the posterior of the senones obtained from the CNN as features. The vectors of posterior probabilities were reduced in dimension with PPCA, and the resulting vectors were modeled by the usual *i-Vector* system. In this case, note that only zeroth order statistics of the initial senones were used.

In 2014, Boonsuk presented an LID system with an latent-dynamic conditional random field (LDCRF) fed with phonological features based on the sound pattern of English, where each phone can be broken in 14 phonological feature attributes [Boonsuk et al., 2014].

Zhang in 2014 made an experiment to study the behavior of different features and fusions in an LID system [Zhang et al., 2014]. He studied MFCC, PLP, linear frequency cepstral coefficient (LFCC), gammatone frequency cepstral coefficient (GFCC), PNCC, perceptual minimum variance distortionless response (PMVDR), RASTA-PLP,

RASTA-LFCC, multi-peak MFCC, Thomson MFCC, and sine-weighted cepstrum estimator (SWCE) MFCC. First, he obtained 400 dimension length-normalized *i-Vectors*, and for classification, he proposed a Gaussian backend and a Gaussianized cosine distance scoring. For fusion, he used two complementary alternatives: the front-end fusion, where he concatenated the *i-Vectors* coming from different features, and the back-end fusion, where he used a discriminative LR to combine the outputs of the two previous classifiers.

McCree in 2014 presented an *i-Vector*-based system with a Gaussian classifier retrained discriminatively with MMI [McCree, 2014].

Matejka in 2014 presented another approach of bottleneck features for LID [Matejka et al., 2014]. First, the signal was filtered in several frequency bands applying Bark critical band windowing. Frequency domain linear prediction (FDLP) was applied to the filters to obtain a smooth parametric model of temporal envelope. He used 476 modulation coefficients and 11 pitch coefficients, totaling 487 coefficients for the input to a first neural network that obtained a first output of 80 bottleneck features. 5 consecutive frames of these bottleneck features were stacked to obtain a context of $\pm$ 10 frames and they were used as input to a second neural network, that obtained a second output of 80 bottleneck features. Bottleneck features had linear activation functions. The second output were the features to train a standard *i-Vector* system. The *i-Vector* extractor was based on a 1024 GMM and they had 400 dimensions. *i-Vectors* were classified with another neural network, and LR was used to calibrate the results.

In 2014, we introduced an unsupervised accent modeling in the *i-Vector* space [Martínez et al., 2014b]. First, we performed an agglomerative hierarchical clustering (AHC) step to find accents within each language. Then we classified with probabilistic linear discriminant analysis (PLDA), in a similar way to a speaker recognition system. Actually, the grouping did not have necessarily to correspond to actual accents, but just to similar acoustic characteristics.

4

Experimental Setup

Contents

In this chapter, we present the basic architecture of our LID systems. This will be conveniently adapted for the acoustic and prosodic approaches. Also, we describe the experimental setup of our work, including database, and evaluation metrics.

4.1 General Architecture of our LID system

In Figure 4.1, we can see the general architecture of our LID systems. Before we decide the language spoken in test utterance, the main blocks that it passes through are the feature extractor, which parametrizes the signal; the language models, which calculate a score for each target language from the features; the calibrator, which transforms the scores so we can use the *Bayes decision threshold*; and the decisor, which makes decisions, typically comparing the calibrated score to the *Bayes decision threshold*. Preferably, we use three different datasets to conduct experiments: the *Train* dataset, to train model parameters; the *Dev* (of development) dataset, to train the calibration; and the *Test* dataset, which is used to evaluate our system. The *Dev* data should match as much as possible the *Test* data, in order to have a system adapted to the type of data under evaluation.

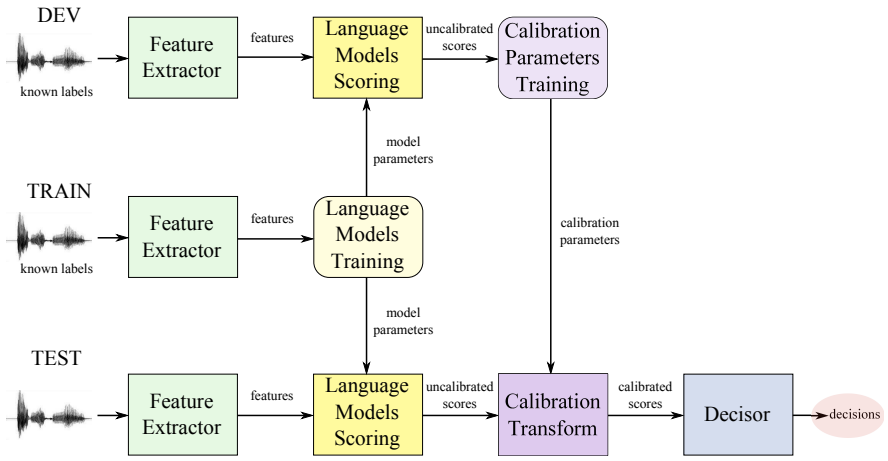


Figure 4.1: General architecture of our LID system - First, the audio signal is parametrized. Language models use features to compute scores for each language. Scores are calibrated in the calibrator. Finally, decisions are made in the decisor block. *Train* dataset is used to train the language models, *Dev* is used to train the calibration block, and *Test* is used to evaluate the system.

In this Thesis, the feature extractor converts the audio signal into acoustic or prosodic features. The feature extraction process for acoustic features will be detailed in Section 6.1, and in Section 7.2, for prosodic features. In addition, *i-Vectors* will be computed in this block where appropriate.

The language model block is the core of the classifier because it will generate scores for each language. In our experiments, we will use GMM, JFA or *i-Vector* to model languages. GMM and JFA are generative models. In the *i-Vector* approach, we will compare generative and discriminative classifiers. In fact, any classifier able to generate scores could be here. The scores coming out of this block are not necessarily calibrated, so a post-processing step is required to be able to apply the *Bayes decision threshold*.

The calibration is a final step in the system that allows to transform the scores in such a way that: *i)* scores are well-calibrated so cost effective Bayes decisions can be made, by setting the threshold to the *Bayes decision threshold*, η,

$$\eta = \log \frac{C_{fa}}{C_{miss}} - logit(\pi), \tag{4.1}$$

with π being the prior probability of the target language, and C_{fa} and C_{miss} the costs of false alarms and miss, respectively; *ii)* scores coming from different recognizers are fused to obtain a better recognizer. In our experiments, the calibration block has two steps, a generative and a discriminative calibration. They were trained with FoCal Multi-class Toolkit [Brümmer, 2007]. We will explain this block in detail in Section 4.2.

Finally, decisions using the *Bayes decision threshold* can be made with the calibrated scores. Remember that our task is the detection of languages, and this means that the same utterance can be said to be in zero, one, or more than one language.

4.2 Calibration and Fusion

The calibration block in Figure 4.1 is composed of two blocks in cascade. The first is a generative calibration called Gaussian backend (GB), and the second is a discriminative calibration performed with MLR. This decomposition is shown in Figure 4.2. The training of these two blocks is preferably done with a *Dev* dataset not seen during training, that matches as much as possible the *Test* dataset. The reason is that this block must serve as a bridge between our system and the outer world, and it must allow

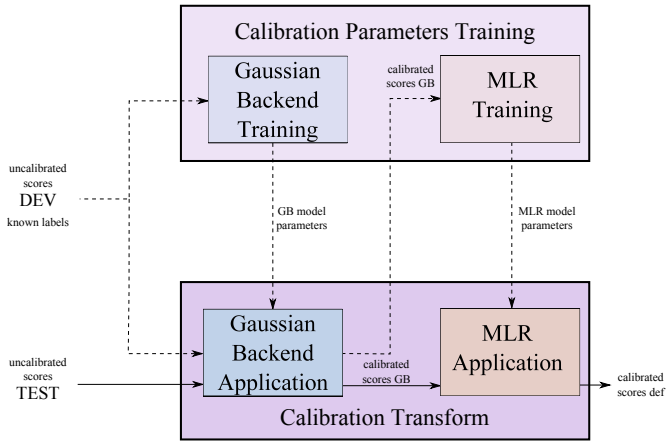


Figure 4.2: Calibration Blocks - The calibration is composed of two blocks in cascade: a GB, and an MLR. Both are trained on a *Dev* dataset. The output of the GB is used to train the MLR block.

us to adapt our system to the type of data seen in real applications. This configuration with two blocks in cascade is not crucial to obtain good results. In general, good calibrations are achieved only with the discriminative block. Actually, we have observed that adding the generative calibration only helps in some situations, and it eventually depends on the database, but there is no theoretical reason to choose one configuration or another. The criterion for the configuration selection is merely empirical [Brümmer, 2007].

4.2.1 Generative Calibration

The GB is essentially an ML Gaussian classifier trained on the scores coming from the models of each target language. Thus, a Gaussian model is trained for each language l, and consequently, we have to calculate the mean, μ_l, and the covariance, $\mathbf{\Sigma}_l$, of each Gaussian. The covariance is shared among languages, and is equal to the within-class (WC) covariance of the data. So, $\mathbf{\Sigma}_l = \mathbf{\Sigma}_{WC}$. The ML training process can be summarized as follows:

- Take all *Dev* utterances belonging to each target language. Suppose we have N_l utterances for language l.
- For each utterance, score the models of the L target languages. An L-dimension vector of uncalibrated scores, $\mathbf{s}_l(\mathbf{o}_n)$, is obtained for utterance $\mathbf{o}_n$ of language l.

- For each language, obtain the mean of the distribution,

$$\mu_l = \frac{\sum_{n=1}^{N_l} \mathbf{s}_l(\mathbf{o}_n)}{N_l}. \tag{4.2}$$

- With $\bar{\mathbf{S}}_l$ being a matrix whose columns are the score vectors of language l centered around their mean, $\bar{\mathbf{S}}_l = [\mathbf{s}_l(\mathbf{o}_1) - \mu_l \ ... \ \mathbf{s}_l(\mathbf{o}_{N_l}) - \mu_l]$, the WC covariance matrix is obtained,

$$\mathbf{\Sigma}_{WC} = \frac{\sum_{l=1}^{L} \bar{\mathbf{S}}_l \bar{\mathbf{S}}_l^\intercal}{\sum_{l=1}^{L} N_l}. \tag{4.3}$$

During the evaluation, the log-likelihood for each target language, l_l^{gen}, is just the log Gaussian probability density of the score-vector, given the language. The formula to compute this log-likelihood is the log of eq. (2.1), which results in

$$\begin{aligned} l_l^{gen}(\mathbf{o}_n) = \ln p(\mathbf{s}_l(\mathbf{o}_n)|l) = \\ -\frac{L}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{\Sigma}_{WC}| - \frac{1}{2}\mathbf{s}_l(\mathbf{o}_n)^\intercal \mathbf{\Sigma}_{WC}^{-1}\mathbf{s}_l(\mathbf{o}_n) + \mathbf{s}_l(\mathbf{o}_n)^\intercal \mathbf{\Sigma}_{WC}^{-1}\mu_l - \frac{1}{2}\mu_l^\intercal \mathbf{\Sigma}_{WC}^{-1}\mu_l \end{aligned} \tag{4.4}$$

However, given that the covariance is shared among classes, we drop the constant terms (first, second and third terms on the right-hand side [RHS] of eq. 4.4), which would only shift the same amount the L log-likelihoods of each utterance. Then, GB becomes a linear transform, where the scores are scaled by vector $\mathbf{a}_l$ and shifted by the scalar b_l,

$$l_l^{gen}(\mathbf{o}_n) = \ln p(\mathbf{s}_l(\mathbf{o}_n)|l) = \mathbf{s}_l(\mathbf{o}_n)^\intercal \mathbf{\Sigma}_{WC}^{-1}\mu_l - \frac{1}{2}\mu_l^\intercal \mathbf{\Sigma}_{WC}^{-1}\mu_l = \mathbf{a}_l \mathbf{s}_l(\mathbf{o}_n) + b_l, \tag{4.5}$$

with

$$\mathbf{a}_l = \mu_l^\intercal \mathbf{\Sigma}_{WC}^{-1}, \tag{4.6}$$

$$b_l = -\frac{1}{2}\mu_l^\intercal \mathbf{\Sigma}_{WC}^{-1}\mu_l \tag{4.7}$$

Since this is not the last block in our system, dropping the quadratic term can bring differences in the results. Nevertheless, we prefer to keep this calibration step linear for simplicity. A deeper analysis of this issue is given in Section 5.4.1.1.

4.2.2 Discriminative Calibration

In the discriminative calibration step, a calibration-sensitive metric, C_{llr} [Brümmer, 2007] is optimized. The objective is to better calibrate the previously calibrated log-likelihood vectors in such a way that this error metric is minimized. C_{llr} is defined as

$$C_{llr} = -\frac{1}{T}\sum_{t=1}^{T}\omega_t \log_2 P_t, \tag{4.8}$$

where, if we define a trial as the evaluation of an audio recording against a target language in a detection task, T is the number of trials (equal to the number of *Test* files times the number of target languages), and ω_t is a weight to normalize the class proportions in the evaluation trials,

$$\omega_t = \frac{\pi_{c(t)}}{Q_{c(t)}} \tag{4.9}$$

,

$$Q_i = \frac{nr.\,of\,trials\,of\,class\,H_i}{T}, \tag{4.10}$$

$c(t)$ is the true language of trial t with $c(t) \in \{1, 2, ..., L\}$, $\pi_{c(t)}$ is its prior, and P_t is the posterior probability of hypothesis $H_{c(t)}$ of true language given the L-dimension vector of discriminatively calibrated log-likelihoods, $\mathbf{l}^{dis}(\mathbf{o}_t)$, which is computed as follows,

$$P_t = P(H_{c(t)}|\mathbf{l}^{dis}(\mathbf{o}_t)) = \frac{\pi_{c(t)}e^{\mathbf{l}^{dis}_{c(t)}(\mathbf{o}_t)}}{\sum_{l=1}^{L}\pi_l e^{\mathbf{l}^{dis}_{l}(\mathbf{o}_t)}}, \tag{4.11}$$

with $\mathbf{o}_t$ being the observation of trial t. See how C_{llr} evaluates an N-class problem built from several binary detection trials.

C_{llr} has the sense of a cost and it is measured in terms of bits of information. $0 \leq C_{llr} \leq \infty$, where 0 means perfect recognition.

Well-calibrated discriminatively log-likelihoods, $\mathbf{l}^{dis}(\mathbf{o}_t)$, are the final output of our calibration procedure. They are obtained as,

$$\mathbf{l}^{dis}(\mathbf{o}_t) = \alpha\mathbf{l}^{gen}(\mathbf{o}_t) + \vec{\beta}, \tag{4.12}$$

where $\mathbf{l}^{gen}(\mathbf{o}_t)$ is the log-likelihood vector obtained from the GB. Through C_{llr} minimization we obtain the scalar α and the vector β that transform the scores coming from the GB block. This optimization is made via discriminative MLR.

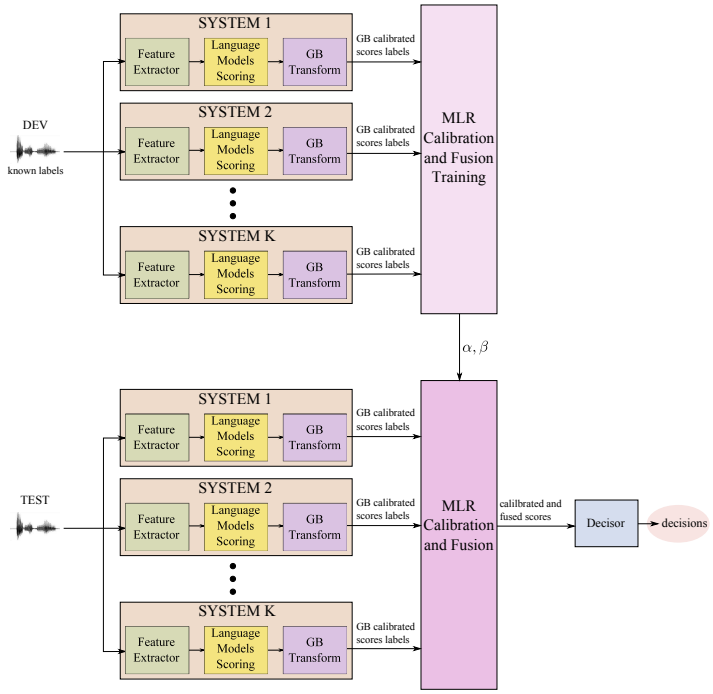


Figure 4.3: Fusion of K systems - The fusion is performed via discriminative MLR, and it also performs calibration. For training the fusion parameters, the *Dev* dataset is used.

4.2.3 Fusion

Imagine we have K LID systems, hopefully as complementary as possible, that we want to combine to build a better classifier. The strategy followed in this Thesis to combine them is to fuse the output scores given by the GB of each system. The previously explained discriminative calibration step will be used to perform this fusion. In Figure 4.3, we can see the architecture of a fusion system where K systems are fused. The training of the fusion is done with the *Dev* dataset. Our new discriminatively calibrated log-likelihoods will be a linear combination of the GB outputs of our K systems, $\mathbf{l_k}^{gen}$,

$$\mathbf{l}^{fus}(\mathbf{o}_t) = \sum_{k=1}^{K} \alpha_k \mathbf{l_k}^{gen}(\mathbf{o}_t) + \beta. \tag{4.13}$$

As we can check, the fusion is a generalization of the calibration of a single system (where $K = 1$), and since the fusion is also a calibration, because of the linearity of the operation, there is no need to pre-calibrate each input system, or to post-calibrate the

fusion [Brümmer, 2007].

4.3 Evaluation Metric

The primary metric used along this Thesis is C_{avg}. C_{avg} is a metric defined by NIST to evaluate systems submitted to NIST LREs. It has the meaning of a cost, so the lower, the better. As NIST defines a verification task in his evaluations, a single cost is computed for each target language against the rest of languages, and a final average of all the costs is computed as a single system performance metric. For a closed-set evaluation, this average metric is defined as [National Institute of Standards and Technology (NIST), 2007, 2009]

$$C_{avg} = \frac{1}{L}\sum_{L_T}\{C_{miss}\cdot\pi_{L_T}\cdot P_{miss}(L_T) + \sum_{L_{NT}} C_{fa}\cdot\pi_{L_NT}cdotP_{fa}(L_T, L_{NT})\}, \quad (4.14)$$

where L is the number of target languages, C_{miss} and C_{fa} are the costs of a miss and of a false alarm, respectively, π_{L_T} is the a priori probability of a target, $\pi_{L_NT} = (1-\pi_{L_T})/(L-1)$ is the a priori probability of non-target, L_T and L_NT are the target and non-target languages, and P_{miss} and P_{fa} are the probabilities of miss and false alarm obtained by our system.

We select the working point $C_{miss} = C_{fa} = 1$, and $\pi_{L_T} = \pi_{L_NT} = \frac{1}{L}$, a flat prior. Note that in NIST evaluations the target language prior is set to $\pi_{L_T} = 0.5$, because the problem is a binary detection task. However, we think that a flat prior over all target languages is a more realistic situation. Nonetheless, in Chapter 9, where we report results on the 2009 NIST LRE dataset, we will use both target language priors for ease of comparison with other works in the bibliography where NIST evaluation rules are followed.

4.4 Database

A database was designed to report the experiments on this Thesis by taking recordings from other databases. The idea was that all the results were given on the same dataset and were comparable. 6 languages of the 2011 NIST LRE evaluation dataset were included: English, Farsi, Hindi, Mandarin, Russian, and Spanish. The reason of this

	Total		**CTS**		LRE03		LRE05		LRE07t		CALLFR		OHSU		SRE04		SRE06		SRE08		**VOA3**	
Language	**#**	**hrs**	**#**	**hrs**	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	**#**	**hrs**
English	1412	21.67	1280	19.86	495	1.32	498	1.42	0	0	39	6.70	60	3.86	0	0	197	6.58	0	0	132	1.81
Farsi	1375	19.12	830	14.45	717	2.01	0	0	0	0	56	10.42	0	0	0	0	1	0.02	56	2.01	545	4.67
Hindi	1376	21.89	1153	17.51	385	1.02	385	1.08	0	0	0	0	72	4.72	0	0	23	0.84	288	9.85	223	4.38
Mandarin	1357	21.31	1068	19.99	349	0.92	300	0.85	0	0	20	3.71	20	1.31	138	4.90	141	4.99	100	3.32	289	1.32
Russian	1367	21.89	680	17.44	240	0.67	0	0	40	2.65	0	0	0	0	140	4.97	120	4.11	140	5.05	687	4.45
Spanish	1473	20.94	1173	17.64	450	1.27	449	1.21	0	0	26	4.80	60	3.78	145	5.15	37	1.24	6	0.21	300	3.30

Table 4.1: *Train* Data for LID.

	Total		**CTS**		LRE07e		LRE09cts		**BNBS**		LRE09bnbs		VOA3.3		VOA3.10		VOA3.30	
Language	**#**	**hrs**	**#**	**hrs**	#	hrs	#	hrs	**#**	**hrs**	#	hrs	#	hrs	#	hrs	#	hrs
English	300	0.97	98	0.37	60	0.25	38	0.12	202	0.60	22	0.07	60	0.04	60	0.12	60	0.37
Farsi	300	0.95	65	0.23	60	0.21	5	0.02	235	0.72	55	0.17	60	0.04	60	0.12	60	0.39
Hindi	300	0.96	79	0.31	60	0.24	19	0.07	221	0.65	41	0.13	60	0.04	60	0.11	60	0.37
Mandarin	310	1.00	102	0.39	65	0.22	37	0.17	208	0.61	28	0.08	50	0.03	65	0.12	65	0.38
Russian	300	0.92	90	0.33	60	0.21	30	0.12	210	0.59	30	0.10	60	0.04	60	0.11	60	0.34
Spanish	310	0.95	65	0.23	65	0.23	0	0	245	0.72	65	0.19	50	0.03	65	0.12	65	0.38

Table 4.2: *Dev* Data for LID.

	Total		total.3		total.10		total.30		**CTS**		cts.3		cts.10		cts.30		**BNBS**		bnbs.3		bnbs.10		bnbs.30	
Language	**#**	**hrs**	#	hrs	#	hrs	#	hrs	**#**	**hrs**	#	hrs	#	hrs	#	hrs	**#**	**hrs**	#	hrs	#	hrs	#	hrs
English	1356	3.44	452	0.29	683	1.71	221	1.44	363	1.03	121	0.06	121	0.22	121	0.75	993	2.40	331	0.22	562	1.49	100	0.69
Farsi	1215	3.73	405	0.24	406	0.81	404	2.68	591	1.85	197	0.12	197	0.40	197	1.33	624	1.88	208	0.12	209	0.41	207	1.35
Hindi	1254	3.24	418	0.26	621	1.54	215	1.44	210	0.62	70	0.04	70	0.13	70	0.45	1044	2.62	348	0.22	551	1.41	145	0.99
Mandarin	1296	3.78	432	0.26	504	1.18	360	1.34	777	2.29	259	0.15	259	0.50	259	1.64	519	1.49	173	0.11	245	0.68	101	0.70
Russian	1323	4.33	441	0.28	441	0.95	441	3.10	417	1.27	139	0.08	139	0.28	139	0.91	906	3.06	302	0.20	302	0.67	302	2.19
Spanish	1257	3.93	419	0.25	419	0.86	419	2.82	693	2.02	231	0.13	231	0.44	231	1.45	564	1.91	188	0.12	188	0.42	188	1.37

Table 4.3: *Test* Data for LID - 2011 NIST LRE test data.

selection was that those were the only languages for which we could collect 20 h for training and 1 h for development for each. We also wanted an experimental scenario where training and development data were balanced among target languages. The databases from which data were taken, hours of speech (after voice activity detection [VAD]), and number of utterances used for *Train*, *Dev*, and *Test* datasets are reflected in Tables 4.1, 4.2, and 4.3, respectively. VOA3 received an especial processing explained in Martínez et al. [2011c]. Basically, we only used segments marked as telephone by NIST, we removed English labeled as other language, and we removed repeated speakers within each language. The evaluation data was composed of conversational telephone speech (CTS) and broadcast narrowband speech (BNBS) [National Institute of Standards and Technology (NIST), 2011]. In order to cover both types of channel in training, VOA3 was BNBS and the rest of datasets were CTS, and in development, VOA3 and part of LRE09 were BNBS, and the other part of LRE09 and LRE07e was CTS. Note that neither LRE07 data nor VOA3 data used in development was seen in the *Train* dataset. Additionally, the VOA3 data used for development was segmented to obtain 3 s, 10 s,

and 30 s long files that match better with the *Test* dataset.

There are also experiments to study the influence of the number of hours of training and development data on the results. In the case of training, we built two smaller datasets, one with 7 h and one with 1 h per language, evenly distributed over the different databases and channels. In the case of development, we created a dataset of 4 h per language with the same database and channel proportions as the 1 h dataset. In both training and development cases, the smallest datasets were fully contained into larger ones. It will be indicated adequately when these datasets are used. Nonetheless, in most experiments, the standard procedure is to work with the 20 h and 1 h datasets.

5

Evaluation Methods

Contents

In this chapter, we explain different evaluation methods used with our linear Gaussian models. Referring to Figure 4.1, we explain how the box *Language Models Scoring* computes scores for each language. Specifically, we investigated the performance of GMMs, GMM with MAP adaptation, JFA, and *i-Vectors*. In addition, some dimensionality reduction techniques, train and test mismatch adaptations, and channel and noise compensation methods are presented. A good understanding of these models is fundamental to comprehend the results of this Thesis. In Appendix A, the training of these models is detailed.

5.1 Gaussian Mixture Model

A GMM is trained independently for each language from scratch, and during evaluation, the log-likelihood of each utterance is obtained for each model. The likelihood function of a GMM can be seen in eq. (2.16). Those log-likelihoods are the scores passed to the calibration step. By using GMMs, Figure 4.1 is converted into Figure 5.1. In this Figure, we have omitted the development branch for clarity, but the GMM scoring block would be identical to one in the test branch.

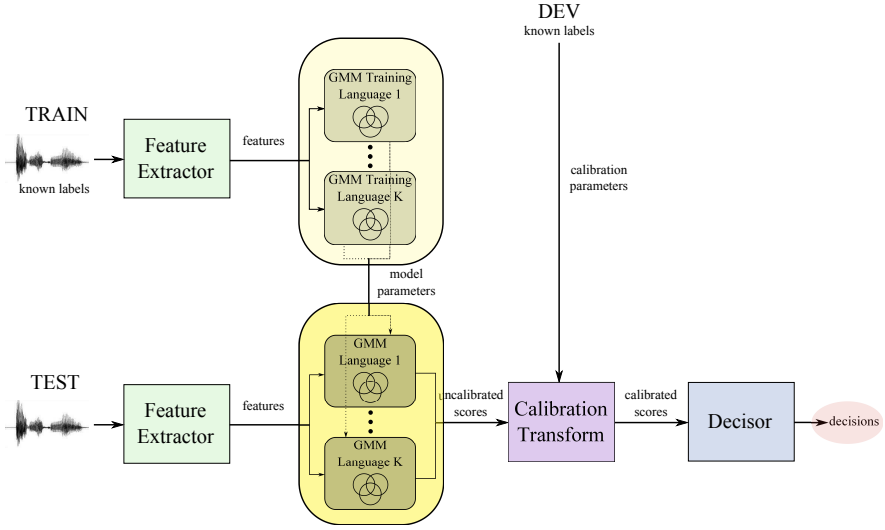


Figure 5.1: GMM System Architecture - A GMM is trained for each language. During evaluation, each model is scored for every test utterance. The development branch is omitted for clarity, but the GMM scoring block would be identical to the one in the test branch.

In our experiments, a full-covariance GMM is trained for every language. We have followed a splitting strategy, where we start with one Gaussian trained with 10 iterations of the EM algorithm (for one Gaussian it is not really necessary an iterative process). Then, we split the resulting Gaussian in two, and run other 10 iterations of the EM algorithm. Next, every Gaussian is split in other two, and 10 iterations of the EM algorithm are run. We repeat this until we reach the desired number of Gaussians. This is a splitting strategy with $H = 2$, as explained in Section A.4.1 of Appendix A.

5.2 MAP Gaussian Mixture Model

In this case, first we train a full-covariance GMM UBM with all training data including all languages. Second, for each target language, we adapt the UBM via MAP with the training data of that language. The covariances are not adapted, and consequently, are shared among all languages. During evaluation, every test utterance is evaluated on the adapted GMMs to obtain a log-likelihood for each language, as in the GMM case. MAP is normally used when the amount of data of each target language is not enough to robustly train a model. By starting with a UBM, we benefit from data of other languages, and more robust parameters are trained, especially for the case of the covariances, which are more sensitive to data scarcity. This method can be considered to be an intermediate step between GMM and JFA, because we make MAP adaptation from a GMM, like in JFA, but we do not train a channel subspace, like in GMM. Therefore, the comparison with JFA is fairer with this approach than with GMMs.

The training of the UBM is done with the EM algorithm with the same splitting strategy followed for GMMs explained in previous section. The MAP adaptation is made with the zeroth and first order accumulated statistics of each language, as per eq. (2.36), and the relevance factor is $r = 14$. In experiments not reported in this Thesis, we checked that the results are stable for a wide range of values of r. We chose a central value in this interval.

The architecture of this system is shown in Figure 5.2. Note that the training is divided in two steps, but there is a single training dataset which is used in both. The development branch is omitted to make the figure clearer, but the GMM scoring block would be identical to one in the test branch.

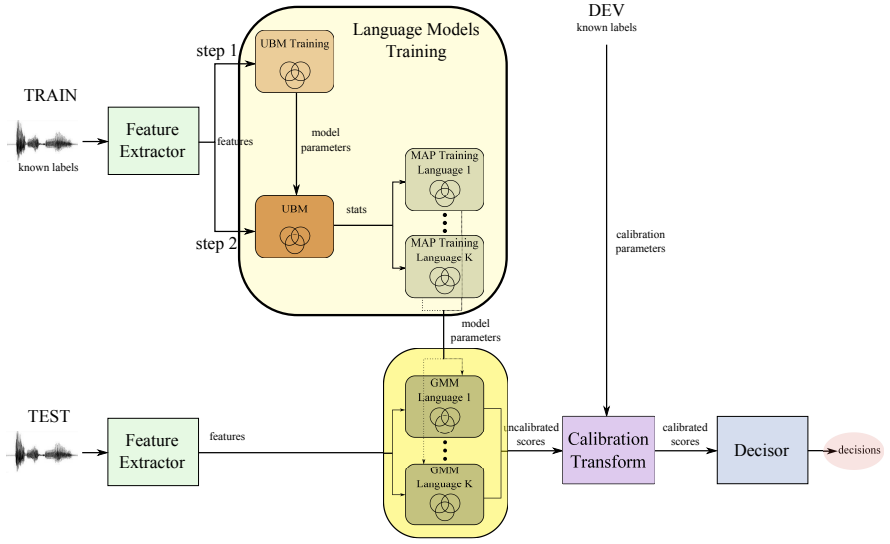


Figure 5.2: MAP GMM System Architecture - First, a UBM is trained, and second, a GMM per language is built by adapting the UBM with the zeroth and first order statistics of the corresponding language. During evaluation, each GMM is scored for every test utterance. The development branch is omitted for clarity, but the GMM scoring block would be identical to the one in the test branch.

5.3 Joint Factor Analysis

The training of JFA also has several steps. First, a GMM UBM is trained with all training data. As in previous sections, we train a full-covariance model, with a splitting strategy with $H = 2$. The UBM allows obtaining fixed alignments, and computing sufficient statistics. These statistics are the input to the next step, in which we adapt the UBM means to each language via MAP with eq. (2.36) with $r = 14$. Finally, we compute the channel loading matrix, $\mathbf{U}$. We calculate a unique channel subspace, common to all language models, as per eq. (A.80). $\mathbf{U}$ is computed by running 20 iterations of the EM algorithm, alternating between an ML step and an MD step, as explained in Section 2.11 of Appendix A. Means and covariances are not updated. Note that the covariances of the UBM are full, but for the JFA model, we only keep the diagonal part. Also note that the covariances are shared among language models.

At the end, we have a JFA model for each language, with shared channel subspace and covariances, and different means. During evaluation, every test utterance is evaluated against each JFA model, and the resulting log-likelihoods are passed to the

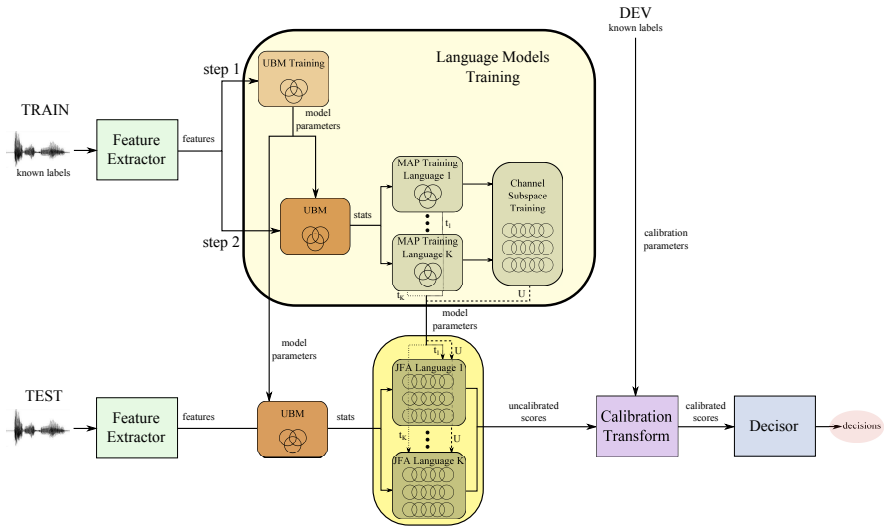


Figure 5.3: JFA System Architecture - First, a UBM is trained, and second, MAP adaptation is done for each language from the UBM with the zeroth and first order supervectors of statistics, and a channel subspace common to all languages, **U**, is trained. During evaluation, each JFA model is scored for every test utterance. The development branch is omitted for clarity, but the JFA scoring block would be identical to the one in the test branch.

calibration block. The architecture of a JFA system is shown in Figure 5.3. We have separated the UBM training from the MAP adaptation and from the channel subspace training. The development branch is omitted to keep the figure uncluttered.

5.3.1 Evaluation of JFA

Different JFA scoring techniques have been developed in the literature, aiming at giving the best accuracy in the lowest computational time [Glembek et al., 2009]. In this section we review the most important ones. The final scores obtained for each class are given in terms of log-likelihood ratio (LLR) between the log-likelihood given by the model of each language and the log-likelihood given by the UBM. Actually, the final score of an utterance is given as the average LLR of all frames in that utterance. In all the following scoring methods, the final LLR for language l is computed, for file j with N_j frames, as

$$LLR_l(j) = \frac{1}{N_j} \ln p(\mathbf{O}(j)|\Theta_l) - \ln p(\mathbf{O}(j)|\Theta_{UBM}), \tag{5.1}$$

with Θ_l being the model parameters of language l, Θ_{UBM} being the model parameters of the UBM, and $\mathbf{O}(j)$ being D-dimension features extracted for file j.

In the next paragraphs we review the principal JFA scoring techniques from the most to the least accurate as per the approximations made in their formulation.

5.3.1.1 Frame by Frame

The full evaluation of the model is infeasible even with a few number of Gaussian components or a few frames. This would correspond to TMFA, presented in Section 2.10 of Appendix A.

5.3.1.2 Integrating over Channel Distribution

In JFA, we approximate the combinatorial explosion of TMFA by assuming that there is only one path over the different possible combinations of Gaussian components, but this path is evaluated softly, in the sense that all Gaussian components partly contribute to the generation of each frame. Thus, the closest solution to the exact TMFA calculation given by JFA is to integrate over the channel distribution while keeping the fixed soft alignments given by the UBM,

$$p(\mathbf{O}|\Theta_l) = \int p(\mathbf{O}|\mathbf{X},\Theta_l)p(\mathbf{X})d\mathbf{X} = \int p(\mathbf{O}|\mathbf{X},\Theta_l)\mathcal{N}(\mathbf{0},\mathbf{I})d\mathbf{X}, \tag{5.2}$$

with $\mathbf{X}$ being the channel factor, and $\Theta_l = \{\mathbf{t}_l, \mathbf{\Sigma}, \mathbf{U}\}$ being the model parameters mean supervector, covariance supermatrix, and channel loading matrix of language l, respectively. This equation can be solved analytically as shown in Kenny et al. [2005]. The solution for utterance j is given in terms of supervectors of sufficient statistics obtained over the UBM [Kenny et al., 2007],

$$\begin{aligned}\ln p(\mathbf{O}(j)|\Theta_l) = \sum_{k=1}^{K} N_k(j) \ln \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}_k|^{\frac{1}{2}}} - \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}^{-1}\hat{\mathbf{S}}(\mathbf{j})) \\ -\frac{1}{2}\ln|\mathbf{L}| + \frac{1}{2}||\mathbf{L}^{-\frac{1}{2}}\mathbf{U}^{\intercal}\mathbf{\Sigma}^{-1}\hat{\mathbf{F}}(\mathbf{j})||^2,\end{aligned} \tag{5.3}$$

where $\mathbf{L}$ is defined in eq. (A.72), $\mathbf{L}^{\frac{1}{2}}$ is its Cholesky decomposition, and

$$\hat{\mathbf{F}}(j) = \mathbf{F}(j) - \mathbf{N}(j)\mathbf{t}_l, \tag{5.4}$$

$$\hat{\mathbf{S}}(j) = \mathbf{S}(j) - 2\operatorname{diag}(\mathbf{F}(j)\mathbf{t}_l^{\intercal}) + \operatorname{diag}(\mathbf{N}(j)\mathbf{t}_l\mathbf{t}_l^{\intercal}), \tag{5.5}$$

with supervectors of sufficient statistics $\mathbf{N}(j)$, $\mathbf{F}(j)$, and $\mathbf{S}(j)$ defined in eqs. (2.40-2.42), respectively. Note that the terms 1 and 3 in eq. (5.3), and the first term in eq. (5.5) cancel out in the LLR computation.

5.3.1.3 Channel Point Estimate

In this strategy we avoid the integration over the channel distribution by obtaining a fixed value for the channel factor. This fixed value is the MAP point estimate of the channel factor $\mathbf{x}$. The log-likelihood function is given by eq. (A.67). Again, fixed soft alignments given by the UBM are considered, and the equation is a lower bound to the true log-likelihood, since the Kullback-Leibler (KL) divergence of the true posterior alignments and the alignments given by the UBM will be greater than 0. The analytical form of A.67 is, as shown in *Theorem 1* of Kenny [2006],

$$\begin{aligned} \ln p(\mathbf{O}(j)|\mathbf{X},\Theta_l) = \sum_{k=1}^{K} N_k(j) \ln \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}_k|^{\frac{1}{2}}} \\ -\frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}^{-1}\mathbf{S}(j)) + \mathbf{M}_l^{\intercal}(j)\mathbf{\Sigma}^{-1}\mathbf{F}(j) + \frac{1}{2}\mathbf{M}_l^{\intercal}(j)\mathbf{N}(j)\mathbf{\Sigma}^{-1}\mathbf{M}_l(j), \end{aligned} \tag{5.6}$$

where $\mathbf{M}_l(j)$ is the channel compensated GMM mean for language l and utterance j defined in eq. (2.35), and the rest of variables were defined in the previous section. Note that in the LLR computation, the terms 1 and 2 cancel out.

The MAP point estimate of the channel factors, $\mathbf{x}$, is done over the model of the language in the numerator of the LLR, and over the model of the UBM in the denominator of the LLR. A simplification that we investigated was to use the UBM MAP point estimate for both the model of the language and the UBM [Glembek et al., 2009]. This reduces the computational cost because only one vector of channel factors is calculated for each utterance. In the following approaches presented in Sections 5.3.1.4, and 5.3.1.5 this simplification was also investigated.

5.3.1.4 Channel Point Estimate with UBM weights

In this approach, we use UBM weights instead of UBM soft assignments to compute the log-likelihood $\ln p(\mathbf{O}(j)|\mathbf{X},\Theta_l)$ [Glembek et al., 2011]. To make it clearer, instead of expressing the log-likelihood in terms of supervectors of statistics, we will express it in

terms of the features $\mathbf{O}(j)$, and the approximation sign indicates that soft alignments were substituted by UBM weights,

$$\begin{aligned}\ln p(\mathbf{O}(j)|\Theta_l) &= \frac{1}{N_j}\sum_{n=1}^{N_j}\ln\sum_{k=1}^{K}\gamma_n^j(z_k)\mathcal{N}(\mathbf{o}_n(j)|\mathbf{t}_{lk}+\mathbf{U}_k\mathbf{x}(j),\mathbf{\Sigma}_k)\\ &\approx \frac{1}{N_j}\sum_{n=1}^{N_j}\ln\sum_{k=1}^{K}\omega_k\mathcal{N}(\mathbf{o}_n(j)|\mathbf{t}_{lk}+\mathbf{U}_k\mathbf{x}(j),\mathbf{\Sigma}_k),\end{aligned} \tag{5.7}$$

where K is the number of Gaussian components in the mixture, $\gamma_n^j(z_k)$ is the assignment made to Gaussian component k for frame n of utterance j, and $\mathbf{x}(j)$ is the MAP point estimate of the channel factor of that utterance.

5.3.1.5 Linear Scoring

Let us define the channel-compensated UBM for utterance j as

$$\mu_c(j) = \mu + \mathbf{U}\mathbf{x}(j). \tag{5.8}$$

Then, we move the origin of the supervector space to $\mu_c(j)$, and our model is expressed as

$$\mathbf{M}_{lc}(j) = \mathbf{M}_l(j) - \mu_c(j), \tag{5.9}$$

and the first order channel-compensated sufficient statistics as

$$\mathbf{F}_c(j) = \mathbf{F}_l(j) - \mathbf{N}(j)\mu_c(j). \tag{5.10}$$

Now, after omitting the terms that cancel out in the LLR computation, eq. (5.6) can be expressed as,

$$Q(\mathbf{O}(j)|\mathbf{X},\Theta_l) = \mathbf{M}_{lc}(j)^\intercal\mathbf{\Sigma}^{-1}\mathbf{F}_c(j) + \frac{1}{2}\mathbf{M}_{lc}(j)^\intercal\mathbf{N}(j)\mathbf{\Sigma}^{-1}\mathbf{M}_{lc}(j). \tag{5.11}$$

If we approximate equation 5.11 by its first order vector Taylor expansion as a function of $\mathbf{M}_{lc}(j)$, only the first term is kept, leading to

$$Q(\mathbf{O}(j)|\mathbf{X},\Theta_l) = \mathbf{M}_{lc}(j)^\intercal\mathbf{\Sigma}^{-1}\mathbf{F}_c(j). \tag{5.12}$$

Since the compensated UBM is now the coordinate origin, or in other words, it is a vector of zeros, the LLR of this approach is directly

$$LLR_l(j) = \frac{Q(\mathbf{O}(j)|\mathbf{X},\Theta_l)}{N_j} \tag{5.13}$$

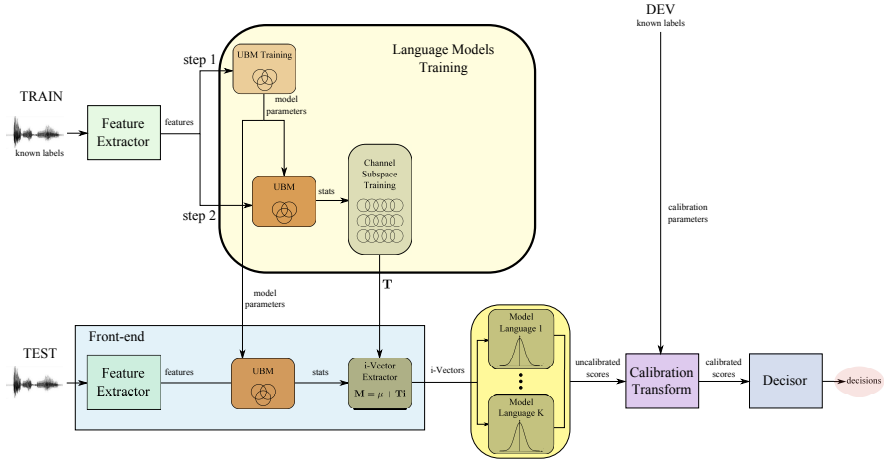


Figure 5.4: *i-Vector*-Based System Architecture - First, a UBM is trained, and second, a JFA model is trained and used as feature extractor. We build models for each language with *i-Vectors*, which are used to score the *Test* dataset. The development branch is omitted for clarity, but the *i-Vector* scoring block would be identical to the one in the test branch.

5.4 Total Variability Subspace: *i-Vectors*

The training of JFA as feature extractor is made with the very same configuration as the training of JFA as classifier explained in the previous section. The difference between both is that, in the use as feature extractor, we train a single model with all data of all languages. More details about the training can be seen in Section 2.12 of Appendix A. Once we extract *i-Vectors*, we use them for the classification of languages. In Figure 5.4, we present the architecture of a LID system based on *i-Vectors*. As we can see, the *i-Vector* extraction can be seen as part of the feature extraction process. The development branch is omitted to keep the figure clearer.

In the next paragraphs we present different evaluation methods built upon i-Vectors. We divide them into generative and discriminative classifiers. In the first, the goal is to build probabilistic models that generated the training i-Vectors of each language. In this group we can find ML Gaussian classifier and CSS (CSS) classifier. In the second, the goal is to minimize the classification error in the training data. In this group we can find SVM and LR.

Also, notice that i-Vectors contain all sources of variability, included the desired information, i.e. language, but also undesired information like channel. In Section

5.4.2, we present some methods to compensate channel mismatch in i-Vectors.

5.4.1 Evaluation of i-Vectors

5.4.1.1 Gaussian Classifier

The i-Vectors of each language are modeled with a Gaussian distribution with full-covariance matrix shared among languages. This covariance is equal to the WC covariance matrix of the training data. During evaluation, every utterance is scored against the Gaussian models of all languages, and the obtained log-likelihoods, given in eq. (4.4), are used for classification. This approach was successfully applied in Martínez et al. [2011c], and nowadays, this classifier is used in many state of the art systems.

If log-likelihoods were directly used to decide about the language, the quadratic term (third term in RHS of eq. (4.4)) could be ignored because it is independent of the class, given that the covariance is shared among classes. This would lead to a linear classifier as the remaining terms are only linear in the i-Vector. In our case, however, the log-likelihoods are used as inputs to another classifier, the calibration block described in Section 4.2. For this reason, we include the quadratic term, and thus, we avoid the i-Vector- (and therefore utterance-) dependent shift in our scores. Note that this is opposite to the GB described in Section 4.2.1, where we dropped this term for simplicity.

The heteroscedastic version of this classifier trains a different covariance matrix for each language. We also investigated this approach, although we advance that it is less efficient, unless you have a huge amount of data [Lopez-Moreno et al., 2014]. Also, we considered modeling each language with a GMM, but they only outperformed the Gaussian classifier when the amount of data was big enough [Lopez-Moreno et al., 2014], as could be expected after checking that the heteroscedastic version did not improve the results either.

5.4.1.2 Cosine Similarity Scoring

This method measures the angle between two i-Vectors [Dehak et al., 2011b]. If the two vector are in the same direction, the cosine of the angle between them will be 0° and the scoring will be 1. Otherwise, if they point to perpendicular directions, their cosine will be 90° and the scoring will be 0. If they are pointing to opposite directions,

the scoring will go to -1. We calculate the CSS between two vectors **a** and **b** as the cosine of the angle between them, Θ,

$$CSS = cos(\Theta) = \frac{a \cdot b}{||a||||b||}. \tag{5.14}$$

Note that only the angle between the i-Vectors is considered, and not their magnitudes. It is believed that the magnitude of the i-Vectors affects channel or session information, but not language, so removing the magnitude in scoring is expected to increase the robustness of the system [Dehak et al., 2010].

In our experiments, one of the vectors of the formula above was the mean i-Vector of the evaluated language, and the other was the i-Vector of the test utterance.

5.4.1.3 Support Vector Machine

SVM is a discriminative classifier that constructs a hyperplane to separate data points from different classes, with a margin between points from different classes as wide as possible [Vapnik, 1995]. It only uses certain points named support vectors. In our experiments, we built binary classifiers in a one-versus-all strategy with the software LIBSVM [Chang and Lin, 2011]. We used a linear kernel of the form $\mathbf{v}' \cdot \mathbf{u}$, for input vectors **v** and **u**, and C-support vector classification (C-SVC) [Chang and Lin, 2011], where the only tunable parameter is C. C is the regularization parameter. The higher C, the lower the error that is allowed, because we give more importance to classify all training data correctly, and a low C results in a more flexible hyperplane that tries to minimize the margin error in training, at the expense of having more missclassifications. Then, $C = \infty$ would correspond to having all training data correctly classified. In addition, we used L2 regularization.

5.4.1.4 Logistic Regression

LR is a discriminative and linear classifier [Bishop, 2006]. The main difference with SVM is the loss function to be minimized. Whereas in SVM, we maximize the margin between the hyperplane and the training data points, in LR we maximize the probability of the correct class given in the form of a logistic function. Note that, in spite of its name, LR is a classifier, not a regressor, and it is linear because the input to the logistic function is the dot product between the two input kernel vectors. Hence, a linear hyperplane is built in the kernel space. C is the only tunable parameter and has

the same meaning as for SVM. In addition, we used L2 regularization. We built binary classifiers in a one-versus-all strategy with the software LIBLINEAR [Fan et al., 2007].

5.4.2 Channel Compensation for i-Vectors

In the next paragraphs different techniques to compensate i-Vectors for undesired mismatches between train and test caused, for example, by channel or noise, are presented.

5.4.2.1 Centering and Whitening

Our model assumes that our i-Vectors follow a Gaussian distribution. However, this can not be always an exact assumption. i-Vectors centering and whitening have the objective of obtaining i-Vectors that follow more faithfully a Gaussian distribution [Garcia-Romero and Espy-Wilson, 2011]. Centering is just the removal of the global mean of the i-Vectors, computed with the training data, in such a way that the new mean of centered i-Vectors will the coordinate origin. Once they are centered, we apply the whitening, a decorrelation transform that converts the global covariance of the training *i-Vectors* into the identity matrix. Now, i-Vectors are centered and evenly distributed around the origin. Centering and whitening are usually a previous step to length normalization, because if we apply length normalization without a previous whitening, all the i-Vectors would fall in a small region of a hypersphere, and they would lose their discriminative power.

5.4.2.2 Length Normalization

Length normalization means that all i-Vectors are normalized to have unit length, and therefore, they will be projected into the unit hypersphere. The benefit of this transformation is that the mismatch between training and testing i-Vectors is reduced [Garcia-Romero and Espy-Wilson, 2011]. The length normalization is computed for an i-Vector $\mathbf{i}$ as

$$\hat{\mathbf{i}} = \frac{\mathbf{i}}{||\mathbf{i}||} \tag{5.15}$$

5.4.2.3 Linear Discriminant Analysis

LDA is a technique to separate as much as possible data belonging to different classes [Fisher, 1936; Bishop, 2006]. The idea is to project data into a space where the means

of the classes are as much separated as possible, whilst keeping the variance of each class as small as possible. The transform, given by the orthogonal matrix $\mathbf{W}_{\max}$, is calculated by maximizing the following expression,

$$\mathbf{W}_{\max} = \underset{\mathbf{W}}{\operatorname{argmax}}\, J(\mathbf{W}) = \underset{\mathbf{W}}{\operatorname{argmax}}\, \frac{\mathbf{W}^\intercal \mathbf{S}_{BC} \mathbf{W}}{\mathbf{W}^\intercal \mathbf{S}_{WC} \mathbf{W}}, \tag{5.16}$$

where $\mathbf{S}_{WC}$ is the WC class scatter matrix defined in eq. (4.3), and $\mathbf{S}_{BC}$ is the between-class (BC) scatter matrix defined as

$$\mathbf{S}_{BC} = \frac{1}{N} \sum_{l=1}^{L} (\bar{\phi}_l - \bar{\phi})(\bar{\phi}_l - \bar{\phi})^\intercal, \tag{5.17}$$

with $\bar{\phi}$ being the global mean of the data, $\bar{\phi}_l$ being the mean of language l, N is the total number of training data points, and L being the number of classes.

The technique is useful for classification, but also as dimensionality reduction, because after applying the transform, we obtain a space of dimension the number of classes minus one. In our case, each class corresponds to a language. The projections with maximum separability are the eigenvectors of $\mathbf{S}_{WC}^{-1}\mathbf{S}_{BC}$ with highest eigenvalues. Once we apply the transform, we hope that only the important dimensions are kept while the noisy ones are discarded. If $\mathbf{W}$ is the projection matrix, an i-Vector would be transformed as $\hat{\mathbf{i}} = \mathbf{W}^\intercal \mathbf{i}$.

5.4.2.4 Within-Class Covariance Normalization

WCCN was presented in Hatch et al. [2006]; Hatch and Stolcke [2006] as a method to minimize the expected rate of false acceptances and false rejections in an SVM-based speaker recognition classifier. The physical idea behind is to normalize all the classes by the same within class covariance matrix to remove undesired class-independent variability from the signal. Note that our definition of the covariance matrix in eq. (4.3) is slightly different but equivalent to the one in Hatch et al. [2006]. We can express $\mathbf{S}_{WC} = \mathbf{B}\mathbf{B}^\intercal$, with $\mathbf{B}$ being the Cholesky decomposition of $\mathbf{S}_{WC}$. Then, each i-Vector is transformed as $\hat{\mathbf{i}} = \mathbf{B}^\intercal \mathbf{i}$.

5.4.2.5 Nuisance Attribute Projection

NAP is a technique that finds and removes nuisance directions [Solomonoff et al., 2004]. A nuisance attribute is defined as some quality or attribute that affects the appearance

of an observation without being correlated with the classes to be classified. In order to remove such directions, we consider, as in WCCN, the WC covariance matrix as the description of the directions within a class that must be minimized. These are considered to be class-independent. To remove such directions we apply the projection $\mathbf{P} = \mathbf{I} - \mathbf{R}\mathbf{R}^\intercal$ on i-Vectors. $\mathbf{R}$ is a matrix with columns equal to the eigenvectors with highest eigenvalues of $\mathbf{S}_{WC}$. The i-Vectors are transformed as $\hat{\mathbf{i}} = \mathbf{P}\mathbf{i}$.

6

Acoustic LID Results

Contents

In this chapter, the performance of LID systems with acoustic features is analyzed on the *Test* dataset presented in Section 4.4. We studied GMM, MAP GMM, JFA and *i-Vector* approaches, described in Chapter 5. It is shown how the performance with GMMs was improved by introducing a subspace to model the undesired variability among different files. However, as it already happened in the field of speaker recognition [Dehak, 2009], we observed that this subspace still contained language information, and the *i-Vector* approach arose as an alternative to avoid this problem. In addition, *i-Vectors* allow building simple classifiers thanks to their fixed-length and low-dimensionality, and nowadays, they are the state of the art technology in LID.

6.1 Acoustic Features

The term acoustic feature is commonly used to refer to parameters that contain spectral information. In this type of features, speech information is extracted at different frequency bands. MFCC and PLP coefficients are the most extended ones. In this Thesis, we used MFCCs. MFCCs are computed by first calculating the fast Fourier transform (FFT) of the signal; then we map the FFT into a mel filter bank, a set of overlapping triangular filters distributed along the speech spectrum according to the mel scale, which tries to approximate the human ear response; we take the logarithm of the filtered spectrum; and finally a DCT of the resulting log values is calculated to decorrelate the feature elements. We used 25 mel filters. The rate of extraction was 10 ms, and the window length in which MFCCs were computed was 25 ms. A VAD was used to keep only speech frames. In the next paragraphs, we explain different feature normalization techniques applied to remove as much as possible channel, noise, and speaker information while keeping language information, and also how and when to extract the SDC features. The schematic process from the audio file to the MFCC+SDC features is depicted in Figure 6.1.

6.1.1 Feature Normalization

Several techniques are used to make the acoustic features as independent as possible of speaker and channel information, in such a way, that only language information is preserved. In our work, we use three different techniques: VTLN, CMVN, and RASTA filtering.

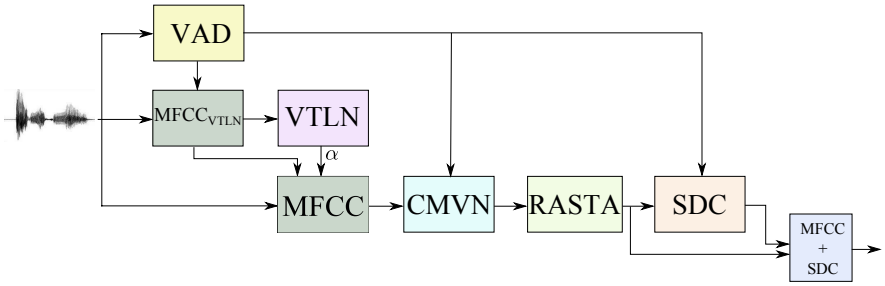


Figure 6.1: Acoustic features extraction process - MFCCs are computed with vocal tract length normalization (VTLN). Then, cepstral mean and variance normalization (CMVN) and RASTA filtering are applied to MFCCs. Finally, SDC are computed and concatenated to MFCCs using the VAD output to filter out nonspeech frames. For VTLN, MFCCs with different configuration ($\text{MFCC}_{\text{VTLN}}$) are computed.

6.1.1.1 Vocal Tract Length Normalization

In Bamberg [1981], it was investigated the relationship between the vocal tract length and the formant frequencies. For a vocal tract of length L, each formant frequency will be proportional to $\frac{1}{L}$. Then, it is desired to rescale the frequency axis to make speech of all speakers appear as if it was produced by a vocal tract with a single standard length [Wegmann et al., 1996]. There are different methods to accomplish this task. We follow an approach similar to the ones proposed in Wong and Sridharan [2002]; Sarkar et al. [2010]. The goal is to find the scaling factor, α, for each speaker. This is achieved by first training a GMM for each possible scaling factor, as follows:

- Build a UBM GMM with the MFCCs of all training data. These MFCCs have been previously computed only for VTLN purposes ($\text{MFCC}_{\text{VTLN}}$).
- Sweep all possible values of α for each file in the training dataset. For each α, evaluate the rescaled version of the file on the UBM. For each file, select the α that obtains maximum likelihood.
- Group the files with the same α together. Perform a MAP adaptation over the UBM with each group. You will obtain a new GMM for each α.

During the feature extraction process, each file is evaluated without rescaling over all adapted models. The α corresponding to the model that obtains the highest likelihood is selected to rescale that file. Then, MFCCs with the configuration for classification (25

mel filters, 25 ms long windows, 10 ms window shift) are computed using the selected α. In our experiments, we built models from $\alpha = 0.85$ to $\alpha = 1.15$ in steps of 0.02. We remark that the MFCCs computed to select α and the final MFCCs computed with the selected α used for classification can be obtained with different configuration. In our system, both configurations are the same. The main difference is the obvious VTLN effect, because the first ones are not rescaled by any α and the second ones are rescaled by the selected α. This is clearly seen in Figure 6.1.

6.1.1.2 Cepstral Mean and Variance Normalization

CMVN [Viikki and Laurila, 1998; Barras and Gauvain, 2003] is a double transformation applied, in our case, in a per-file basis. We have to bear in mind that only the frames detected by the VAD are used. First, the mean of the MFCCs of the current file is subtracted to every frame of the same file. The idea behind mean subtraction is that, if the channel is constant, it will be reflected in the cepstral mean. Thus, it is a method to primarily remove the channel contribution. Second, the variance of the MFCCs is calculated over the whole file, and each frame, after mean subtraction, is divided by the standard deviation. This allows removing noise effects and consequently having a better match between training and testing features. Note that after these transformations, all features will have zero mean and unit variance.

6.1.1.3 RASTA

RASTA filtering is a band pass filtering applied over the resulting normalized MFCCs [Hermansky and Morgan, 1994]. The goal is to remove any slow variation in the short-term spectrum. Even though humans are not sensitive to slow varying components in the speech signal caused by the communication channel or to steady background noise present in the environment, because these does not impair human speech communication, it makes good engineering sense. Mainly, it is useful to alleviate the effect of convolutional noise and to smooth the fast frame-to-frame spectral changes present in the short-term spectral estimate due to analysis artifacts [Hermansky and Morgan, 1994]. In other words, it keeps the modulation frequencies more important for human speech communication, which are normally between 0.26 Hz and 50 Hz.

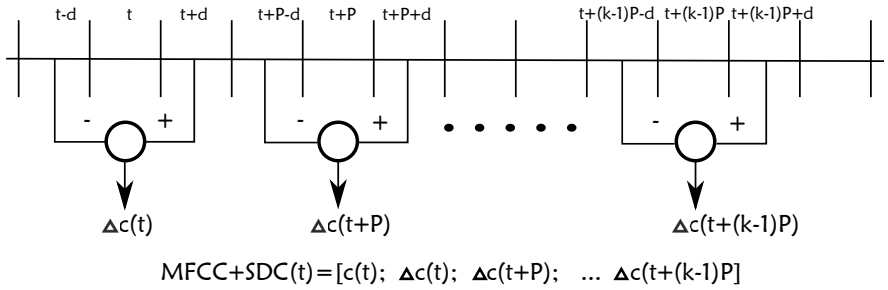


Figure 6.2: SDC computation scheme - SDC features stacked with MFCC at frame t for parameters N-d-P-k.

6.1.1.4 Shifted Delta Cepstra Coefficients

After the normalization process, a common practice is to calculate the first and second derivatives of the MFCC, in order to compensate for the independence assumption among frames. However, Torres-Carrasquillo obtained better results in a LID task with SDC [Torres-Carrasquillo et al., 2002b], a stacking of future MFCC derivatives onto the current frame derivative. The reason of this success was that these features also capture long-term information, and this was proved to be beneficial for LID. Four parameters define SDC: N, d, P, and k. N is the number of cepstral coefficients computed at each frame, d represents the time shift for the delta computation, P is the shift between consecutive blocks, and k is the number of blocks whose delta coefficients are concatenated. We used a 7-1-3-7 configuration, and stacked also MFCCs of the current frame, to obtain a 56-dimension vector every 10 ms. Only the frames detected by the VAD were conserved. In Figure 6.2, it is shown a schematic view of how SDC coefficients are computed.

6.2 Results

6.2.1 GMM

In this section, we test the LID approach described in Section 5.1 based on GMM.

In Figure 6.3, we evaluate the GMM for different number of Gaussians, and we compare the results obtained with the *Dev* dataset including 1 h of data per language (straight green lines) and the results obtained with a *Dev* dataset of 4 h of data per language (striped black lines). The first clear observation is that the results for the 30

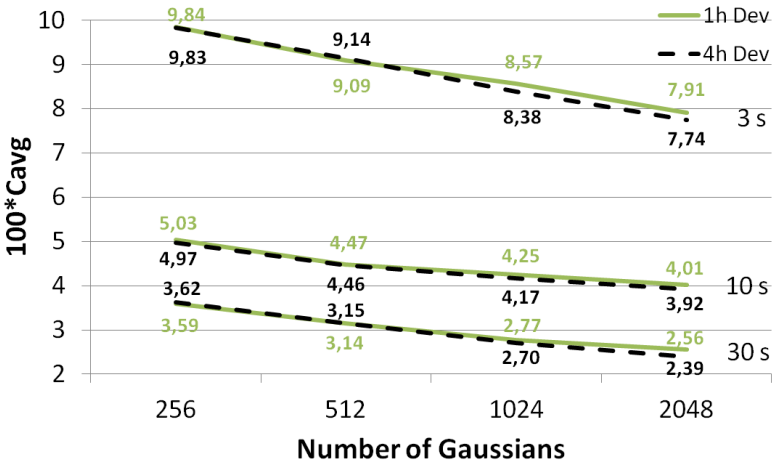


Figure 6.3: Performance of a GMM-based LID system as a function of the number of components - Results with 20 h for training and 1 h (straight green lines) and 4 h (striped black lines) for development, for 3 s, 10 s and 30 s tasks. Results in terms of $\mathbf{100 \cdot C_{avg}}$.

s task were much better than for the 10s and 3 s tasks, and the results of the 10 s task were much better than for the 3 s task. This trend was observed in all the experiments of the Thesis. Nonetheless, it is a general characteristic of speech technologies: the longer the test audio segment, the better the results. We have more information. Second, the error decreased as we increased the number of Gaussians. Third, as we increased the amount of development data, C_{avg} was slightly reduced. For example, with a GMM with 2048 components, we obtained $100 \cdot C_{avg} = 7.91$ and $100 \cdot C_{avg} = 7.74$, with 1 h and 4 h of development per language, respectively, for the 3 s task; for the 10 s task, we obtained $100 \cdot C_{avg} = 4.01$ and $100 \cdot C_{avg} = 3.92$, with 1 h and 4 h of development per language, respectively; and for the 30 s task, we obtained $100 \cdot C_{avg} = 2.56$ and $100 \cdot C_{avg} = 2.39$, with 1 h and 4 h of development per language, respectively. For the rest of the experiments in the Thesis, we used the *Dev* dataset with 1 h of data per language.

The goal of the next experiment was to analyze the influence that the amount of training data has in system performance. In Figure 6.4, we show C_{avg} for three different systems trained on 1 h, 7 h, and 20 h per language. We include results for GMMs with 256, 512, 1024 and 2048 Gaussian components. The main idea gathered from this experiment is that, in general, the more the training data, the better the system perfor-

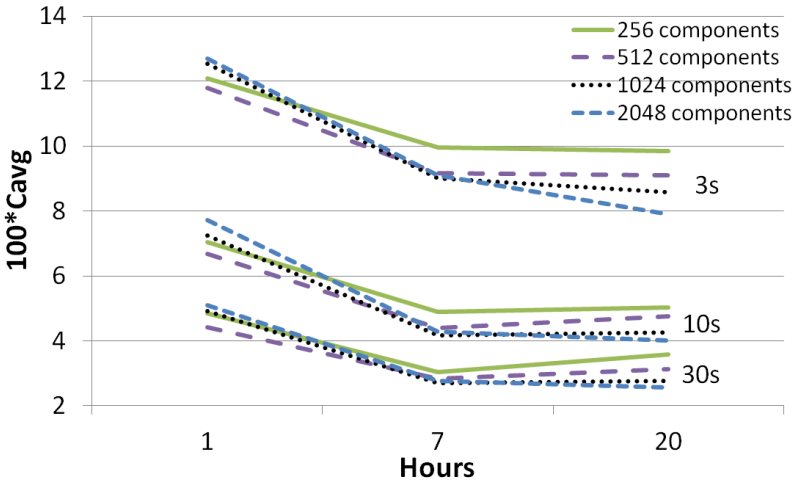


Figure 6.4: Performance of a GMM-based LID system as a function of the training time per language - 256 (straight green lines), 512 (long striped purple lines), 1024 (dotted black lines), and 2048 (short striped blue lines) Gaussians for the 3 s, 10 s, and 30 s tasks, 1 h development dataset. Results in terms of $\mathbf{100 \cdot C_{avg}}$.

mance. As we increased the number of hours of training, more complex systems, with higher number of parameters, could be robustly trained. This is, in general, a property of ML methods. Increasing the amount of data for training has as a consequence that the test samples have more chances of having been seen during training. However, this is not always true. We can see one exception in these experiments in some cases for the 10 s and 30 s tasks, where the performance degraded when passing from 7 h to 20 h for training with models with less than 2048 components. We think that this is due to the type of data used for training, which fitted better to the *Test* dataset in the case of 7 h than in the case of 20 h, in spite of our efforts to build datasets with the same database and channel proportions (see Section 4.4). In the 10 s and 30 s tasks, only the model with 2048 Gaussian components benefited from having more hours of training data. One last point we want to remark is that, when we used only 1 h for training, it was not the system with 2048 Gaussians the one performing the best, but the one with 512 Gaussians. This indicates that 1 h was not enough to robustly train a large number of parameters.

In order to see the effectiveness of the two blocks that form the calibration step, we tested the same GMM-based LID system with and without each of them. The results are in Table 6.1. It is evident that the most effective solution was the cascade of the

	GB & MLR	noGB & MLR	GB & noMLR	noGB & noMLR
3 s	7.91	8.59	8.29	12.30
10 s	4.01	4.68	4.18	11.28
30 s	2.56	3.33	2.58	11.02

Table 6.1: Effect of GB and MLR in the calibration of a GMM-based LID system - $100 \cdot \mathbf{C_{avg}}$ for a GMM-based system with 2048 Gaussians, using 20 h for training and 1 h for development, for different activations of GB and MLR calibration, for the 3 s, 10 s, and 30 s tasks.

generative and discriminative calibration. Nonetheless, most of the benefit came from the GB, as we can see if we compare the results with the two blocks (GB & MLR) with those obtained only with the GB (GB & noMLR), and with those obtained only with MLR calibration (noGB & MLR). The numbers obtained only with the GB are closer to the numbers obtained with the cascade of both. This is especially true for the 30 s task. In the last column (noGB & noMLR), it can be seen that if neither GB nor MLR calibration was activated, the error increased dramatically. In the rest of experiments of the Thesis, we used the configuration with the two blocks in cascade.

6.2.2 GMM MAP

In this section, we investigate the performance of the LID approach described in Section 5.2 based on MAP adaptation of a GMM UBM. There is a need in this Thesis to report experiments with this technique because JFA and *i-Vector* models were created by MAP adaptation of the UBM. Hence, in order to make fair comparisons, we have to compare JFA and *i-Vectors* with GMM MAP training, and not with GMM trained from scratch.

In Table 6.2, we show the results of a GMM MAP adaptation of the means of each language for a 2048-component UBM. Remember that the weights and the covariance matrix remain the same as in the UBM, and that the UBM is a GMM trained identically to the models of the previous section but including data of all languages. We used the *Train* dataset of 20 h per language, the *Dev* dataset of 1 h per language, and the calibration included the generative and discriminative blocks in cascade.

We can see that the results obtained with the GMM trained from scratch, reported in Table 6.1 (column GB & MLR), were better than the results obtained with MAP adaptation. The difference was not big for the 10 s ($100 \cdot C_{avg} = 4.01$ with GMM vs.

Task	MAP
3 s	10.61
10 s	4.43
30 s	2.63

Table 6.2: Results of a GMM-based LID system trained with MAP - $100 \cdot C_{avg}$ for a GMM-based LID system trained with MAP adaptation of UBM means, using 20 h for training and 1 h for development, for the 3 s, 10 s, and 30 s tasks. The covariance and weights remain the same as in the UBM. The number of Gaussian components was 2048.

$100 \cdot C_{avg} = 4.43$ with MAP) and 30 s ($100 \cdot C_{avg} = 2.56$ with GMM vs. $100 \cdot C_{avg} = 2.63$ with MAP) tasks, but it was larger for the 3 s task ($100 \cdot C_{avg} = 7.91$ with GMM vs. $100 \cdot C_{avg} = 10.61$ with MAP). The reason is that 20 h of training data per language were enough to robustly train GMM model parameters, and the models adapted from a common UBM were less descriptive of the languages. They were forced to stay closer to the UBM than the models created from scratch, and this limitation did not help. After watching these results, one could think that the channel adaptation performed by JFA could rather be made from the GMM models trained from scratch. And it is probably true. However, this is dependent on the database size. As we looked for a standard recipe valid in most scenarios, we maintained MAP adaptation in JFA training at the expense of, probably, obtaining higher C_{avg}.

6.2.3 JFA

In this section, experiments with the JFA model described in Section 5.3 are presented. In all, we used the *Train* dataset of 20 h per language, the *Dev* dataset of 1 h per language, and the calibration included the generative and discriminative blocks in cascade.

The two most common evaluation methods of JFA are integration over channel distribution and linear scoring. They were described in Sections 5.3.1.2 and 5.3.1.5, respectively. In Figure 6.5, the performance of these two methods for different number of channel factors is compared. In this experiment, in the case of linear scoring, channel factors were obtained over the UBM, and hence they were the same for all languages. It can be observed that there is a plateau from 100 to 300 channel factors, with slightly better performance with 200 and 300 than with 100 in the 3 s and 10 s tasks, whereas

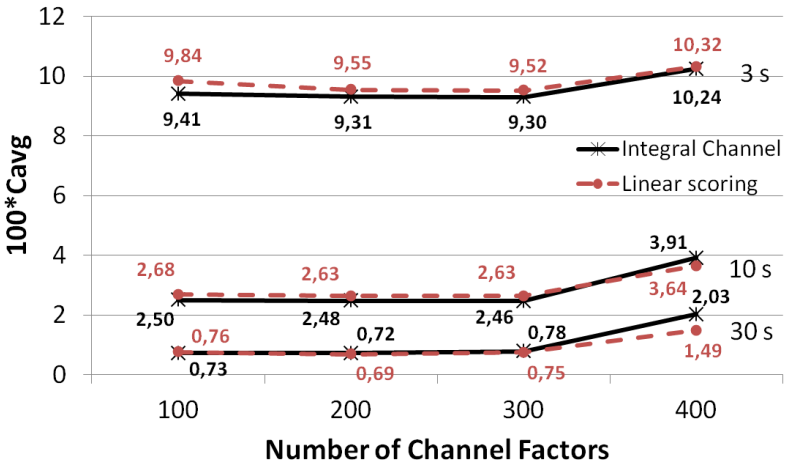


Figure 6.5: Performance of a JFA-based LID system with different number of channel factors - Evaluated with linear scoring (dotted red) and integrating over channel distribution (straight black), with 20 h for training and 1 h for development, for the 3 s, 10 s and 30 s tasks. Results in terms of $\mathbf{100 \cdot C_{avg}}$.

in the 30 s task the optimal result was obtained with 200 channel factors. We can also observe that the performance with both evaluation methods was similar. If we fix the number of channel factors to 200, marginally better results were obtained with integration over channel distribution in the 3 s ($100 \cdot C_{avg} = 9.31$ with integral vs. $100 \cdot C_{avg} = 9.55$ with linear scoring) and 10 s ($100 \cdot C_{avg} = 2.48$ with integral vs. $100 \cdot C_{avg} = 2.63$ with linear scoring) tasks, while in the 30 s task there was no significant difference between both of them ($100 \cdot C_{avg} = 0.72$ with integral vs. $100 \cdot C_{avg} = 0.69$ with linear scoring). For the rest of the experiments, we used 200 channel factors.

Next, we compare the rest of evaluation methods of JFA. Remember from Section 5.3.1 that according to the level of approximation, the most accurate method that we used is integration over channel distribution, followed by channel point estimate, and finally linear scoring. In methods that require to compute channel factors, they are theoretically computed with the specific model of each language, resulting in channel factors that are different for each of these languages, but we can reduce the computational cost by calculating channel factors over the UBM, obtaining a single vector of channel factors that is shared among languages. In practice, we did not see clear advantage of computing channel factors over the MAP models of each language and the results were similar or even better when we computed them over the UBM. The

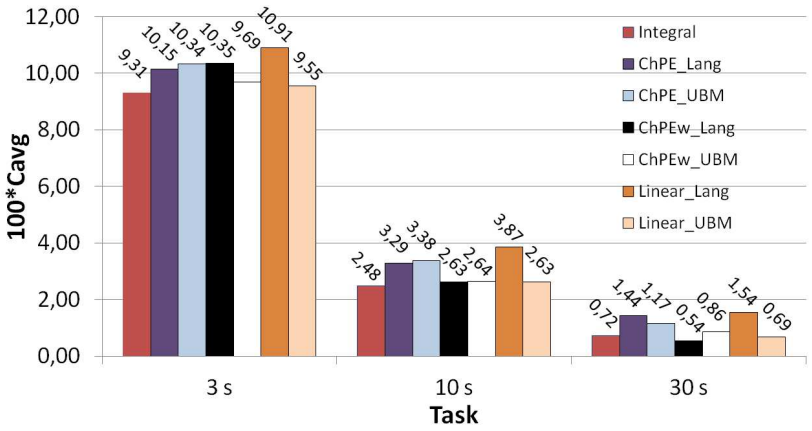


Figure 6.6: Performance of JFA obtained with different evaluation methods - Results with 200 channel factors, 20 h for training and 1 h for development, per language, in the 3 s, 10 s and 30 s tasks. Methods: integral over channel distribution (Integral), channel point estimate (ChPE), channel point estimate with UBM weights (ChPEw), and linear scoring (Linear). "_LANG" means that channel factors were obtained over language-specific models; "_UBM" means that channel factors were obtained over the UBM. Results in terms of $\mathbf{100 \cdot C_{avg}}$.

results are shown in Figure 6.6.

For the 30 s task, we can see that the best performance was obtained with channel point estimate with UBM weights and channel factors obtained with MAP models ($100 \cdot C_{avg} = 0.54$), but linear scoring with channel factors obtained with the UBM and integration over channel distribution also performed very well ($100 \cdot C_{avg} = 0.69$ and $100 \cdot C_{avg} = 0.72$, respectively). For the 3 s and 10 s tasks, integration over channel distribution performed the best ($100 \cdot C_{avg} = 9.31$ and $100 \cdot C_{avg} = 2.48$, for 3 s and 10 s tasks, respectively), and again, channel point estimate with UBM weights (with channel factors obtained with MAP models in the 10 s task and with the UBM in the 3 s task), and linear scoring with channel factors obtained with the UBM were the other two methods with lowest C_{avg}.

Actually, we expected integration over channel distribution to perform better than the rest of methods, especially in short durations, because in addition to being closer to the exact computation, channel factors are not computed but integrated out, and these are less reliable when the segment duration is short. As it can be seen in eq. (A.72), the covariance depends on the inverse of the zeroth order statistic. Thus, in short durations,

zeroth order statistic is small, the covariance is larger, and as a consequence, channel factors are less reliable.

It is remarkable that the approximation in channel point estimate with UBM weights gave better results than using soft alignments, although theoretically it is less exact. In fact, standard channel point estimate using soft alignments did not perform as expected. A surprising result is that linear scoring with channel factors computed over MAP models was one of the methods with worse results. However, if channel factors were computed over the UBM, linear scoring was one of the best scoring methods.

In general, we would recommend integration over channel distribution if you want to obtain very good performance at a reasonable speed, and the linear scoring with channel factors computed over the UBM, if you can afford a slight decrease in performance and you look for maximum speed.

In Dehak [2009], it was checked that channel factors calculated for a JFA model in a speaker recognition task also contained information about the speaker. In order to check if our channel factors also contained information about the language, we built a classifier where the input features were channel factors. The architecture was similar to the one shown in Figure 5.4 for *i-Vectors*, but using channel factors instead. First, we extracted channel factors computed over the UBM, and not over the language-specific MAP models, to have a single vector per utterance. Then, channel factors were used to model a Gaussian distribution per language, similarly to the system presented in Section 5.4.1.1. During evaluation, we extracted channel factors and evaluated them over the Gaussian models of each language. The scores generated by these models were calibrated before making decisions. Actually, this could be considered to be another JFA evaluation method where the input information is further compressed. But we want to separate it from the rest of methods to make clear that the only information used for classification was our channel information. The results are in Table 6.3. The number of extracted channel factors was 200. As we can see, the results were not as good as those obtained with the rest of JFA evaluation methods seen in Figure 6.6, but it is clear that the channel factors still contained information about the language. Thus, the next step was to build the total variability space, where JFA was used as feature extractor and the subspace included not only channel, but all sources of variability.

Task	200 channel factors
3 s	12.13
10 s	5.12
30 s	2.62

Table 6.3: Performance of a LID system based on JFA channel factor classification - $100 \cdot C_{avg}$ for LID with channel factor classification. A Gaussian distribution per language was used as classifier. 200-dimension channel factors were obtained over the UBM. Results using 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks.

6.2.4 *i-Vectors*

In this section, experiments using *i-Vectors* are shown. Description of an LID system based on *i-Vectors* can be seen in Section 5.4.1. First, results with the ML Gaussian classifier are presented. Then, several techniques are applied to compensate possible mismatches between train and test. Last, we study discriminative techniques for *i-Vector* classification, also analyzing some compensation techniques. In all the experiments, we used the *Train* dataset of 20 h per language, the *Dev* dataset of 1 h per language, and the calibration included the generative and discriminative blocks in cascade.

6.2.4.1 Gaussian Classifier

In Table 6.4, we analyze the results with the Gaussian classifier for different *i-Vector* dimensions. The best results for the 30 s condition were obtained with 600 dimensions ($100 \cdot C_{avg} = 0.36$). Adding more dimensions did not give further improvements and the computational load was highly increased. For the 10 s and 3 s tasks, small reductions in C_{avg} were still obtained with 700 and 800 dimensions (with 800 dimensions, $100 \cdot C_{avg} = 8.17$ in the 3 s task, and $100 \cdot C_{avg} = 1.75$ in the 10 s task), but a huge increase in computational load was required. Hence, for the rest of the experiments we used 600-dimension *i-Vectors*.

Theoretically, each language has its own mean and covariance. However, we used a shared WC covariance matrix for all languages. We investigated using a different covariance matrix for each language, but the results were worse, as shown in Table 6.5. However, we have shown in recent experiments that better results can be obtained

	i-Vector Dimension					
Task	100	200	400	600	700	800
3 s	8.44	8.40	8.36	8.29	**8.17**	**8.17**
10 s	2.04	1.92	1.88	1.83	1.77	**1.75**
30 s	0.55	0.46	0.44	**0.36**	0.39	0.39

Table 6.4: Results of an *i-Vector*-based LID system as a function of *i-Vector* dimension - $100 \cdot C_{avg}$ for an *i-Vector*-based LID system with Gaussian classifier and different *i-Vector* dimensions. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks.

Task	Heteroscedastic
3 s	13.75
10s	5.91
30s	1.99

Table 6.5: Results for an *i-Vector*-based LID system with heteroscedastic Gaussian classifier - $100 \cdot C_{avg}$ for a 600-dimension *i-Vector*-based LID system with Gaussian classifier and different covariance matrix for each language. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks.

with different covariance matrix for each class, if the amount of training data for each language is large enough [Lopez-Moreno et al., 2014].

Additionally, one could think that *i-Vectors* could have different modes within the same language and a GMM would be more appropriate than a single Gaussian. However, the fact that the heteroscedastic experiment did not outperform the one with shared covariance matrix, discouraged us to use GMMs, because if there were no data to train a different covariance matrix for each class, definitely there were no data to train more than one mode of a GMM either. Nonetheless, in Lopez-Moreno et al. [2014], it was also possible to train a GMM using a large enough dataset.

As it happens with channel factors in JFA, *i-Vectors* are less reliable when the audio segment is short. The reason is the same as for JFA. Larger covariance is obtained, since it depends on the inverse of the zeroth order statistic, which is small for short utterances. This relationship can be seen in eq. (A.104).

	NAP Dimension									
Task	20	30	40	50	60	75	100	125	150	200
3 s	8.31	8.31	8.30	8.29	8.27	8.26	8.25	8.25	8.29	8.27
10 s	1.83	1.83	1.83	1.83	1.83	1.83	1.83	1.83	1.81	1.81
30 s	0.37	0.37	0.37	0.37	0.37	0.37	0.37	0.38	0.38	0.38

Table 6.6: Results of an *i-Vector*-based LID system with NAP - $100 \cdot C_{avg}$ for a 600-dimension *i-Vector*-based LID system with Gaussian classifier and NAP with different dimensions. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks.

6.2.4.2 Gaussian Classifier with Nuisance Attribute Projection

In this section we study the performance of NAP technique applied with the Gaussian classifier of *i-Vectors*. In Table 6.6, we show results of the *i-Vector* system using 600 dimensions for different number of dimensions in the NAP projection. In this system, *i-Vectors* were previously centered around the origin. As we can see, the the system was practically insensitive to variations over a wide range of NAP dimensions. Comparing with the results in Table 6.4, there was no improvement over the *i-Vector* system without NAP.

6.2.4.3 Gaussian Classifier with Other Compensation Techniques

We experimented with other compensation techniques for the 600-dimension *i-Vector* system with Gaussian classifier. Results are shown in Table 6.7. Unfortunately, there were no gains with respect to the case without compensation.

6.2.4.4 Discriminative Training

In Figure 6.7, we show results for two discriminative classifiers built on top of 600-dimension *i-Vectors*. The first is an SVM and it is represented by the straight blue line. The second one is an LR classifier and it is represented by the dashed orange line. The parameter C was swept to find optimal performance. In general, it can be observed that LR performed better than SVM, and the optimal value of C was 0.001 in all cases, except for the LR system in the 3 s and 10 s tasks, for which the optimal value of C was 0.0001. Nonetheless, results with the generative Gaussian classifier outperformed discriminative ones.

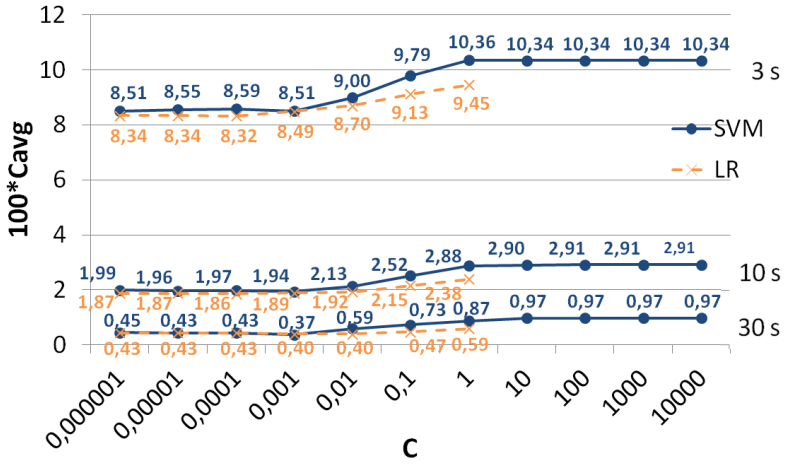


Figure 6.7: Comparison between SVM and LR classifiers for *i-Vector*-based LID - Results with 20 h for training and 1 h for development per language, for the 3 s, 10 s, and 30 s tasks, in terms of $\mathbf{100 \cdot C_{avg}}$. 600-dimension *i-Vectors*. SVM (straight blue) and LR (dashed orange).

Task	CW	CWN	CLDA	CNLDA
3 s	8.29	8.33	8.29	8.32
10 s	1.83	1.81	1.77	1.82
30 s	0.37	0.37	0.43	0.40

Table 6.7: Results for an *i-Vector*-based LID system with different compensation techniques - $100 \cdot C_{avg}$ for a 600-dimension *i-Vector*-based LID system with Gaussian classifier and different channel compensation techniques. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks. The abbreviations are

- Centering+Whitening (CW)
- Centering+Whitening+Length Normalization (CWN)
- Centering+Linear Discriminant Analysis (CLDA)
- Centering+Length Normalization+Linear Discriminant Analysis (CNLDA)

6.2.4.5 Compensation Methods for Discriminative Training

Next, we incorporate channel compensation techniques to discriminative methods. In Figure 6.8, we see how LDA, NAP with 60 dimensions, length normalization, and combinations of these three, affected SVM. The regularization parameter was $C = 0.001$. In this case, LDA was the most effective transform in the 3 s and 10 s tasks, but

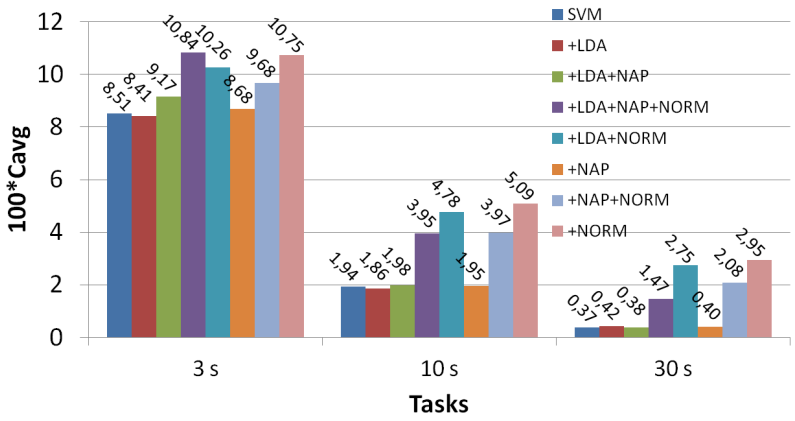


Figure 6.8: Results for an LID system using 600-dimension *i-Vectors* with SVM classifier and several channel compensation techniques - Specifically, LDA, NAP with 60 dimensions, length normalization (NORM), and combination of them. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks, in terms of $100 \cdot C_{avg}$.

not enough to beat the generative results, while in the 30 s task, the best results with SVM were obtained without any compensation method.

The same exercise was done with an LR classifier with $C = 0.0001$. NAP was also applied with 60 dimensions. In Figure 6.9, we can see the results. LDA and NAP helped, but not significantly. The results with the generative Gaussian classifier were still better.

6.3 Comparison and Fusion of Acoustic Systems

In this section, we compare the results obtained with the different acoustic approaches and investigate if they are complementary. In Figure 6.10, we present some of the best individual results obtained with GMM, MAP, JFA and *i-Vector* approaches, and different fusions at score level, with the system architecture proposed in Figure 4.3. Specifically, we make all possible combinations of the MAP system with 2048 Gaussians, the JFA with integral over channel distribution scoring and 200 channel factors, and the 600-dimension *i-Vector* system with Gaussian classifier. We also included GMM with 2048 for comparison, although as we have stated before, it is not a fair comparison in the sense that, unlike the other three approaches, GMMs are built from scratch and not

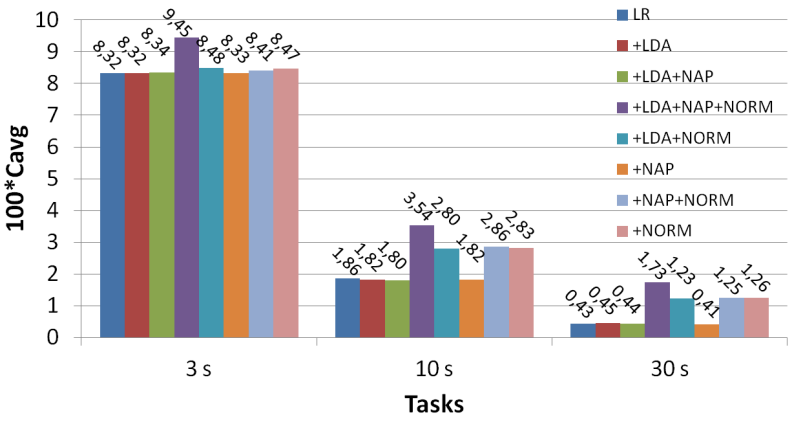


Figure 6.9: Results for an LID system using 600-dimension *i-Vectors* with an LR classifier and several channel compensation techniques - Specifically, LDA, NAP with 60 dimensions, length normalization (NORM), and combinations of them. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks, in terms of $100 \cdot C_{avg}$.

from a MAP adaptation of a UBM. As we can see, the performance varied depending on the duration of the test segments.

In the case of 30 s, the best performing system was the one based on *i-Vectors* ($100 \cdot C_{avg} = 0.36$), and the fusion did not help, probably because we had already reached nearly optimal performance using SDC features with the *i-Vector* approach alone. Comparing *i-Vectors* with JFA evaluated with integration over channel distribution and with MAP, the first obtained a 50% relative improvement with respect to the second, and a 86.31% with respect to the third. The relative improvement of JFA with respect to MAP was of 72.62%. Note that in this case it is harder to extract significant conclusions, since the number of errors is very small (below 40 in the best-performing system).

In the case of 10 s test segments, the best individual result was obtained with the *i-Vector* system ($100 \cdot C_{avg} = 1.83$). This meant a 26.21% relative improvement over the best JFA evaluation method, namely integral over channel distribution scoring, and 58.69% relative improvement over MAP. The relative improvement of JFA with respect to MAP was of 44.01%. Interestingly, the fusion of MAP and *i-Vector* obtained lower C_{avg} than the fusion of JFA and *i-Vector*. By fusing the three approaches the best result was obtained, $100 \cdot C_{avg} = 1.62$. This meant a 11.48% relative improvement

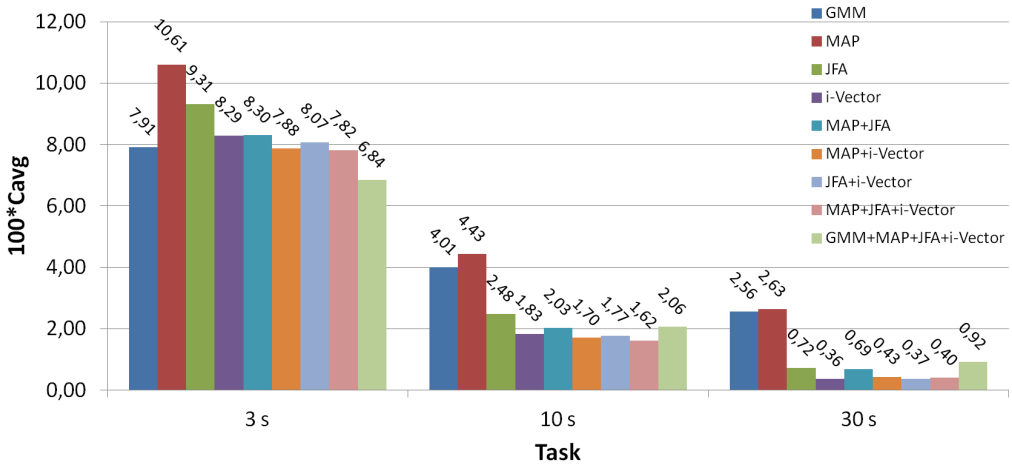


Figure 6.10: Results for fusion of acoustic systems - GMM with 2048 components, GMM MAP with 2048 components, JFA evaluated with integral scoring and 200 channel factors, and 600-dimension *i-Vectors* evaluated with a Gaussian classifier. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks. Results in terms of $100 \cdot C_{avg}$.

over the *i-Vector* approach alone.

In the case of 3 s test segments, and considering only the three techniques that used MAP adaptation, the best individual result was obtained with the *i-Vector* approach ($100 \cdot C_{avg} = 8.29$). However, a better result was obtained with GMM ($100 \cdot C_{avg} = 7.91$). See that the subspace was still doing its job, because the results with JFA and *i-Vectors* were better than the results with GMM MAP. However, note that the difference between GMM and MAP was relatively higher (25.44%) in this case than in the 10 s and 30 s tasks (9.48% and 2.66%, respectively), and the improvement that the subspace allowed us to achieve was not enough to beat the results with GMMs. The relative improvement of the *i-Vector* approach with respect to JFA was of 10.96%, and with respect to MAP was of 21.86%. The relative improvement of JFA with respect to MAP was of 12.25%. As it already happened in the 10 s task, the fusion of MAP and *i-Vector* obtained better result than the fusion of JFA and *i-Vector*. By fusing the three MAP-based techniques, we obtained $100 \cdot C_{avg} = 7.82$, a relative improvement of 5.67% over the *i-Vector* system alone. By adding the GMM system to this fusion, we obtained $100 \cdot C_{avg} = 6.84$, showing the complementarity of this technique with the others when the utterances were short. This meant a 12.53% relative improvement

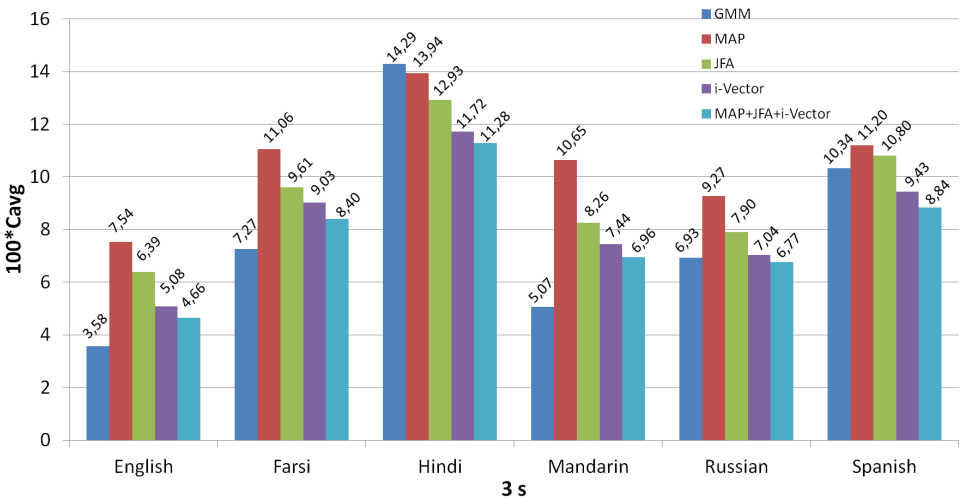


Figure 6.11: Results per language for acoustic systems in the 3 s task - GMM with 2048 Gaussian components, MAP with 2048 Gaussian components, JFA evaluated with integral scoring and 200 channel factors, 600-dimension *i-Vector* with Gaussian classifier, and fusion of previous MAP, JFA, and *i-Vector* systems. Results with 20 h for training and 1 h for development, per language, in terms of $100 \cdot C_{avg}$.

over the previous fusion without MAP, and a 17.49% relative improvement over the *i-Vector* system alone.

These results prove the complementarity of MAP, JFA and *i-Vectors* in the 3 s and 10 s tasks. Probably, they would also be complementary in the 30 s task in more challenging databases. Also, it is clear the advantage of the subspace, because the results obtained with JFA and *i-Vectors* were always better than those obtained with MAP. At the same time, the *i-Vector* approach outperformed JFA. A reason may be that *i-Vectors* do not discard any language information, as we saw in Section 6.2.3. Finally, GMM alone outperformed MAP, and as we proposed in Section 6.2.2, probably better results would have been obtained if we had used the GMM models instead of the MAP models to build JFA and *i-Vector* systems.

In Figures 6.11, 6.12, and 6.13, we split C_{avg} into the individual results obtained for each language with the previous systems, in the 3 s, 10 s, and 30 s tasks, respectively. In this way, we can see if some languages benefited more from using one approach or another.

In the 3 s task, the languages with lowest error were English, Mandarin and Russian. Opposite, Hindi obtained the worst results. It is interesting that the fusion was the best

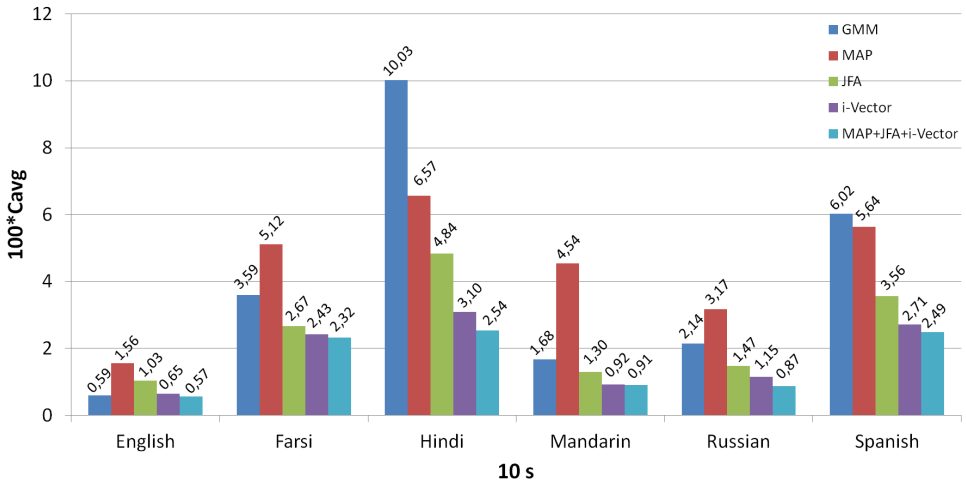


Figure 6.12: Results per language for acoustic systems in the 10 s task - GMM with 2048 Gaussian components, MAP with 2048 Gaussian components, JFA evaluated with integral scoring and 200 channel factors, 600-dimension *i-Vector* with Gaussian classifier, and fusion of previous MAP, JFA, and *i-Vector* systems. Results with 20 h for training and 1 h for development, per language, in terms of $100 \cdot C_{avg}$.

performing system for all languages, except for Mandarin, for which GMMs performed the best. Unlike the behavior for the rest of languages, MAP performed better than GMM for Hindi. The rest of results followed the general trend commented above.

In the 10 s task, the best results were obtained again for English, whereas Hindi was again the language with worst results. However, in this case and for the *i-Vector* system, Hindi and Spanish obtained very similar results. In addition, both languages had in common that GMM performed worse than MAP. The fusion was the system performing best for all languages, and the rest of results followed the general trend.

In the 30 s task, excellent results were obtained for Mandarin. C_{avg} of English and Russian were close to Mandarin. The results obtained for Farsi, Hindi, and Spanish were similar and with higher C_{avg} than for the other three. In this case, the *i-Vector* system performed the best for all languages except Hindi, and identical results than with *i-Vectors* were obtained with the fusion for Mandarin and Russian. It is remarkable the very bad results obtained with GMM for Hindi.

Finally, we show the confusion matrices for the *i-Vector* system in Tables 6.8, 6.9, and 6.10, for the 3 s, 10 s, and 30 s tasks, respectively. In the rows we have the true spoken language, and in columns, the decisions made by our system. Results are

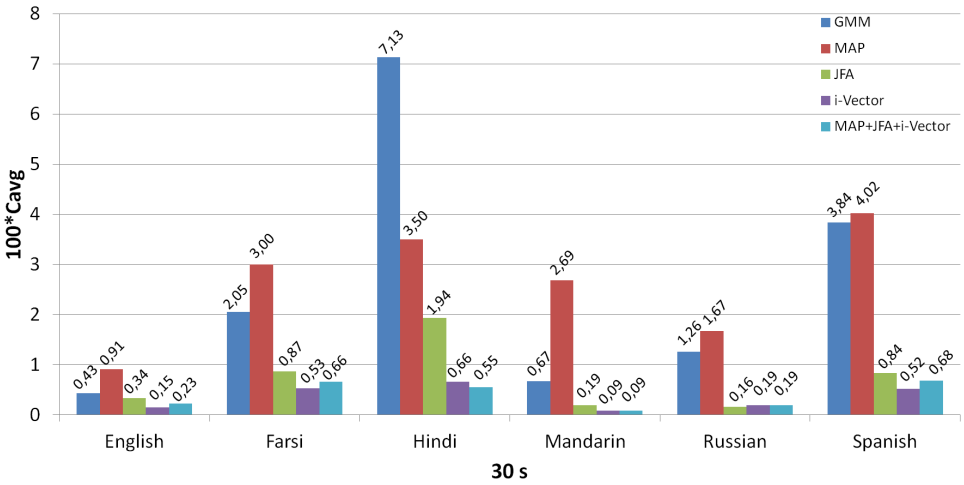


Figure 6.13: Results per language for acoustic systems in the 30 s task - GMM with 2048 Gaussian components, MAP with 2048 Gaussian components, JFA evaluated with integral scoring and 200 channel factors, 600-dimension *i-Vector* with Gaussian classifier, and fusion of previous MAP, JFA, and *i-Vector* systems. Results with 20 h for training and 1 h for development, per language, in terms of $100 \cdot C_{avg}$.

shown in percentage of files. Thus, we can know the number of files incorrectly classified using these percentages and Table 4.3, where we indicated the number of target files per language and task. We want to remind that this is a detection task, where each utterance can be said to be spoken in zero, one or more than one language, and thus, the sum of each column does not have to sum 1.

For the 3 s task, it is interesting to see that the lowest numbers of correctly classified files belonged to Hindi (54.07%) and Spanish (58.23%). The other languages varied from 70% (Farsi) to 83% (English) of files correctly classified. The major confusion was for Spanish files classified as Hindi (10.50%). Also, there was big confusion in Hindi files classified as Farsi (8.37%), Russian (7.66%), and Spanish (7.18%). Moreover, 7.90% of Farsi files were classified as Mandarin.

For the 10 s task, the percentage of correctly classified files improved very much with respect to the 3 s task. Hindi, Farsi and Spanish were between 86% and 90% of correct classifications, whereas the other three languages were over 96%. The biggest confusion was between Hindi and Spanish. Also, Farsi with Mandarin and Russian, Hindi with Russian, and Spanish with Farsi obtained a percentage over 1%.

Finally, for the 30 s task, all languages obtained a correct classification rate over

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English	83.85	2.43	2.65	2.43	1.33	1.99
Farsi	3.95	70.37	2.96	7.90	3.46	1.98
Hindi	3.35	8.37	54.07	5.26	7.66	7.18
Mandarin	0.93	5.09	3.94	76.62	3.24	1.62
Russian	2.04	3.63	4.31	1.81	78.46	2.04
Spanish	4.06	5.01	10.50	3.82	5.01	58.23

Table 6.8: Confusion matrix for an *i-Vector*-based LID system in the 3 s task - Confusion matrix (%) with the 600-dimension *i-Vector*-based LID system and Gaussian classifier in the 3 s task. Results with 20 h for training and 1 h for development, per language. Rows are true spoken language, and columns are decisions made by our system.

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English	97.80	0.44	0.29	0.15	0.00	0.15
Farsi	0.25	89.41	0.74	1.72	1.48	0.74
Hindi	0.48	0.97	86.47	0.32	1.13	2.25
Mandarin	0.00	0.20	0.99	97.62	0.40	0.00
Russian	0.00	0.45	0.68	0.23	96.83	0.23
Spanish	0.95	1.91	2.39	0.72	0.72	87.11

Table 6.9: Confusion matrix for an *i-Vector*-based LID system in the 10 s task - Confusion matrix (%) with the 600-dimension *i-Vector*-based LID system and Gaussian classifier in the 10 s task. Results with 20 h for training and 1 h for development, per language. Rows are true spoken language, and columns are decisions made by our system.

96%. In the case of English, Mandarin, and Russian, it was over 99%. The biggest errors were made for Spanish files, with 0.48% classified as Farsi, and 0.48% classified as Hindi, and for Hindi files, with 0.47% classified as Russian. Interestingly, in this case no Hindi file was confused with Spanish.

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English	99.10	0.00	0.00	0.00	0.00	0.00
Farsi	0.00	97.28	0.25	0.25	0.25	0.00
Hindi	0.00	0.00	96.74	0.00	0.47	0.00
Mandarin	0.00	0.00	0.00	99.72	0.00	0.00
Russian	0.00	0.00	0.00	0.00	99.55	0.00
Spanish	0.00	0.48	0.48	0.00	0.00	96.90

Table 6.10: Confusion matrix (%) with the 600-dimension *i-Vector*-based LID system and Gaussian classifier in the 30 s task. Results with 20 h for training and 1 h for development, per language. Rows are true spoken language, and columns are decisions made by our system.

7

Prosodic LID Results

Contents

Given the success of the *i-Vector* classifier for acoustic LID, we kept the same architecture for the prosodic system. The only difference were the input features extracted from speech. Whereas in Chapter 6 we focused on spectral information, now we focus on prosody and evolution of formants along time. Prosodic LID can be considered to be a traditional research field in the LID community, and formant information has also been used in many works. Our major contribution in the experiments presented in this Chapter is the use of prosodic and formant information together in an i-Vector-based LID system. This allowed us to obtain much better results than previous approaches [Ng et al., 2010; Martínez et al., 2012a, 2013b]. However, the performance of this approach was below the acoustic one. Nevertheless, both types of information were complementary, and a the fusion of both approaches increased system accuracy. This behavior is in agreement with most of the works of the literature.

7.1 Introduction to Prosodic LID

Prosody is encoded in syllable length, loudness, and rhythm. These attributes make humans perceive rhythm, stress, and intonation in speech. Additionally, formants are very useful to disambiguate vowels, and vowels can be a good hint to identify a language. Furthermore, vowel evolution in time is dependent on prosody itself and also on phonotactics, another important cue to distinguish languages. As vowels are surrounded by consonants, formant transitions are affected by phonotactics.

Traditionally pitch, energy, and duration of specific speech segments have been used to capture prosody for LID systems [Foil, 1986; Lin and Wang, 2005; Mary and Yegnanarayana, 2008]. One of the first attempts to automatically identify language from prosody is found in Foil [1986]. In that work, several techniques were used to model pitch and energy contours, and linear prediction coefficients were used to also model formants. The author chose formant values for two reasons: *i)* it is known that human ear and brain make use of formant information to distinguish sounds, and *ii)* additive wideband noise has less effect in the peaks of the spectrum. We also know that the repertoire of voiced sounds is different for every language, and thus vowels are different in different languages, and consequently formants differ from one language to another. In addition, the frequency of formants is heavily dependent on stress, and less (but also) significantly on duration [Gay, 1978]. This second phenomenon happens because of the

influence of neighboring segments and it is known as undershoot [Lindblom, 1963]. Since stress and duration behave different in different languages, formants should be a good feature for LID.

Some authors obtain prosodic features with the help of an ASR system [Mary and Yegnanarayana, 2008], but this makes the process computationally expensive. In most works ASR is avoided. In Lin and Wang [2005], pitch contours were approximated using Legendre polynomials over long temporal intervals with very promising results. This approach was adopted for speaker identification [Dehak et al., 2007a; Ferrer et al., 2010; Kockmann et al., 2010a], where pitch and also energy contours were approximated using linear combination of Legendre polynomials over syllable or syllable-like units. The regression coefficients together with durations of corresponding segments were the features describing the three characteristics of prosody. In Dehak et al. [2007b], formants were used additionally to pitch, energy, and duration, and they were modeled in units representing syllables, for another speaker identification task. In that work, the use of formant information made possible to reduce the error function with regard to the prosodic system without formants, but there were no further improvements when fusing a cepstral system with the prosodic system including formants, compared to the fusion of the cepstral system with the prosodic system without formants.

Our work is inspired by the prosodic-based speaker identification literature. One of the most popular prosodic approaches for speaker identification was JFA [Dehak et al., 2007a,b; Kockmann et al., 2010a]. The standard *i-Vector* approach initially proposed to model MFCC features, was tested on polynomial coefficient prosodic features in Kockmann et al. [2011], and a remarkable performance was shown for a speaker identification task, compared to JFA. These approaches are applicable only to features that are always defined and are relatively low-dimensional, like the polynomial coefficient features described above. For more complex sets of features, another subspace modeling technique called the subspace multinomial model (SMM) [Kockmann et al., 2010b] was introduced, which models the vector of weights from a background GMM that takes into account probabilities of undefined values.

For LID, the *i-Vector* approach combined with prosodic feature was used for the first time in our work in Martínez et al. [2012a]. In Martínez et al. [2013a], we expanded the previous work with the addition of formant information.

Prosody Aspect	Intonation	Rhythm	Stress	Vowel Evolution
Feature	Pitch	Duration	Energy Duration	Formants

Table 7.1: Relationship between features and prosodic aspects.

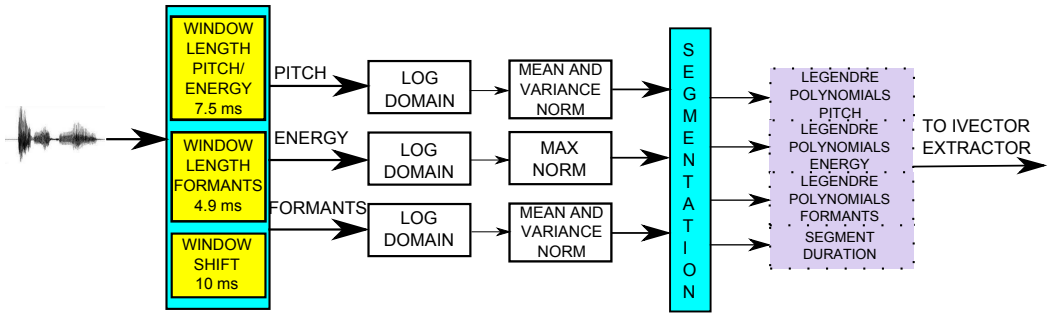


Figure 7.1: Extraction process of prosody and formant features - For each audio file, we compute pitch, energy, and formants, we normalize them, we obtain meaningful segments, and we compute the contour of each segment with Legendre polynomials. Finally, the segment duration is added to the polynomial coefficients to obtain the feature vector used as input to the LID classifier.

7.2 Prosodic Features

In this section we explain the feature extraction process of prosodic features and formant information. The process is depicted in Figure 7.1. First, we calculated the pitch, the energy, and the formants of each audio file. Second, we normalized these measures. The third step in the process was the segmentation. The goal was to obtain meaningful regions where we could compute reliable contours for pitch, energy, and formants. Finally, we computed contours by means of Legendre polynomials within each of the previous segments. The coefficients of those polynomials were our features, which together with the segment duration, were the input to our LID classification system. Then, we can consider that our features contained information about the three components of prosody: intonation in pitch, rhythm in duration, and stress in energy and in duration. Formant contours primarily carry information about vowel evolution. In Table 7.1, we summarize how our features collect the prosodic information of speech. In the following paragraphs, we explain this process in depth.

7.2.1 Pitch, Energy and Formant Extraction

Our features carried information about the evolution of pitch, energy and formant central frequencies along time. The Snack Sound Toolkit [Sjölander, 1997] was used to extract these features. They were obtained every 10 ms with 7.5 ms long windows. First, pitch, energy and formant values were converted to log domain, to simulate human perception. Next, energy was normalized by subtracting its maximum value in the log scale. This made it more robust to language-independent phenomena such as channel variations. The log pitch and log formant values were normalized by subtracting mean and dividing by standard deviation estimated over the corresponding file. Thus we avoided the dependence on the absolute pitch value of the speaker. Formant processing was similar to Dehak et al. [2007b], but unlike this work, they were converted to log domain and we studied the influence from F1 to F4.

7.2.2 Segment Definition

Once we extracted pitch, energy and formant central frequencies for whole speech recordings, the next step was to model their contours over time. First, we had to define over which time intervals we would create those contours and segment the signal accordingly. In Kockmann et al. [2010a], different segment definitions were tested and segmentation based on syllables detected using an ASR system was found to perform best. Since the language is unknown in the case of LID, we wanted to avoid the use of ASR. In the present work, we compared fixed length segments of 200 ms shifted every 10 ms and 50 ms, with segments whose boundaries were delimited by energy valleys [Dehak et al., 2007a; Kockmann et al., 2010a; Martínez et al., 2012a]. In Figure 7.2, we show how the segmentation process was made once we had the pitch and energy calculated for an audio recording, for the case of fixed segmentation of 200 ms shifted every 50 ms, and in Figure 7.3, for energy valley segmentation.

7.2.3 Contour Modeling

For each segment, we dropped all unvoiced frames for which no pitch was detected. Then pitch, energy and formant central frequencies were approximated by linear com-

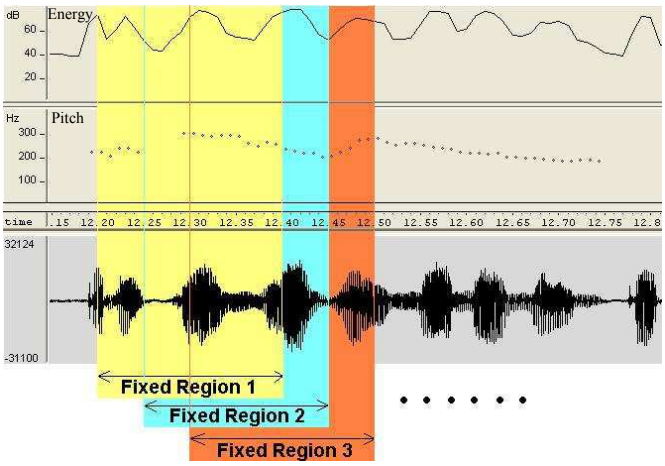


Figure 7.2: Fixed Segmentation - This file is divided in fixed segments of 200 ms shifted every 50 ms. For the computation of the contours, only voiced frames are used.

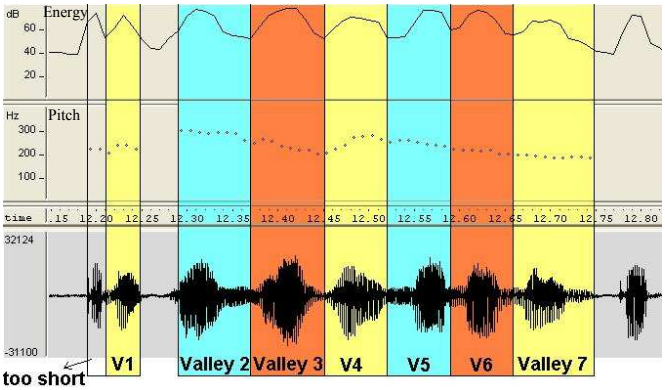


Figure 7.3: Energy Valley Segmentation - On every minimum of the energy signal, a new segment is created. For the computation of the contours, only voiced frames are used.

bination of Legendre polynomials as

$$f(t) = \sum_{i=0}^{M} a_i P_i(t) \tag{7.1}$$

where $f(t)$ is the contour being modeled and $P_i(t)$ is the ith Legendre polynomial. Each coefficient a_i represents a characteristic of the contour shape: a_0 corresponds to the mean, a_1 to the slope, a_2 to the curvature, and higher order represents more precise detail of the contour. Note that a Legendre polynomial of order M gives $M+1$ coefficients for pitch, $M+1$ for energy, and $M+1$ for every additional formant. The contours were calculated by least squares [Bishop, 2006]. In Figure 7.4, we show the 6 first basis of Legendre polynomials. In Figure 7.5, a real F1 curve extracted from a 200 ms segment with 20 voiced frames is compared to its Legendre approximation.

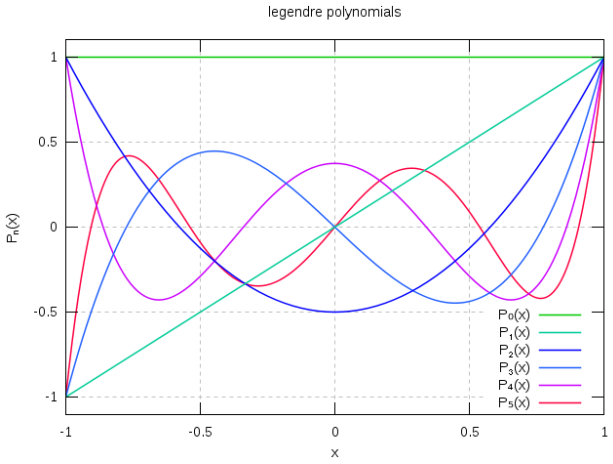


Figure 7.4: Basis of Legendre polynomials - 6 first basis of Legendre polynomials used to model pitch, energy, and formant contours. Each basis models a different aspect of the contour. The coefficients associated to the basis are the features of our system.

Finally, the number of voiced frames used to calculate the polynomials was included to also include segment duration. Thus, for example, if we used order 5 polynomials, every contour was modeled with 6 coefficients, and a 13 dimension feature vector was obtained for the case with no formants, up to 37 dimensions if the first 4 formants (F1-F4) were included. These were the features used as input to our classifier.

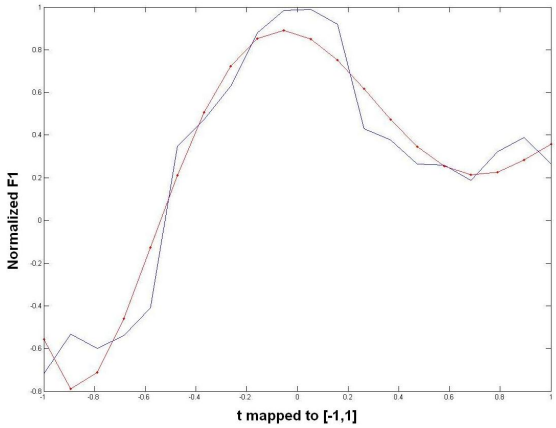


Figure 7.5: Legendre approximation to actual F1 curve - Comparison of actual F1 curve (straight blue) with Legendre approximation of order 5 (dotted red) in a fixed-length contour of 200 ms.

7.3 Results

The system architecture of our prosodic and formant system based on *i-Vectors* can be seen in Figure 5.4. It is the same as for the acoustic system, trained in the same way and with the same data, but substituting SDC by prosodic and formant features. Moreover, we used the configuration that performed best for the acoustic system, that is, a 2048 Gaussian components UBM and 600-dimension *i-Vectors*. In a prosodic system, performance can be influenced by several parameters in the feature extraction process, like the type of features used, the type of segments where contours are computed, or the polynomial order used to model the contours. In the next experiments we analyze these factors.

In Figure 7.6, we study how each individual feature affects the performance. Initially, we used fixed segments of 200 ms to emulate syllable-like units, and 10 ms shift, or in other words, 190 ms overlap. The initial polynomial order was 5, and thus, each contour was modeled with 6 coefficients. We can see that the best results were obtained with the combination of pitch, energy, duration, F1 and F2 ($C_{avg} = 3.01$, $C_{avg} = 6.68$, $C_{avg} = 13.90$ for the 30 s, 10 s, and 3 s tasks, respectively). Adding more formants did not help. It is interesting to see that all the features were useful alone, but obviously, far from the performance obtained with the combination we have just mentioned.

We want to make a brief remark on formant behavior. See that adding only F1

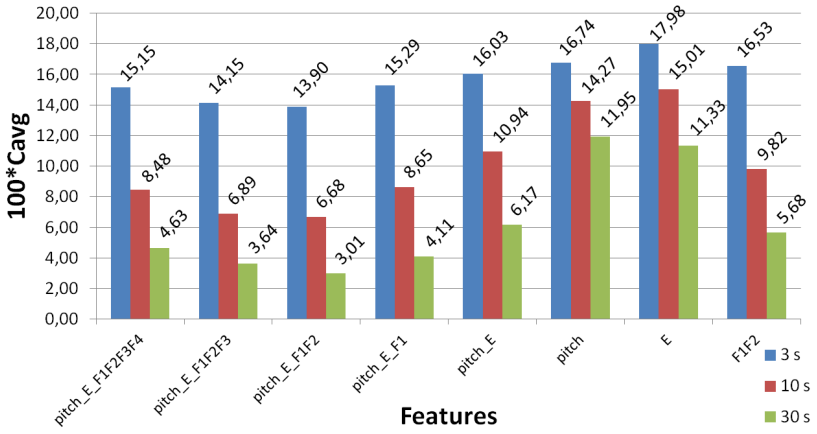


Figure 7.6: Results for an LID system using 600-dimension *i-Vectors* and different combinations of prosodic and formant features - Results using 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks, in terms of $100 \cdot C_{avg}$. "E" stands for energy, and all feature combinations include segment duration.

to the prosodic features alone (pitch, energy and duration) made a clear difference, and the addition of F2 allowed a further reduction of C_{avg}. Then, adding F3 and F4 was not beneficial and the error was increased again. Theoretically, F1, F2, and F3 are primarily acoustic correlates of vowel height, place of articulation and rounding, and are known as vowel formants, because they make possible vowel discrimination, whereas F4 and higher formants are often said to be acoustic correlates of the speakers' vocal tract characteristics [Stevens, 1998]. Moreover, F4 and higher formants appear to have little useful perceptual effect when their central frequency is changed [O'Shaughnessy, 2008]. In addition, estimations of F3, and also of higher formants, are often difficult owing to the low energy in their frequency ranges [O'Shaughnessy, 2008], and hence, it is logical that they do not contribute so reliably to the classification. Given these facts, it can be said that F1 and F2 were expected to contribute the most to the discrimination of languages and that is what we observed in our experiments.

After the conclusions drawn from the previous results, we used pitch, energy, duration, F1 and F2 for the rest of our experiments. The next one consisted in analyzing different segments where contours were computed. We compared the fixed segments of 200 ms with 10 s of previous experiment, with fixed segments of 200 ms and 50 ms shift, and with segments delimited by energy valleys. The results are in Table 7.2. As we can observe, the best results were obtained with fixed segments of 200 ms and 10 ms

Task	Fixed 200 ms - Shift 10 ms	Fixed 200 ms - Shift 50 ms	Energy Valleys
3 s	**13.90**	14.94	16.79
10 s	**6.68**	8.21	12.14
30 s	**3.01**	3.86	6.85

Table 7.2: Results for a prosodic and formant *i-Vector*-based LID system for different segment definitions - $100 \cdot C_{avg}$ for an LID system using 600-dimension *i-Vectors*, pitch, energy, duration, F1, and F2 features, and different segments where contours are computed. Results using 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks.

Task	Order 4	Order 5	Order 6
3 s	15.01	**13.90**	15.08
10 s	8.35	**6.68**	8.71
30 s	3.62	**3.01**	4.21

Table 7.3: Results for a prosodic and formant *i-Vector*-based LID system for different Legendre polynomial orders - $100 \cdot C_{avg}$ for an LID system using 600-dimension *i-Vectors*, pitch, energy, duration, F1, and F2 features, and different polynomial orders. Results using 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks.

shift. Although it is computationally more expensive than fixed segments with 50 ms shift, because many more coefficients are computed, we think that it is worth the extra cost if you can afford it due to the difference in the error. On the other hand, energy valley segments presents the benefit of even higher speed, but at the cost of obtaining higher C_{avg}.

Finally, we investigated the impact of the Legendre polynomials order. In Table 7.3, we present results obtained with orders 4, 5, and 6. The systems used in previous experiments with order 5 performed the best. The results obtained with orders 4 and 6 were clearly worse than the results obtained with order 5.

As we already did in previous chapter, now we split C_{avg} into the individual results obtained for each language, with the system including pitch, energy, duration, F1, and F2 features, using fixed segments of 200 ms and 10 ms shift, and 5 order polynomials. These are reflected in Figure 7.7 for the 3 s, 10 s, and 30 s tasks.

First, we can see that Farsi, Hindi, and Spanish were the languages obtaining highest

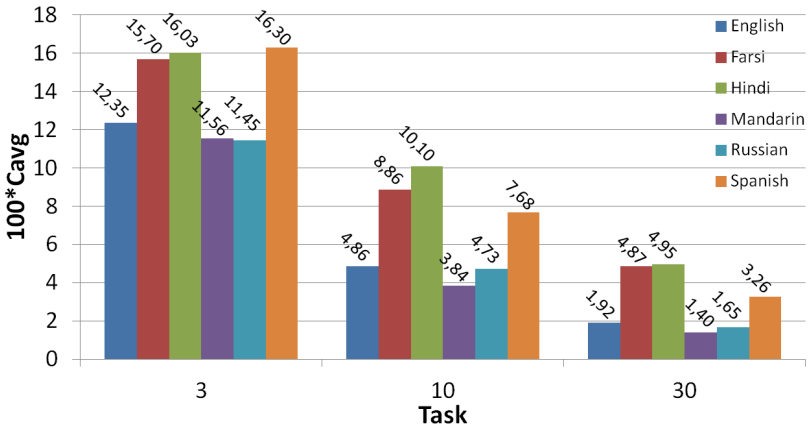


Figure 7.7: Results split by languages for a prosodic and formant *i-Vector*-based LID system - 600-dimension *i-Vector*-based system with Gaussian classifier, built on pitch, energy, duration, F1, and F2 features, with 200 ms fixed segments, 10 ms shift, and order 5 polynomials. Results using 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks, in terms of $100 \cdot C_{avg}$.

C_{avg}, as it already happened with the acoustic system. The most remarkable and interesting result is that Mandarin was the language with best performance in the 10 s and 30 s tasks. It makes much sense, since Mandarin is a tonal language in which prosody plays a fundamental role, much more prominent than in the rest of languages.

Finally, in Tables 7.4, 7.5, and 7.6, we can see the confusion matrices for the 3 s, 10 s, and 30 s tasks, respectively.

For the 3 s task, we can appreciate very low accuracy for Farsi (38.27%), Hindi (34.21%), and Spanish (44.15%). Many Hindi files were confused with Farsi (11%) and Spanish (13.64%). The rest of confusions were below 10%.

For the 10 s task, accuracies increased considerably. The lowest were again for Hindi (58.62%), Farsi (60.34%), and Spanish (66.59%). The percentage of Farsi files classified as Hindi was 5.67%, 6.76% of Hindi files were classified as Spanish, and 8.11% of Spanish files were classified as Hindi.

For the 30 s case, the lowest correct classification rate was in Farsi (73.76%). For English, Mandarin, and Russian it was over 90%. Farsi were confused with Hindi in 2.97% of the files, and with Russian in 2.48% of the files. 3.10% of Spanish files were confused with Hindi.

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English	51.33	7.96	3.98	5.53	4.65	8.19
Farsi	7.65	38.27	5.43	6.67	7.65	6.67
Hindi	3.11	11.00	34.21	5.02	5.74	13.64
Mandarin	4.40	5.56	5.79	57.18	3.01	5.56
Russian	4.31	3.40	5.67	4.76	58.05	7.94
Spanish	5.97	4.53	9.55	4.53	5.73	44.15

Table 7.4: Confusion matrix for the prosodic and formant *i-Vector*-based LID system for the 3 s task - Features are pitch, energy, duration, F1, and F2 features, using 200 ms fixed segments, 10 ms shift, and order 5 polynomial. System using 600-dimension *i-Vectors* and Gaussian classifier. We used 20 h for training and 1 h for development, per language. Rows are true spoken language, and columns are decisions made by our system (in %).

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English	78.04	2.34	1.46	1.61	2.20	0.59
Farsi	1.72	60.34	5.67	1.48	4.19	2.96
Hindi	1.29	7.25	58.62	1.29	2.42	6.76
Mandarin	1.39	0.60	2.38	82.74	1.19	0.99
Russian	0.91	0.68	1.59	0.68	86.17	1.36
Spanish	1.91	2.63	8.11	0.72	4.53	66.59

Table 7.5: Confusion matrix for the prosodic and formant *i-Vector*-based LID system for the 10 s task - Features are pitch, energy, duration, F1, and F2, using 200 ms fixed segments, 10 ms shift, and order 5 polynomial. System using 600-dimension *i-Vectors* and Gaussian classifier. We used 20 h for training and 1 h for development, per language. Rows are true spoken language, and columns are decisions made by our system (in %).

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English	90.05	0.45	0.90	0.45	0.90	0.00
Farsi	0.74	73.76	2.97	1.49	2.48	0.74
Hindi	0.00	1.86	78.14	0.00	0.93	0.93
Mandarin	0.56	0.00	0.83	94.72	0.56	0.00
Russian	0.00	0.45	0.00	0.23	95.92	0.00
Spanish	0.24	0.24	3.10	0.95	0.95	82.10

Table 7.6: Confusion matrix for the prosodic and formant *i-Vector*-based LID system for the 30 s task - Features are pitch, energy, duration, F1, and F2, using 200 ms fixed segments, 10 ms shift, and order 5 polynomial. System using 600-dimension *i-Vectors* and Gaussian classifier. We used 20 h for training and 1 h for development, per language. Rows are true spoken language, and columns are decisions made by our system (in %).

8

Fusion of Acoustic and Prosodic LID Systems

Contents

8.1 Analysis of Fusion of Acoustic and Prosodic *i-Vector*-based LID

In this chapter, we investigate if the acoustic and prosodic systems presented in previous chapters are complementary. Theoretically, acoustic, prosodic and formant information capture different aspects of speech, so we expected the fusion of these approaches to be a more robust system than any of them individually. Fusion was performed at score level, as described in Section 4.2.3. Also, we present a deep analysis of the fusion system, including an analysis of how the channel type affects classification.

Several fusions were tested, including different combinations of acoustic and prosodic systems. Among the acoustic ones, we selected GMM MAP-based system with 2048 Gaussian components, JFA with integral over channel distribution scoring and 200-dimension channel subspace, and 600-dimension *i-Vectors* with Gaussian classifier, to be included in the fusions. We also analyzed if the GMM system trained from scratch offers additional gains. The system selected for the prosodic and formant approach was based on 600-dimension *i-Vectors*, with pitch, energy, duration, F1 and F2, fixed segments of 200 ms and 10 ms shift, and order 5 polynomials. The results of the fusion can be seen in Figure 8.1. The main general idea that we extract of them is that, although the performance of prosodic systems is quite lower than the performance of acoustic systems, they are complementary. If we compare the results with those shown on Figure 6.10 for acoustic systems, in all cases, adding the prosodic system brings further improvements.

For the 3 s task, the best results were obtained when adding the prosodic system to the fusion of all acoustic systems (GMM+MAP+JFA+i-Vector+Prosodic), with $100 \cdot C_{avg} = 6.26$. This meant a 8.48% relative improvement over the same system without prosodic and formant information. Observe, that we have included the GMM system in this result. Nonetheless, if we drop the GMM system to include only the MAP based acoustic systems, the fusion with the prosodic system (MAP+JFA+i-Vector+Prosodic) still resulted in a relative improvement of 12.28%, with $100 \cdot C_{avg} = 6.86$ over the same system without prosodic and formant information.

In the 10 s task, the best performing fusion was the one including all MAP-based acoustic systems and the prosodic system, with $100 \cdot C_{avg} = 1.33$. This meant a 17.90% relative improvement over the same fusion without prosodic and formant information.

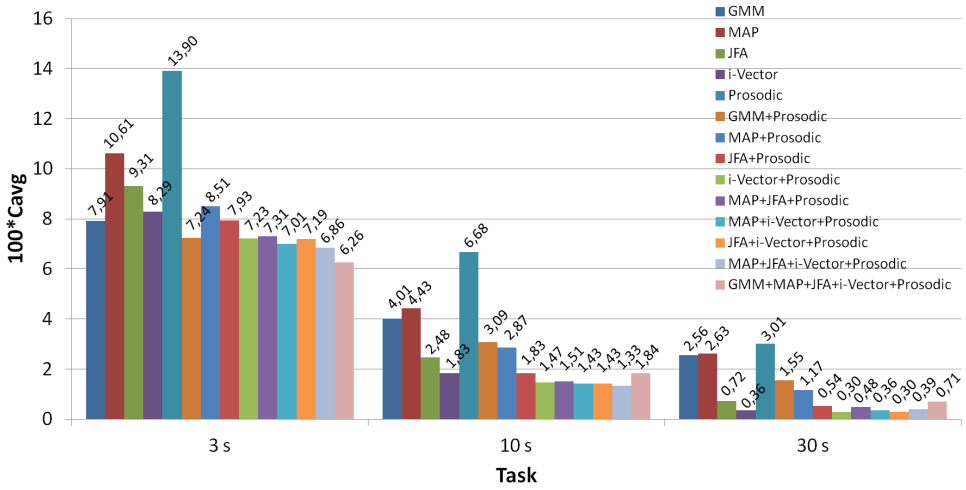


Figure 8.1: Results for fusion of acoustic and prosodic systems - GMM with 2048 components, GMM MAP with 2048 components, JFA with 200-dimension subspace and integral over channel distribution scoring, acoustic 600-dimension *i-Vector* system with Gaussian classifier, and prosodic and formant 600-dimension *i-Vector* system with pitch, energy, duration, F1 and F2, fixed segments of 200 ms and 10 ms shift, and order 5 polynomials. Results with 20 h for training and 1 h for development, per language, for the 3 s, 10 s, and 30 s tasks, in terms of $100 \cdot C_{avg}$.

As we already saw for the fusion of acoustic systems, adding the acoustic GMM-based system was harmful. Nonetheless, adding the prosodic system to the fusion including GMM was beneficial.

In the 30 s task, the prosodic system also helped. However, in this case, the best performing system was the fusion of acoustic and prosodic *i-Vectors*, with $100 \cdot C_{avg} = 0.30$. This meant a relative improvement over the acoustic *i-Vector* system alone of 16.67%. Adding JFA to this fusion did not change the result, but the additional inclusion of MAP or GMM was harmful.

8.2 Analysis by Languages

As final study, we analyze in detail the performance of one of the best performing fusions, namely the fusion of acoustic and prosodic *i-Vector*-based systems. This fusion performed the best in the 30 s task, very close to the best in the 10 s task, and although there were fusions performing better in the 3 s task (concretely the fusion

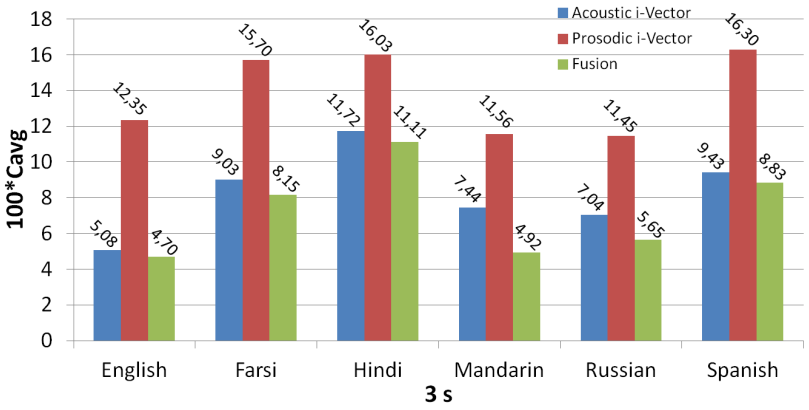


Figure 8.2: Results for fusion of acoustic and prosodic *i-Vector* systems for the 3 s task - 600-dimension *i-Vectors* with Gaussian classifier. Prosodic system includes pitch, energy, duration, and also F1, and F2, obtained in fixed segments of 200 ms and 10 ms shift, and order 5 polynomials. Results with 20 h for training and 1 h for development, per language, in terms of $100 \cdot C_{avg}$.

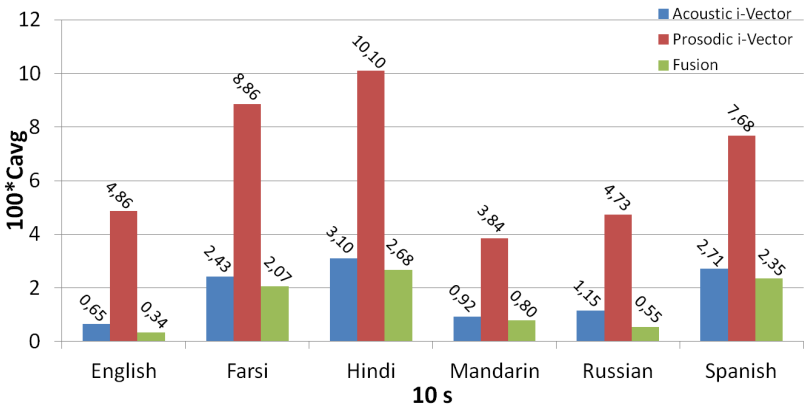


Figure 8.3: Results for fusion of acoustic and prosodic *i-Vector* systems for the 10 s task - 600-dimension *i-Vectors* with Gaussian classifier. Prosodic system includes pitch, energy, duration, and also F1, and F2, obtained in fixed segments of 200 ms and 10 ms shift, and order 5 polynomials. Results with 20 h for training and 1 h for development, per language, in terms of $100 \cdot C_{avg}$.

of all acoustic systems, including GMM, and prosodic system), we think that in a real scenario, this would be our preferred solution owing to consistency, simplicity and better generalization of conclusions.

First, C_{avg} can be seen for each language individually in Figures 8.2, 8.3, and 8.4, for the 3 s, 10 s, and 30 s tasks, respectively. If we compare these results with those

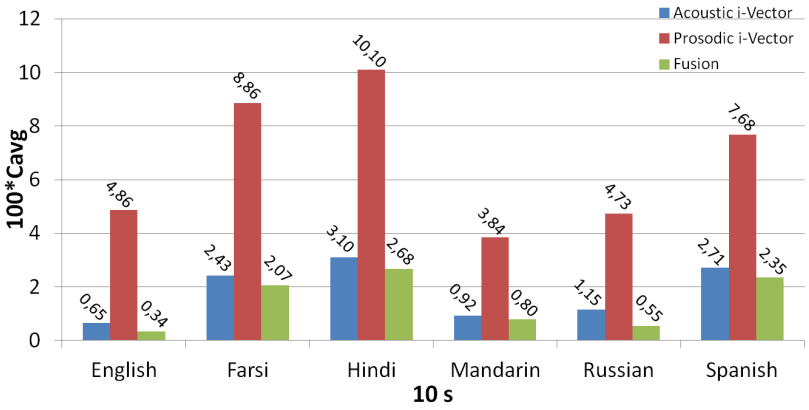


Figure 8.4: Results for fusion of acoustic and prosodic *i-Vector* systems for the 30 s task - 600-dimension *i-Vectors* with Gaussian classifier. Prosodic system includes pitch, energy, duration, and also F1, and F2, obtained in fixed segments of 200 ms and 10 ms shift, and order 5 polynomials. Results with 20 h for training and 1 h for development, per language, in terms of $100 \cdot C_{avg}$.

obtained with each *i-Vector* system individually (results of the prosodic system were already shown in Figure 7.7, but here are included again for an easier comparison), we see a clear advantage of the fusion for all languages in all tasks, except for English and Spanish on the 30 s task, for which the acoustic *i-Vector*-based system performed slightly better. However, we already saw above that in general, the fusion was beneficial. Especially remarkable were the benefits for Mandarin, which could be expected, since Mandarin is a tonal language, and as we saw in Section sec:ProsResults, the prosodic and formant system performed very well for this language. Therefore, the fusion of acoustic and prosodic system could be expected to be richer for Mandarin than for the rest of languages. Surprisingly, Russian was the other language for which the fusion performed best.

In the 3 s task, the language that most benefited from the fusion was Mandarin, with a 33,87% relative improvement over the acoustic *i-Vector* system alone. Next, Russian, with a 19,74%. The rest of languages were between 5% and 10% relative improvements.

In the 10 s task, Russian with a 52.17%, and English with a 47.69%, were the languages with highest decrease of C_{avg} with the fusion with respect to the acoustic *i-Vector*-based system alone. The other languages were between 13% and 14% relative improvements.

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English CTS	72.73	4.96	6.61	4.13	2.48	2.48
English BNBS	89.43	2.11	0.91	1.51	0.91	1.21
Farsi CTS	6.09	75.63	4.06	3.05	3.05	1.02
Farsi BNBS	1.44	70.19	4.81	6.73	4.81	1.92
Hindi CTS	1.43	5.71	65.71	10.00	4.29	2.86
Hindi BNBS	3.16	7.76	55.17	2.01	4.60	9.20
Mandarin CTS	0.77	3.86	3.47	80.31	2.70	1.54
Mandarin BNBS	0.00	1.73	1.73	89.60	0.00	1.16
Russian CTS	3.60	3.60	3.60	2.16	76.98	1.44
Russian BNBS	0.66	2.32	2.98	0.33	85.43	2.65
Spanish CTS	6.06	6.06	15.58	3.46	6.06	49.35
Spanish BNBS	2.66	5.32	4.79	0.53	4.26	77.13

Table 8.1: Confusion matrix of the acoustic and prosodic fusion *i-Vector*-based LID system for the 3 s task - 600-dimension *i-Vectors* with Gaussian classifier. Results with 20 h for training and 1 h for development, per language. Rows are true spoken language and transmission channel, and columns are decisions made by our system (in % of files).

In the 30 s tasks, Russian with 57.89%, Mandarin with 44.44%, and Hindi with 34.89% relative improvements of the fusion with respect to the acoustic *i-Vector*-based system alone, were the languages that most benefited from the fusion. On the other hand, C_{avg} for English was increased a 26.67% in the fusion, with respect to the acoustic *i-Vector*-based system alone, and C_{avg} for Spanish was increased a 7.69%.

Next, we show the confusion matrices for the fusion of acoustic and prosodic *i-Vector* systems. In this case, we split the results into CTS and BNBS transmissions, to see if there were common patterns that help to understand better the results. They are in Tables 8.1, 8.2, and 8.3, for the 3 s, 10 s, and 30 s tasks, respectively.

In the three tasks, the most clear unbalanced result was for Spanish. In the 3 s task, 77.13% of BNBS files were correctly classified, while only 49.35% of CTS files were correctly classified. In the 10 s and 30 s tasks, this difference was progressively reduced. The most reasonable explanation we find for this result is that, unlike the rest of languages, we did not have LRE09cts data in the *Dev* dataset (see Table 4.2). Probably, these data helped a lot to make more robust models in the rest of languages, and Spanish could not benefit from this.

In the 3 s task, English also presented worse results for CTS than for BNBS. There

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English CTS	96.69	0.83	1.65	0.00	0.00	0.00
English BNBS	98.93	0.36	0.00	0.00	0.00	0.00
Farsi CTS	0.00	95.43	1.02	0.51	0.51	0.51
Farsi BNBS	0.00	85.65	0.96	3.35	0.96	0.96
Hindi CTS	0.00	1.43	95.71	1.43	0.00	0.00
Hindi BNBS	0.36	0.91	89.29	0.36	1.09	1.09
Mandarin CTS	0.00	0.00	1.16	97.68	0.00	0.00
Mandarin BNBS	0.00	0.00	0.41	99.59	0.00	0.00
Russian CTS	0.00	0.00	0.72	0.00	97.84	0.00
Russian BNBS	0.00	0.00	0.33	0.00	99.01	0.00
Spanish CTS	0.43	2.16	5.19	1.73	0.00	81.82
Spanish BNBS	0.00	0.53	1.60	0.00	0.53	94.68

Table 8.2: [Confusion matrix of the acoustic and prosodic fusion *i-Vector*-based LID system for the 10 s task] - 600-dimension *i-Vectors* with Gaussian classifier. Results with 20 h for training and 1 h for development, per language. Rows are true spoken language and transmission channel, and columns are decisions made by our system (in % of files).

	English	Farsi	Hindi	Mandarin	Russian	Spanish
English CTS	100.00	0.00	0.00	0.00	0.00	0.00
English BNBS	98.00	0.00	0.00	0.00	0.00	0.00
Farsi CTS	0.00	98.98	0.51	0.00	0.00	0.00
Farsi BNBS	0.00	96.14	0.00	0.00	0.00	0.00
Hindi CTS	0.00	0.00	100.00	0.00	0.00	0.00
Hindi BNBS	0.00	0.00	97.24	0.00	0.69	0.00
Mandarin CTS	0.00	0.00	0.00	99.61	0.00	0.00
Mandarin BNBS	0.00	0.00	0.00	100.00	0.00	0.00
Russian CTS	0.00	0.00	0.00	0.00	100.00	0.00
Russian BNBS	0.00	0.00	0.00	0.00	100.00	0.00
Spanish CTS	0.43	0.87	0.87	0.00	0.00	94.37
Spanish BNBS	0.00	0.00	0.00	0.00	0.00	99.47

Table 8.3: [Confusion matrix of the acoustic and prosodic fusion *i-Vector*-based LID system for the 30 s task] - 600-dimension *i-Vectors* with Gaussian classifier. Results with 20 h for training and 1 h for development, per language. Rows are true spoken language and transmission channel, and columns are decisions made by our system (in % of files).

is one database, SRE08, that was present in the rest of languages except English. This could influence the performance in the 3 s task. For the 10s, and 30 s task, we did not see such a clear difference.

For Hindi, we did not include CALLFRIEND data since preliminary experiments with GMMs seemed to indicate a worse performance when this database was included. In the results, CTS models seemed to be working better than BNBS models.

A large percentage of Spanish CTS files (15.58%) were confused with Hindi in the 3 s task. In addition to the lack of Spanish LRE09cts data, perhaps, this type of errors could have been accentuated by the fact of not having included CALLFRIEND for Hindi. The problem persisted in the 10 s task, with 5.19% of Spanish CTS files misclassified as Hindi.

There were other confusions between pairs of languages for a specific transmission channel that we can not fully explain, like the 10% of Hindi files misclassified as Mandarin. In addition to language differences, probably, they are just due to database selection. Thus, some languages benefited from one database more than others just because the audio files were cleaner, or there were less labeling errors. For example, we know that VOA3 contains a non-negligible number of English recordings labeled as other languages. We did a cleaning process to alleviate this problem [Martínez et al., 2011c], but probably the use of VOA3 benefited more English than the rest of languages.

As final note, we want to remark again the very good results obtained, especially for the 30 s task, where the lowest correct classification rate was 94.37%, and all confusion rates were below 1%. Also, in the 10 s task, all but 3 correct classification rates were over 94%. Probably, the biggest effort from now on should be put on the 3 s task. The short duration of audio recordings makes this problem a very challenging and interesting one.

9

Results on 2009 NIST LRE Database

Contents

In this chapter, we present results on the whole 2009 NIST LRE database [National Institute of Standards and Technology (NIST), 2009] using the techniques developed in this Thesis. Actually, in the last years many researchers have used this database in his/her experiments, and therefore, the results reported in this chapter will allow an easier comparison with other works. Remember that in previous chapters, we only used 6 target languages for which we could collect specific number of hours for train and development datasets. However, now we do not restrict the size of train and development, but we use all available data. The whole database includes 23 target languages, and the number of hours per language is unbalanced. Thus, this is a more challenging problem, and we will be able to see if the conclusions previously seen hold in other scenarios. Additionally, we will present results using a flat prior for all languages, as in previous chapters, but also using a prior equal to 0.5 for the target language and 0.5 split among the rest of languages (remember we perform a binary detection task where each each audio file is evaluated against each target language individually), which is the common strategy in NIST evaluations adopted in many works of the literature.

9.1 2009 NIST LRE database

We report results on 2009 NIST LRE database. This database includes 41793 files totaling 40 different languages. We only focused on the 23 target languages of the closed-set task, what reduces the number of files to 31178. The channel type can be CTS or BNBS. The distribution of files belonging to the target languages among the 3 s, 10 s, and 30 s tasks can be seen in Table D.1 of Appendix D. Also, in order to see which languages can be more confusable, we classify them by families in Table 9.1.

9.2 Training Database

For training, we used all available data that we had. All of them have been distributed by NIST in LRE evaluations. In Table D.2 of Appendix D, the databases used for training indicating the number of files and time of speech (after VAD) for each language are reflected. 57 different languages are included in these databases (including *English Others*, which refers to English considered to be neither American nor Indian). All databases are CTS except VOA3, which is entirely BNBS, and 2011 NIST LRE (we

Family	Branch	Languages				
Afro-Asiatic	**Chadic**	Hausa				
	Semitic	Amharic				
Altaic	**Turkic**	Turkish				
Austro-Asiatic	**Mon-Khmer**	Vietnamese				
Creole	**French-Creole**	Creole Haitian				
Indo-European	**Germanic**	American English	Indian English			
	Indo-Iranian	Dari	Farsi	Hindi	Pashto	Urdu
	Italic	French	Portuguese	Spanish		
	Slavic	Bosnian	Croatian	Russian	Ukrainian	
Kartvelian	**Georgian**	Georgian				
Korean	**Korean**	Korean				
Sino-Tibetan	**Sinitic**	Cantonese	Mandarin			

Table 9.1: Distribution of languages into families.

have split it into development and evaluation in the table), which includes both CTS and BNBS. Thus, both channel types seen in the test data were seen in the training data. VOA3 received a special treatment, the same as explained in Section 4.4. The evaluation part of 2011 NIST LRE was only used for those languages with less amount of data in the other databases like Turkish or Ukrainian, with languages without other BNBS data like Indian English, and with Urdu, a very challenging language because it is very highly confusable with Hindi.

9.3 Development Database

For development, we reserved data from 2007 NIST LRE, 2011 NIST LRE, and VOA3, which were not used during training. As we explained in Chapter 5, development data was used to train the calibration and fusion blocks. As we did in Section 4.4, VOA3 was segmented into 3 s, 10 s, and 30 s files to match better the test dataset. In this case, we used all data available in the evaluation part of 2011 NIST LRE belonging to the target languages. Note that CTS data were represented by 2007 NIST LRE and part of 2011 NIST LRE, while BNBS data were represented by VOA3 and part of 2011 NIST LRE. In Table D.3 of Appendix D, we can see the number of files and hours of speech (after VAD) for the databases and languages used in our experiments for development. 54 different languages were included (including *English Others*).

	CALLFR	OHSU	SRE04	SRE06	SRE08	SRE10	SWCHBR	LRE03	LRE05	LRE07t	LRE11d	LRE11e	VOA3
UBM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
i-Vector Subspace	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
i-Vector Classifier	✓	✓	✓	✓	✓			✓	✓	✓		✓	✓

Table 9.2: Data distribution in the training process of the acoustic system.

	CALLFR	OHSU	SRE04	SRE06	SRE08	SRE10	SWCHBR	LRE03	LRE05	LRE07t	LRE11d	LRE11e	VOA3
UBM	✓	✓	✓	✓	✓			✓	✓	✓			✓
i-Vector Subspace	✓	✓	✓	✓	✓			✓	✓	✓		✓	✓
i-Vector Classifier	✓	✓	✓	✓	✓			✓	✓	✓		✓	✓

Table 9.3: Data distribution in the training process of the prosodic system.

9.4 Experimental Setup

We fused our acoustic system based on *i-Vectors*, including a UBM with 2048 Gaussian components, 600-dimension *i-Vectors*, and Gaussian classifier, and our prosodic and formant information systems based on *i-Vectors*, including pitch, Energy, F1, and F2 features, fixed 200 ms long segments shifted every 10 ms, order 5 Legendre polynomials, 2048 Gaussian components, 600-dimension *i-Vectors*, and Gaussian classifier.

We did not use the full training database at all steps of the training process, and different data distribution was used for the acoustic and prosodic systems. Basically, for training the UBM we used data of all available languages, for training the *i-Vector* extractor we used data of all available languages for the acoustic system, and only data of the target languages for the prosodic system, while for training the *i-Vector* classifier, we used only data of the target languages in both approaches. The difference could help in the fusion, although no big variations were expected with respect to a scenario where both system were trained with exactly the same configuration. In Tables 9.2 and 9.3, we indicate which databases were used for each system at each step of the training process.

For calibration and fusion, we want to remark that, although our development database included nontarget languages, these were only used to train the covariance matrix of the generative calibration step.

9.5 Results

In this Section, we present the results for the acoustic, prosodic with formant information, and fusion systems. Most of the results in the literature working with the

Language	3 s			10 s			30 s		
	Ac	Pr	Ac+Pr	Ac	Pr	Ac+Pr	Ac	Pr	Ac+Pr
Amharic	11.09	22.64	8.76	2.93	9.58	2.18	1.60	5.53	0.99
Bosnian	21.87	27.08	19.17	7.40	15.11	6.91	3.07	9.19	3.29
Cantonese	10.17	18.18	8.62	3.37	6.84	2.34	1.46	3.75	0.99
Creole Haitian	15.24	24.67	13.18	4.88	11.83	3.58	2.77	7.52	2.37
Croatian	18.31	30.99	16.91	6.69	15.45	5.93	4.03	11.26	3.76
Dari	18.60	31.04	16.16	8.08	17.58	7.37	5.11	11.15	4.38
Am. English	14.83	26.57	13.32	4.47	14.27	4.10	2.53	6.89	2.29
In. English	9.45	22.89	8.55	2.70	10.45	2.19	1.46	5.23	1.72
Farsi	13.27	27.47	11.75	3.71	14.70	3.36	2.74	8.42	2.65
French	16.19	26.06	13.97	5.72	13.22	4.61	2.94	7.07	1.89
Georgian	12.94	22.16	10.01	3.24	11.67	2.84	1.48	8.14	1.16
Hausa	16.13	22.04	12.37	2.86	8.94	1.49	0.65	4.09	0.38
Hindi	18.82	27.29	16.42	7.46	15.18	5.66	4.38	10.08	4.32
Korean	11.04	26.66	9.75	1.82	12.63	1.28	0.18	4.23	0.19
Mandarin	10.32	19.82	8.23	2.04	6.45	1.18	0.44	2.24	0.20
Pashto	18.99	28.10	17.25	8.08	18.14	7.15	1.98	9.05	1.76
Portuguese	16.04	21.35	11.59	4.09	10.12	2.42	1.02	5.64	0.76
Russian	12.68	24.75	12.56	4.76	13.76	4.24	3.00	8.69	2.95
Spanish	12.76	27.08	12.34	3.71	14.21	2.22	0.91	6.95	0.55
Turkish	16.61	22.22	13.36	3.90	9.43	3.00	1.25	5.20	0.70
Ukrainian	23.04	32.83	22.40	11.29	19.49	10.36	4.94	14.39	6.01
Urdu	16.45	27.26	15.34	6.52	16.05	5.72	3.92	8.68	3.51
Vietnamese	10.60	15.38	7.98	2.77	6.93	1.55	0.85	3.04	0.51
Average	**15.02**	**24.98**	**13.04**	**4.89**	**12.70**	**3.99**	**2.29**	**7.24**	**2.06**

Table 9.4: Results on the 2009 NIST LRE dataset using 0.5 prior - $100 \cdot C_{avg}$ for the 2009 NIST LRE dataset with 0.5 prior, for acoustic (Ac), prosodic (Pr), and fusion (Ac+Pr) systems.

2009 NIST LRE database are given with a prior equal to 0.5 for the language under evaluation, and the other 0.5 is split among the rest of target languages. In previous experiments, we used a flat prior, where all languages have the same prior. For easiness of comparison with the rest of works in the literature, we will give the results with the 0.5 prior, but for consistency with our arguments in 4.3 and with the previous experiments in this Thesis, we will also give results with the flat prior, in this case equal to $\frac{1}{23}$.

In Table 9.4, we give the results for the 3 s, 10 s, and 30 s task, for the acoustic, prosodic, and fusion systems, using a prior equal to 0.5. First, it is clear that the acoustic system performed better than the prosodic one, and that the fusion helped. Specifically, there was a relative improvement with the fusion over the acoustic system alone of 13.18% in the 3 s task, of 18.40% in the 10 s task, and of 10.04% in the 30

s task. It can be checked that these percentages are in agreement with the results obtained in Chapter 8.

Some languages were better discriminated than others. However, this also depended on the duration of the test file. For example, for the 3 s task, the best results obtained with the fusion system were for Vietnamese, Mandarin, Indian English, Cantonese, and Amharic, with a $C_{avg} < 9\%$ for all of them. For the 10 s task, the best results obtained with the fusion system were for Mandarin, Korean, Hausa, and Vietnamese, all of them with $C_{avg} < 2\%$. Finally, for the 30 s task, the best results obtained with the fusion system were for Korean, Mandarin, Hausa, Vietnamese, and Spanish, all of them with $C_{avg} < 0.6\%$. In addition, there were some languages with much worse performance, mainly due to the similarity with other languages. Concretely, Bosnian and Croatian, Hindi and Urdu, or Russian and Ukrainian were among the most difficult pairs (confusion matrices for the 3 s task can be seen in Appendix C for the acoustic, prosodic, and fusion systems). Nonetheless, we also observe clear differences between the acoustic and prosodic systems for some languages. For example, for Indian English or Georgian, the acoustic system performed very well, but the prosodic system was not so outstanding, and the fusion results were not among the best ones. In other cases where the tone plays a fundamental role in the language, like Mandarin, Cantonese, Vietnamese or Korean, the prosodic system obtained very good results, the acoustic was also good, and consequently, these languages were among the best discriminated also with the fusion system.

In Table 9.5, the results are given using a flat prior. Remember from Section 4.2.2 that this affects the computation of the posterior probability of the target language (P_t), and also the calculation of C_{avg}, seen in eq. (4.14). First, it can be observed that the numbers are much lower than in Table 9.4. The number of errors was drastically reduced, mainly because the number of false alarms was dramatically reduced, at the expense of increasing the number of misses. Interestingly, the differences between the acoustic and prosodic system were reduced, and e.g. for Bosnian, the prosodic system performed even better than the acoustic system in the 3 s task. The relative improvements of the fusion with regard to the acoustic system were of 7.06% for the 3 s task, 8.51% for the 10 s task, and 3.25% for the 30 s task. These numbers are slightly under those relative improvements seen in Chapter 8 obtained with same prior. Nevertheless, the improvement of the fusion of acoustic and prosodic systems is consistent.

Language	3 s			10 s			30 s		
	Ac	Pr	Ac+Pr	Ac	Pr	Ac+Pr	Ac	Pr	Ac+Pr
Amharic	2.81	4.08	2.51	0.67	2.50	0.40	0.22	0.93	0.15
Bosnian	5.15	4.63	4.57	4.06	4.15	3.73	3.42	3.83	3.21
Cantonese	2.85	3.82	2.16	0.70	1.68	0.52	0.26	0.68	0.16
Creole Haitian	3.30	4.32	3.13	1.63	2.72	1.49	0.87	1.47	0.86
Croatian	4.31	4.54	4.21	4.06	4.44	3.75	3.43	4.02	3.24
Dari	4.18	4.34	3.99	2.89	4.10	2.71	1.83	3.89	1.83
Am. English	4.02	4.86	3.75	2.03	3.87	1.94	1.09	3.56	0.91
In. English	2.80	4.69	2.71	1.27	2.79	1.13	0.96	2.60	0.87
Farsi	3.51	4.38	3.55	1.67	3.61	1.81	1.03	2.77	1.15
French	3.90	4.44	3.60	2.04	3.70	1.73	1.22	2.52	0.99
Georgian	2.86	4.19	2.60	1.24	2.67	0.92	0.28	1.52	0.28
Hausa	3.01	4.06	2.62	0.84	2.11	0.74	0.24	1.06	0.13
Hindi	4.39	4.44	4.47	3.89	4.35	3.88	3.36	4.28	3.28
Korean	2.68	4.34	2.41	0.75	3.10	0.56	0.10	1.79	0.04
Mandarin	2.82	4.22	2.14	0.66	2.26	0.37	0.14	1.13	0.12
Pashto	4.22	4.40	4.02	2.58	3.81	2.30	1.41	3.61	1.13
Portuguese	3.20	4.16	2.84	1.01	3.09	0.80	0.23	1.57	0.13
Russian	3.75	4.70	3.74	2.01	3.76	2.18	1.74	3.03	2.24
Spanish	3.18	4.70	3.13	0.80	3.46	0.71	0.08	2.60	0.11
Turkish	3.40	3.97	2.91	1.15	2.29	0.71	0.11	0.67	0.08
Ukrainian	4.46	4.61	4.45	3.36	4.19	3.18	2.84	4.00	3.20
Urdu	4.19	4.31	4.04	3.37	3.99	3.48	2.94	3.62	3.02
Vietnamese	2.49	3.79	2.06	0.67	1.79	0.50	0.41	0.74	0.20
Average	**3.54**	**4.35**	**3.29**	**1.88**	**3.24**	**1.72**	**1.23**	**2.43**	**1.19**

Table 9.5: Results on the 2009 NIST LRE dataset using flat prior - $100 \cdot C_{avg}$ for the 2009 NIST LRE dataset with flat prior, for acoustic (Ac), prosodic (Pr), and fusion (Ac+Pr) systems.

Next, we want to show how important the similarity between development and test is. In short, we found that results can vary significantly depending on if these two datasets were more or less similar. In the previous experiments, and in experiments reported in previous chapters, we used all available data in development to train the calibration and fusion blocks. However, in Table 9.6, we show results where, for the 30 s task, we only used development files marked as 30 s and higher, for the 10 s task, we only used development files marked as 10 s and higher, and for the 3 s task, we used all available development data. For clarity, we only show results for a prior equal to 0.5. First, note that for the 3 s task, the experiment was the same as in Table 9.4, because we used the same development data in the two cases. However, for the 10 s and 30 s tasks, results improved in all cases except for the fusion of the 10 s task, which was approximately the same when we removed from development files with a duration

Language	3 s			10 s			30 s		
	Ac	Pr	Ac+Pr	Ac	Pr	Ac+Pr	Ac	Pr	Ac+Pr
Amharic	11.09	22.64	8.76	2.11	8.74	1.78	1.32	3.49	1.15
Bosnian	21.87	27.08	19.17	9.97	17.26	9.45	3.55	9.82	3.19
Cantonese	10.17	18.18	8.62	2.28	6.03	1.74	0.42	1.53	0.30
Creole Haitian	15.24	24.67	13.18	4.63	11.70	3.69	2.15	5.18	1.57
Croatian	18.31	30.99	16.91	8.71	15.90	8.09	4.63	8.21	4.23
Dari	18.60	31.04	16.16	8.12	18.26	7.39	5.53	10.73	5.07
Am. English	14.83	26.57	13.32	4.88	13.99	4.76	2.27	5.92	2.23
In. English	9.45	2.289	8.55	2.41	9.61	2.22	1.45	4.78	1.34
Farsi	13.27	27.47	11.75	3.21	14.63	2.94	1.92	6.96	1.77
French	16.19	26.06	13.97	5.78	12.74	4.29	1.91	5.02	1.43
Georgian	12.94	22.16	10.01	2.99	10.87	2.48	1.07	4.94	0.53
Hausa	16.13	22.04	12.37	2.73	8.86	1.69	0.60	3.86	0.66
Hindi	18.82	27.29	16.42	6.89	14.91	5.49	3.81	916	3.88
Korean	11.04	2666	9.75	1.61	12.69	1.18	0.09	4.46	0.14
Mandarin	10.32	19.82	8.23	1.91	6.30	1.03	0.30	2.03	0.13
Pashto	18.99	28.10	17.25	7.60	17.84	6.90	2.09	7.12	1.93
Portuguese	16.04	21.35	11.59	3.92	10.85	2.16	0.50	4.03	0.40
Russian	12.68	24.75	12.56	4.45	13.32	3.89	2.52	6.63	2.54
Spanish	12.76	27.08	12.34	3.13	14.56	2.01	0.65	5.49	0.45
Turkish	16.61	22.22	13.36	3.85	9.12	2.89	0.54	2.34	0.31
Ukrainian	23.04	32.83	22.40	10.97	20.42	9.60	3.11	20.02	5.42
Urdu	16.45	27.26	15.34	5.88	15.70	5.46	3.07	8.90	3.20
Vietnamese	10.60	15.38	7.98	2.32	5.77	1.11	1.20	2.00	0.53
Average	**15.02**	**24.98**	**13.04**	**4.80**	**12.61**	**4.01**	**1.94**	**6.20**	**1.84**

Table 9.6: Results on the 2009 NIST LRE with a development including files with an equal or longer duration than test files - $100 \cdot C_{avg}$ for the 2009 NIST LRE dataset using a development dataset including files with a duration equal or longer than the duration of the test files, with 0.5 prior, for acoustic (Ac), prosodic (Pr), and fusion (Ac+Pr) systems.

shorter than the duration of the test files. The most notable case was the 30 s task, where the relative improvement of the fusion was of 10.68% compared to the case of Table 9.4, where we used all available development data regardless of the file duration. This experiments indicate that for development, it is beneficial to include all files with a duration equal or longer than the duration of the files that will be seen in test. The rest of conclusions about which languages were performing best, and which pairs were more confusable are the same as for the experiment with a development dataset including shorter files than the test dataset. For more information, we have included the confusion matrices of the experiments reported in Table 9.6 in Appendix C. These matrices confirm the conclusions about the most confusable language pairs, and about the performance of the acoustic, prosodic, and fusion systems for each language.

10

Conclusions

In this part of the Thesis, we have presented LID systems using acoustic and prosodic features. Acoustic systems are the most widely used in the community because they offer a very good performance. Only the use of phonetic information has provided similar performance in some cases [Soufifar et al., 2011; Diez et al., 2012; D'Haro et al., 2014]. On the other hand, we have shown that prosodic and prosodic features are complementary.

In our first experiments, we have studied the performance of several linear Gaussian models for LID. We have shown how the introduction of subspace modeling allows a big reduction in the error rates. Subspace modeling has been used for compensating the variability that exists in utterances of the same language. Factors like the channel, or the phonetic content in an utterance, introduce undesired uncertainty in the signal that we want to remove. In this direction, JFA was used with a single factor to model this type of variability (it can also be seen as an eigenchannel model) and it performed very good. Finally, the use of *i-Vectors*, a fixed-length low-dimension representation of the utterance, beat all previous techniques. It was shown that the channel factor of a JFA model not only contained information about the channel, but also about the language. Thus, modeling all sources of variability together (language, channel, ...) in a single subspace avoided this problem. This subspace is known as total variability or *i-Vector* subspace. The improvements of *i-Vectors* with respect to JFA were 11% in the 3 s task, 26% in the 10 s task, and 50% in the 30 s task, and with respect to a MAP-GMM without subspace modeling, they raised to 22% in the 30 s task, 59% in the 10 s task, and 86% in the 30 s task. An interesting result was that a GMM model

trained without MAP adaptation performed better then GMMs adapted via MAP for all duration tasks, and even better than both JFA and *i-Vectors* for the 3 s task. This suggests for future work that the models of each language could be better trained directly without MAP adaptation, if the number of hours for training each language is large enough.

We also presented an *i-Vector*-based system with prosodic features and formant information. We used pitch, energy, and duration to capture prosody, and the central frequency of the first two formants to include formant information. We modeled the contour of these features along time. Different intervals of time were studied, and we observed that the best choice was to use fixed segments of 200 ms. The contours were modeled with Legendre polynomials of order 5, and the coefficients of each polynomial were the input to our *i-Vector* system. The results were not as good as with acoustic features. However, the fusion of both sources of information was complementary. Relative improvements of 12.28%, 17.90%, and 16.67% with respect to the acoustic system alone were obtained in the 3 s, 10 s, and 30 s tasks, respectively.

The results with the *i-Vector* were contrasted using a subset of NIST LRE 2011 dataset including 6 target languages, and the full NIST LRE 2009 database. The conclusions were similar with both. This indicates that the results are consistent and can be generalized.

One of the most important contributions of this part of the Thesis is the proposed system architecture for *i-Vector* classification. We studied different generative and discriminative classifiers and one of the most successful was the Gaussian classifier. In this classifier, every language is modeled by a Gaussian distribution with full-covariance matrix. The covariance is shared among the different classes and it is equal to the WC covariance matrix of the training data. This classifier has become very popular among the community thanks to its simplicity and good results, and it can be considered the state of the art system in LID.

Part III

Intelligibility Assessment for Dysarthric Speakers

11

State of the Art

11.1 Intelligibility Assessment of Dysarthric Speakers

Intelligibility assessment of dysarthric speakers is a very interesting topic that has been widely studied among speech therapist researchers. Traditionally, the assessments have been made perceptually [Beukelman and Yorkston, 1979; Doyle et al., 1997]. However, the use of computers to make automatic assessments has not been investigated in depth until some years ago, probably due to the scarcity of available databases. Since the publication of different databases freely accessible to the community, such as Whitaker database of dysarthric speech [Deller Jr. et al., 1993], Nemours [Menendez-Pidal, 1996], UASpeech [Kim et al., 2008], a Korean database of dysarthric speech [Choi et al., 2011], or TORGO [Rudzicz et al., 2011], the number of investigations in this field is increasing.

Basically, two main approaches are found in the literature for automatically predicting intelligibility of dysarthric speakers. In the first, speech intelligibility is calculated directly from the word accuracy rate (*Accuracy*) obtained from an ASR system. That is, it is considered that intelligible speech will obtain high *Accuracy* on an ASR system trained on clean and presumably highly intelligible speech, and low intelligible speech will obtain low *Accuracy* [Doyle et al., 1997; Carmichael and Green, 2004; Sharma et al., 2009; Christensen et al., 2012]. One of the main weaknesses of these systems is that they are trained only on non-dysarthric speakers and the result can be unpredictable for very severe subjects [Middag et al., 2009]. In the second, different features are extracted from speech and used to build an intelligibility assessment system [De Bodt et al., 2002; Middag et al., 2011; Falk et al., 2011, 2012; Paja and Falk, 2012; Bocklet

et al., 2012]. These experiments are supported by perceptual studies that show how intelligibility can be expressed as a linear function of multiple speech dimensions [De Bodt et al., 2002]. In this approach, the use of a speech recognizer or of an automatic speech alignment system is restricted to feature extraction [Van Neuffelen et al., 2009; Middag et al., 2009].

In the following paragraphs, we summarize in chronological order the most remarkable works found in the literature related with intelligibility assessment.

In 1997, Doyle compared the accuracy of an ASR system with the recognition of human listeners [Doyle et al., 1997]. At that time, the performance of the human listeners was clearly superior to the computer-based system.

Menéndez-Pidal presented in 1997 results in word recognition accuracy on the Nemours database of dysarthric speech, comparing human performance with an HMM-based ASR [Menéndez-Pidal et al., 1997]. The results indicated that human performance was much higher than the automatic system. However, results were correlated, indicating that HMM fail or succeed in a similar pattern to humans. In this case, ASR results were directly considered to be a metric of intelligibility of the speaker.

In 2002, De Bodt showed that intelligibility of dysarthric speakers could be expressed as a linear function of 4 dimensions of speech: voice quality, articulation, nasality, and prosody [De Bodt et al., 2002]. Articulation was found to be the most correlated one.

In 2004, Carmichael also performed a similar comparison [Carmichael and Green, 2004]. He used an HMM-ASR system for the automatic recognition of words uttered by dysarthric speakers. The system was inconsistent and in general, far from human listeners.

In 2005, Liu showed that a group of young adult Mandarin speakers with cerebral palsy and dysarthria exhibited smaller vowel working space, in the sense that their formant F1 and F2 frequencies were confined to a smaller interval of values than non-dysarthric speakers [Liu et al., 2005]. They found that the vowel working space area was correlated with the vowel intelligibility and word intelligibility.

Sharma in 2009 used an ASR system with dysarthric speakers, and better results were obtained with the automatic system than with the perceptual ratings in small size vocabularies, while in medium size vocabularies, the performance was similar [Sharma

et al., 2009]. In this case, the performance of the ASR was considered to be a direct metric of speech intelligibility. He showed the first results on UASpeech database.

In 2009, Middag presented an automatic system to predict intelligibility of pathological voices [Middag et al., 2009]. She worked on a Dutch corpus that also included dysarthric speech among other speech pathologies and impairments. She calculated phonological features and aligned them to speech. Each phonological feature contained the posterior probabilities of 24 different phonological classes. Root-mean-square error (RMSE) was below 8.

Middag continued her previous work in 2011 [Middag et al., 2011]. She presented an ASR-free system based on acoustic and phonological features. She obtained better results with an SVR system than with linear regression, with correlations of up to 0.74. She used the same data as in her previous work, which contained dysarthric speech among other voice pathologies.

In 2011, Falk assessed the intelligibility of spastic dysarthric speakers using short-term and long-term temporal dynamics [Falk et al., 2011]. For the first case, he used the zeroth order cepstral coefficient as a measure of short-term log-spectral energy, and the zeroth order delta coefficient as a measure of log-energy change. For the long-temporal dynamics he used two representations of the modulation spectrum. As additional metric, he used the total number of voiced frames in an utterance. He worked on the UASpeech database and obtained a correlation of 0.87. In this case, data of the speakers used for testing were available also for training. This is an optimal scenario, and we will analyze this problem in depth in this Thesis.

Falk extended his work in 2012 [Falk et al., 2012]. He added to the previous features linear prediction residual as a measure of vocal source information, linear prediction parameters to characterize the vocal tract shaping, kurtosis of linear prediction residuals as a measure of atypical vocal source excitation, the two first formants and their bandwidths to have information about nasality, and different pitch statistics to track prosody. Finally, he composed all the features into a single metric by linear regression, as De Bodt proposed in De Bodt et al. [2002]. He obtained correlations about 0.95. Paja developed further this system by dividing the utterances into mid-low and mid-high severity, previous the the intelligibility assessment [Paja and Falk, 2012].

Bocklet in 2012 experimented with Gaussian mean supervectors to assess the intelligibility of people after laryngeal cancer that were partially of totally laryngectomized

[Bocklet et al., 2012]. Even though they did not work with dysarthric speakers, what they did is relevant to our work, in the sense that *i-Vectors* come from a GMM-based system. They obtained correlations above 0.8.

Also in 2012, Kim designed a system to automatically select the best phonetic features from a pre-established set [Kim and Kim, 2012]. He divided the features between model-based and model-free, if they were computed from an HMM or not, respectively. He finally selected five: word recognition rate given by the HMM, kurtosis of the log-likelihood given by the HMM, variance of spectral roll-off, kurtosis of the ratio between spectral flatness and spectral centroid, and skewness of zero-crossing rate, where the first two were model-based features and the last three were model-free features. He obtained a RMSE of 10.8. He worked on word recordings of 94 Korean speakers.

In 2015, Kim developed an intelligibility classification system of pathological speech [Kim et al., 2015]. He worked with TORGO database, which includes dysarthric speech. The data were divided into intelligible and non-intelligible. Several features at sentence-level for prosody, voice quality, and pronunciation, were extracted and fused at feature-level and at score-level. The best performing feature sets were the prosody and pronunciation ones, and the best fusion scheme was the score-level fusion. The classification of these two classes reached a performance of 73.5% in unweighted average recall with an SVM classifier and fusion at score-level of three subsystems, each built with one set of features. Sentence-level features were obtained by computing several moments of the distribution of the features. One drawback of these moments is that they fail to properly model multimodal data in sentence-level features. This work recalls ours because they also extracted sentence-level features. However, we think that our *i-Vector* features are more powerful to model a whole sentence because they are designed to work on multimodal data.

11.2 ASR Accuracy Prediction of Dysarthric Speakers

The other task addressed in this Thesis, the prediction of ASR accuracy for dysarthric speakers, has not been studied for so long time. The first reference we found dated of 2011. In Mengistu et al. [2011], the authors showed that the performance of an ASR system over spastic dysarthric speakers could be predicted with a set of acoustic

measures. They found good correlations with LPC kurtosis and skewness, LPC residual kurtosis, and F0-range. They worked on the UASpeech database. In this Thesis, we present another approach for this problem.

Out of the dysarthric speech field, performance of ASR has been investigated by some researchers. In Litman et al. [2000], the authors discovered that some prosodic features like F0, loudness, long pauses prior a speaker turn, and long turn durations, were indicative of ASR performance. In Hermansky et al. [2013], the authors showed that a mean temporal distance based on a mean difference between features in a given time interval, correlated well with the accuracy of an ASR.

Actually, the use of ASR is a very promising tool for dysarthric speakers, but its usability is limited to small tasks, and human performance is still clearly superior [Mengistu and Rudzicz, 2011]. The number of works investigating how to adapt a standard ASR system to dysarthric speech has increased in the last years [Raghavendra et al., 2001; Rudzicz, 2007; Sharma and Hasegawa-Johnson, 2010, 2013; Christensen et al., 2012, 2013]. We do not want to give a detailed historical review on this topic, since it is not the focus of our Thesis. We just want to highlight the work of Christensen in 2012 in Christensen et al. [2012], because it is the one in which we base our study of ASR prediction. In that work, the authors showed that ASR accuracy can be increased for dysarthric speakers via MAP adaptation.

12

Experimental Setup

Contents

In this chapter, we set the scenario for the experiments. First, we introduce the databases used for training and evaluation. Second, we describe the evaluation metrics to measure the performance of our system.

12.1 Audio Material

Two databases were used in the training process: UASpeech [Kim et al., 2008] and Wall Street Journal Database 1 (WSJ1) [Paul and Baker, 1991]. The first contains dysarthric speech, and the second was used for training ML models with large number of parameters that require large amounts of data, and it does not contain dysarthric speech. The sampling rate was fixed at 16 kHz in both. Next we describe them and explain how they were used in this study.

12.1.1 Universal Access Speech Database

This is a dysarthric speech database recorded from 19 speakers with cerebral palsy. We had data available from 15 of them. Data were recorded with an 8-microphone array at 48 kHz and 1 digital video camera. For each speaker, 765 words were recorded in 3 blocks of 255, 155 of which were common to the 3 blocks and 100 were uncommon words that differed across them. The 155-word blocks included 10 digits, 26 radio alphabet letters, 19 computer commands, and the 100 most common words in the Brown corpus of written English. To calculate the intelligibility rate of each speaker five naive listeners were asked to provide orthographic transcriptions of each word. The percentages of correct responses for each speaker obtained by the five listeners were averaged to calculate the speaker's intelligibility. In Table 12.1, a summary of each speaker in the database with their intelligibility can be seen. For more information about the database, please refer to Kim et al. [2008].

For our experiments, only microphone 6 was used, and two subsets were created, train and test. For testing, we reserved the uncommon subset (300 words per speaker), and for training, the rest (465 words per speaker). This configuration, proposed in Falk et al. [2012], permitted us to make fair experiments because the tested words were never seen during training.

Nr.	Speaker Label	Age	Speech Intelligibility (%)	Dysarthria Diagnosis
1	M04	>18	very low (2%)	Spastic
2	F03	51	very low (6%)	Spastic
3	M12	19	very low (7.4 %)	Mixed
4	M01	>18	very low (15%)	Spastic
5	M07	58	low (28%)	Spastic
6	F02	30	low (29%)	Spastic
7	M16	-	low (43%)	Spastic
8	M05	21	mid (58%)	Spastic
9	F04	18	mid (62%)	Athetoid
10	M11	48	mid (62%)	Athetoid
11	M09	18	high (86%)	Spastic
12	M14	40	high (90.4%)	Spastic
13	M10	21	high (93%)	Mixed
14	M08	28	high (93%)	Spastic
15	F05	22	high (95%)	Spastic

Table 12.1: UASpeech speaker information - In the first and second columns, we have the speaker identification number and label. In the third column, we have the speaker‘s age. In the fourth column, we have the speech intelligibility ratings given in the UASpeech database. In the fifth column, we have the type of dysarthria of each speaker. Athetoid dysarthria is associated to patients with athetoid cerebral palsy. This type of cerebral palsy is associated to damages in the basal ganglia that occur during brain development, and it is affected by hypotonia and hypertonia which provoke inability to control muscle tone, and difficulty to speak.

12.1.2 Wall Street Journal 1

This is a general-purpose English, large vocabulary, natural language, high perplexity corpus containing a substantial quantity of speech data (77800 training utterances totaling about 73 hours of speech). It includes read speech and spontaneous dictation by journalists. The database also contains development and test datasets in a “Hub and Spoke” paradigm to probe specific areas of interest. Each of them contains 7500 waveforms, about 11 hours of speech. Data were collected using two microphones at a sampling rate of 16 kHz. For more information please consult Paul and Baker [1991].

This database was selected because it contains a large amount of speech in American

English, like UASpeech, so we could train our ML models described in next section, the GMM and FA front-end, more reliably than using only UASpeech. In addition, both databases mostly contain read speech (the words in UASpeech are read from prompts). Only the clean speech of WSJ1 (from its training, development, and testing parts) was used, which totals 73.84 hours.

12.2 Application Evaluation Methods

At the time of building an assistive technology application, it is common that we know the patients that will use the system in advance. This is an ideal scenario because we can use pre-collected data of those speakers to train our system. However, this is not always the case, and in other situations we do not know who the users of our application will be. Nonetheless, the system should guarantee a high performance also in such cases. Unfortunately, the performance is not the same. We will see later that there is a significant difference between having and not having available data of the people that will be evaluated by our system for training. Traditionally, this second situation has not been studied in the literature due to the scarcity of data and small size of the available databases, and in our experiments we reported results for the two cases.

These two scenarios required two different training strategies. In the case where we included data of the application user for training, a single predictor was needed, trained on all the dysarthric speakers. In the case where we did not include data of the application user for training, 15 predictors were built on a leave-one-out strategy, with data of the rest of dysarthric speakers.

The intelligibility assessment and *Accuracy* prediction applications were first addressed as a regression problem, in which the objective was to predict the exact value of the intelligibility rating or of the *Accuracy* obtained by the speech recognizer. The results obtained by this approach are very informative, since the intelligibility and *Accuracy* scales are continuous, and any value between 0 and 100 is possible. However, there are different issues that can make continuous ratings misleading. First, a single intelligibility rating per speaker is not a completely fair choice, because the same speaker can utter different phrases with different levels of intelligibility. Second, intelligibility is a subjective measure, and a fixed intelligibility rating can not be considered as a fixed

gold standard, because the same utterance can have different levels of intelligibility for different people. Third, large errors can cause great confusion to the clinician, and make him/her believe that the result is much better or much worse than it really is.

To overcome those troubles (at least partially), both tasks were also addressed as a classification problem. An application that classifies intelligibility would just say if the utterance had *very low, low, mid* or *high* intelligibility, if 4 classes were possible. Even simpler, it could just classify utterances into *low* or *high* intelligibility, if only 2 classes were possible. The same for an ASR *Accuracy* classifier. To build such classifiers, first, the speakers were grouped into 4 or 2 classes according to their intelligibility or *Accuracy*, by splitting the interval [0,1] into equal parts. Then, the classification was made over the regression results by setting thresholds in 0.25, 0.5 and 0.75 for the 4-class problem and in 0.5 for the 2-class problem. Thus, if we obtained a regression value of 0.60, it belonged to class *mid* in the 4-class problem, and to class *high* in the 2-class problem. Note that in this example, if the true rating was 0.70, it would not have counted as an error in any of the two cases, whereas in the regression approach, the error would not be 0. The classification task was only accomplished for the case where we do not have data of the application user to train the system.

Intelligibility and ASR *Accuracy* assessments obtained with the regression system were measured in terms of:

- Pearson correlation (r), a metric that measures how linear the relationship between two variables is, with 1 meaning perfectly linear, 0 no linear relation, and -1 inverse linear relation. Given that our data were described with parametric models, and that we pursued a linear relationship between rated and predicted intelligibility, this type of correlation was appropriate for our problem. Its mathematical definition can be found in Onwuegbuzie et al. [2007].

- RMSE, a measure of the real difference between the predicted and rated values. The smallest this quantity, the closest our predictions to the subjective ratings. Its mathematical description can be found in Armstrong and Collopy [1992].

- Error rate at 12.5% ($\text{error\_rate}_{12.5\%}$), a metric proposed by us during the development of this work to overcome the subjectivity of intelligibility ratings made by the speech therapists [Martínez et al., 2013a]. It shows the percentage of utterances with a prediction error higher or lower than 0.125. The margin 12.5%

was selected to cover intervals of 25%, the same interval that a hard 4-class classification covers. The difference with the 4-class classification is that the intervals are not fixed and depend on the target value. It is defined as

$$error\_rate_{12.5\%} = \frac{C^+ + C^-}{N}, \tag{12.1}$$

$C^+ = \sum(predicted\_values > target\_value + 12.5\%)$,
$C^- = \sum(predicted\_values < target\_value - 12.5\%)$,
N = number of test utterances.

Classification was measured in terms of weighted average precision and weighted average recall. Precision measures the ratio of true positive outcomes and the sum of all outcomes classified as positive (positive means classified as the class under consideration). That is, among all outcomes classified as a given class, what percentage really belongs to that class. Meanwhile, recall refers to the ratio of true positive outcomes and all the outcomes of that class. This measures the percentage of outcomes of a class correctly classified. These two metrics were measured for each class individually. In order to obtain a global measure for the system, we averaged the result of all classes, weighting by the number of outcomes of each class. More information about these metrics can be found in Sokolova and Lapalme [2009].

All metrics above were measured in a per-word (or per-utterance) basis. That is, Pearson correlation was computed over the intelligibility ratings of all the evaluated words; in RMSE and error_rate$_{12.5\%}$, we measured the error of the prediction of each word with regard to its true label; and in classification, we counted the times that each word was correctly assigned to its class. Finally, a single metric for the whole system was obtained by averaging the results of all words. Additionally, we also obtained average results of the predictions for each speaker, since the ultimate goal of our applications was to obtain intelligibility and *Accuracy* assessments for each application user.

13

System Architecture

Contents

13.1 System Architecture

The two applications investigated in this part of the Thesis are based on the same system architecture. Nonetheless, we think that it could be easily adapted to more prediction tasks. The two applications we focused on have a great interest in the field of assistive technologies, and they could also have a great impact on society soon. The goal of the first one was to assess the intelligibility of dysarthric speakers after they uttered a set of words. The goal of the second one was to predict the *Accuracy* that a speech recognizer would obtain for those speakers after they uttered the same words. In our experiments, these words were the set of uncommon words of the UAspeech corpus. Naturally, these words were never seen during the application training process. These uncommon words (e.g., 'naturalization', 'moonshine', 'exploit') were selected from children's novels digitized by Project Gutenberg, using a greedy algorithm that maximized token counts of infrequent biphones. So, they were expected to generalize well and be useful to provide significant metrics of the speakers.

The major novelty of our proposal is the use of *i-Vectors* for the two tasks. From the viewpoint of a practitioner using assistive technology applications, the most interesting characteristic of *i-Vectors* is that they capture the intelligibility information of the utterance in a reduced set of measures. From the view point of a researcher building assistive technology applications, their most interesting characteristic is that they are fixed-length and low-dimension, and a whole utterance can be represented by a single *i-Vector*, independently of its duration. In our opinion, the most relevant feature of our scheme is that it performed an efficient compression of the acoustic parameters extracted from the speech, while keeping the most important information needed to make the assessments. The system architecture is depicted in Figure 13.1. First, PLP coefficients and energy, with their first and second derivatives, were extracted from the speech. Then, a UBM was trained on WSJ1, and used to compute sufficient statistics of each utterance. Next, the *i-Vector* extractor was trained with the sufficient statistics calculated for WSJ1. Finally, the *i-Vectors* obtained for the UAspeech database were used for training and evaluating the predictor. In the next subsections, we describe every component of the system in more detail.

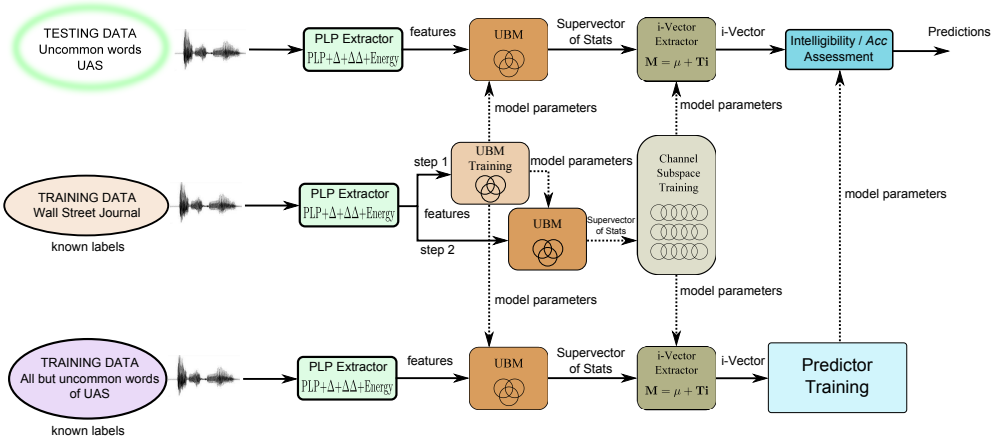


Figure 13.1: Architecture of the intelligibility assessment and *Accuracy* prediction systems - PLPs and supervectors of statistics of WSJ1 were used to train the UBM and the *i-Vector* extractor, respectively. *i-Vectors* of the training dataset of UAspeech were used to train the predictor. *i-Vectors* of the uncommon words of the UAspeech database were used for evaluating the system. There were 300 uncommon words per dysarthric speaker, and each word was represented by one *i-Vector*.

13.1.1 Acoustic Features

Each audio file was parametrized into 12 PLP features plus energy, with derivatives and accelerations, to obtain a 39 dimension vector every 10 ms, in 25 ms length windows. These features use 3 concepts of the pshychophysics of hearing: the critical band spectral resolution, the equal-loudness curve, and the intensity-loudness power law. Previous investigations showed that there is information about intelligibility in the short-term spectral content [Hosom et al., 2003]. There was no a priori theoretical reason why PLP should work better than others commonly used features like MFCC, however there are works on speech intelligibility with pathological voices, where PLPs offered some advantages over MFCCs [Bocklet et al., 2009]. A reason could be that PLPs follow better the peaks of the spectrum, thanks to the linear prediction analysis they perform, what is known as the "peak-hugging" property of linear prediction. Then, a better model of the vocal tract transfer function is obtained [Makhoul, 1975].

13.1.2 GMM and Sufficient Statistics

GMM and sufficient statistics were already presented in the LID part of the Thesis, and a detailed explanation can be found in Appendix A. Here, we will give details about

the training strategy, which is similar to the one used for LID. Thus, a full-covariance GMM was trained to be used as UBM. We used a splitting strategy with $H = 2$ as indicated in Section A.4.1, where we started an iterative process with a single Gaussian trained with 20 iterations of the EM algorithm, then the Gaussian was split in two, next the resulting 2-component GMM was trained by running 20 iterations of the EM algorithm, and so on, until we reached the 2048-component GMM. The UBM was trained on WSJ1. The number of Gaussian components was determined during the experimentation.

The computation of sufficient statistics over the UBM is detailed in eqs. (2.37-2.42). These statistics were the input to the JFA front-end, which is the block in charge of computing *i-Vectors*.

13.1.3 Factor Analysis Front-End: *i-Vector* Extractor

As for the UBM, the *i-Vector* extractor was trained similarly to the LID case. A full explanation of the training process is given in Section sec:ivectors. Basically, we calculated matrix subspace $\mathbf{T}$ by running 20 iterations of the EM algorithm, alternating between an ML step and an MD step, as indicated in Section 2.11. The training was made using WSJ1. Means and covariances were not updated, and they were fixed to the UBM ones. Note that the covariances of the UBM were full, but for the JFA model, we only kept the diagonal part. Then, *i-Vectors* were extracted using eq. (A.103). *i-Vector* dimensionality was determined during the experimentation.

13.1.4 Predictor

The predictor was the block in charge of making assessments from *i-Vectors*. It can be seen as a block that transforms the *i-Vector* associated to a given utterance into an intelligibility rating or an *Accuracy* prediction. Note that the labels used in training were the key information to make our system work as an intelligibility assessment system or as an *Accuracy* predictor. Thus, the predictor was the block responsible to retain the required intelligibility or *Accuracy* information from *i-Vectors*. We investigated two different predictors, linear predictor [Bishop, 2006] and ν-support vector regression (ν-SVR) predictor [Chang and Lin, 2002; Smola and Schölkopf, 2004].

For training the predictor, we used 465 words per speaker, including the 155 different words repeated three times in the UAspeech database. For evaluating the applications,

we used 300 different words per speaker, the 100 uncommon words of the 3 blocks of the UAspeech database. Note that for each word an *i-Vector* was extracted and passed directly to the regressor, either for training, or for testing, as appropriate.

In the next paragraphs, the two predictors are described.

13.1.4.1 Linear Prediction

In a linear predictor, the sum of squares error function

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \mathbf{w}^\intercal \cdot \mathbf{i}_n)^2, \tag{13.1}$$

is minimized using ML to obtain the vector of weights, $\mathbf{w}$, that enables the linear transform of *i-Vectors* into intelligibility ratings [Bishop, 2006]. N is the number of training utterances, y_n is the labeled intelligibility rating for utterance n, and $\mathbf{i}_n$ is the *i-Vector* belonging to utterance n.

Once the weights are trained, new assessments are obtained as

$$\hat{y} = \mathbf{w}^\intercal \cdot \mathbf{i}. \tag{13.2}$$

13.1.4.2 Support Vector Regression

The basic idea of SVR is that only a subset of vectors which are not further than a given margin from the regression curve are used for training [Vapnik, 1995; Smola and Schölkopf, 2004]. The approximating function is

$$f(\mathbf{i}_j) = \sum_{n=1}^{N}\hat{\alpha}_n\phi(\mathbf{i}_n)^T\phi(\mathbf{i}_j) + b, \tag{13.3}$$

where N is the number of files in the training dataset, $\mathbf{i}_j$ is the evaluated *i-Vector*, $\hat{\alpha}_n$ is a scalar equal to the difference between the Lagrange multipliers associated to data point n, b is a bias, and ϕ is the kernel, in our case a radial basis function, which allows to model nonlinearities in *i-Vectors*. LIBSVM software was used to train the SVR predictor and to evaluate test utterances.

Two different SVR approaches were evaluated. The first, ϵ-SVR, where the goal is to build a function, f, that has at most ϵ deviation from the target points, y_j, for all the training data, and at the same time is as flat as possible [Smola and Schölkopf, 2004]. In other words, we do not care about errors as long as they are less than ϵ, but

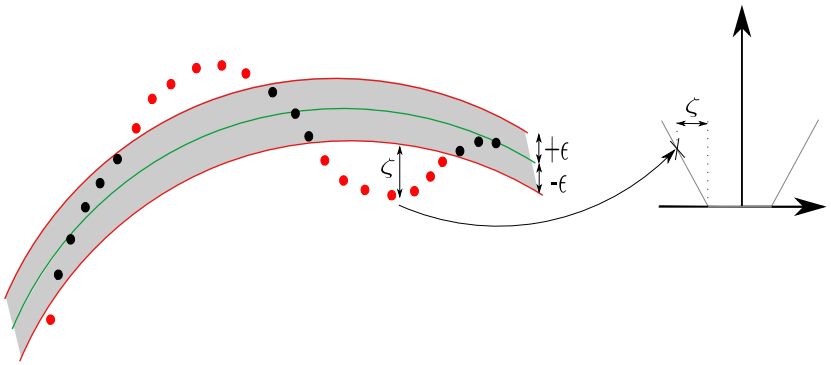


Figure 13.2: Support Vector Regression - In the left, we can see an example of a regression curve, where the ϵ-insensitive zone is shaded and the support vectors are the red points outside this region. In the right, the cost function is plotted. Only points outside the ϵ-insensitive zone contribute to the cost.

we care about overfitting by introducing a regularization term over the parameters. In Figure 13.2, we depict the basic idea of SVR. Only points outside the shaded region add a cost to the optimization function. Thus, the subtraction of their associated Lagrange multipliers, $\hat{\alpha}_n$, is nonzero, and hence, they contribute to the prediction of $f(\mathbf{i}_j)$ in eq. (13.3). On the other hand, the subtraction of the Lagrange multipliers of the points inside the shaded region vanishes, i.e. $\hat{\alpha}_n = 0$. Thus, the larger ϵ, the fewer points contribute to the prediction because fewer support vectors are selected. In the optimization procedure detailed in Smola and Schölkopf [2004], there is a second parameter, in addition to ϵ (without having into account the kernel parameters), relevant for the training of ϵ-SVR, known as C. The parameter C determines the trade-off between the complexity (flatness) and the amount up to which deviations larger than ϵ are tolerated. If C is too large, the objective will be to minimize the optimization function, without regard to the complexity of the resulting function, and if it is too low, the objective will be to obtain a flat regression curve, without regard to the final value obtained in the optimization function.

The second approach is known as ν-SVR [Schölkopf et al., 1998b; Chang and Lin, 2002]. It is a modification of ϵ-SVR that automatically minimizes ϵ, thus adjusting the accuracy level to the data at hand. The parameter ϵ can be useful if the desired accuracy of the approximation can be specified beforehand, but in some cases we may want to be as accurate as possible, without making any prior accuracy commitment [Schölkopf et al., 1998a]. In Schölkopf et al. [1998b], the parameter ν is introduced.

This replaces ϵ and lets one control the number of support vectors and training errors. In that work, the authors state that ν is an upper bound on the fraction of errors and a lower bound on the fraction of support vectors. The parameter C has the same meaning as before. For a detail description of the training, see Schölkopf et al. [1998b,a]; Chang and Lin [2002]; Smola and Schölkopf [2004].

14

Results

Contents

In this chapter, we present the results on intelligibility assessment and on ASR*Accuracy* prediction. For the first task, we investigated the use of linear prediction and SVR to make assessments from *i-Vectors*. Better assessments were observed with the second, therefore, we only used SVR for the ASR *Accuracy* prediction task. Also, we compared the regressor systems predicting intelligibility and *Accuracy* in a continuous scale, with classification systems that group or classify the speakers into one of a pre-defined number of categories, each covering an interval of intelligibility assessment ratings or ASR*Accuracy* values.

14.1 Intelligibility Assessment

A computer application to automatically obtain intelligibility measures would bring several benefits to clinicians, such as objectivity and replicability of results. It would allow that speech therapists from different places apply the same criteria to evaluate intelligibility. In addition, clinicians can get used to the speech of their patients and become more familiar with their manner of pronunciation, what can provoke that they increasingly consider their speech as more intelligible. A computer application would also avoid this problem. Intelligibility assessment is an important part in the monitoring of a patient progress, and computers can contribute to perform this task always with the same criteria and with no human subjectivity. In our experiments, we compared an application that was trained with a dataset including pre-recorded speech of the evaluated user, with an application that was trained with a dataset without prior information about the user.

14.1.1 Linear Prediction

In Table 14.1, we present results for an experiment using linear prediction, where data of the evaluated speaker was available for training. It can be seen that the correlation between our predictions based on *i-Vectors* and the subjective intelligibility ratings given in UASpeech increased as we increased the number of Gaussians in the GMM and the *i-Vector* dimensionality. The best results were obtained for 1024 Gaussians and 400 and 600 dimensions. Probably, if our training dataset had been larger, we could have trained larger models, but note that the complexity and processing time would also increase.

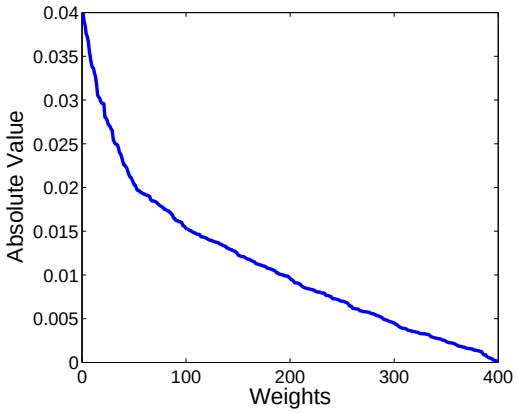


Figure 14.1: Absolute value of linear prediction weights - Sorted in descending order, using 400 dimension *i-Vectors*, extracted with 1024 Gaussian components.

Nr. Gaussians	512	1024							2048
i-Vector Dim	400	5	10	50	100	200	400	600	400
PC	0.87	0.76	0.81	0.85	0.86	0.87	**0.88**	**0.88**	**0.88**
RMSE	0.17	0.22	0.20	0.18	0.17	0.17	**0.16**	**0.16**	**0.16**

Table 14.1: Results on intelligibility assessment with linear prediction - r and RMSE are given for different number of Gaussian components in the UBM and *i-Vector* dimension.

It is remarkable that even with a low *i-Vector* dimensionality like 10, we could obtain a correlation r over 0.8, which indicated that most of the intelligibility information was contained in a few dimensions. Inspection of the ranked absolute value of the weights, $|\mathbf{w}|$, in Figure 14.1, for 400-dimension *i-Vectors* and a GMM with 1024 components reveals that after the first 50 dimensions the weight value had been halved, and the weight 350 was about a tenth of the first. This explains why adding more than 400 dimensions did not result in significant improvements.

Remember that, as we said in Section 12.1.1, we only used microphone 6 in our experiments. We checked if we could obtain some gains by using the data recorded with the rest of microphones. Training the predictor with all available data in UASpeech, and testing only data recorded with microphone 6, $r = 0.89$ and $RMSE = 0.15$ were obtained for the system with 1024 Gaussian components and 400-dimension *i-Vectors*. In this experiment, we also used data of the evaluated speaker during training. Since

Number of Gaussians	512			1024		
Dimension of *i-Vectors*	50	100	400	50	100	400
Pearson Correlation	0.47	0.51	0.59	0.28	0.40	0.49
RMSE	0.30	0.29	0.27	0.33	0.31	0.30

Table 14.2: Results on intelligibility assessment using only UASpeech from training the whole system - r and RMSE on intelligibility assessment with linear prediction for different number of Gaussian components in the UBM and *i-Vector* dimension. WSJ1 was not used at all.

it was not a huge improvement, we kept using only microphone 6 for training in the rest of our experiments.

One possible weakness of our system was that we only used WSJ1 for training the UBM and the FA front-end, a database very different to UASpeech. One might think that this mismatch could create unpredictable *i-Vectors* for the dysarthric speech. In Table 14.2, we reflect the results of an experiment where only UASpeech was used in the training process. That is, the UBM and the *i-Vector* extractor were also trained on UASpeech. We observed a dramatic fall of performance, indicating that having a large amount of data for training ML models helped, even if these data were not recorded from dysarthric speakers. Note that in this experiment, we also used data of the evaluated speaker during training.

14.1.2 Support Vector Regression

14.1.2.1 Intelligibility Assessment by Regression

In the first experiment of this section, we compared the two approaches of SVR presented in Section 13.1.4.2. After the results obtained in previous section with linear prediction, we adopted a system configuration with 1024 Gaussian components and 400-dimension *i-Vectors* for the rest of our experiments. In Figures 14.2 and 14.3, r is plotted as a function of C, the regularization parameter of SVR, for different values of ϵ and ν. In this experiment, data of the evaluated speaker were available for training. The optimal value of C was 1 in both cases, the optimal ϵ was 0.01, and the optimal ν was 1.

The best performance was obtained for ν-SVR with $C = 1$ and $\nu = 1$, resulting in $r = 0.91$ and RMSE $= 0.14$, but in general, the ν-SVR system was stable in the

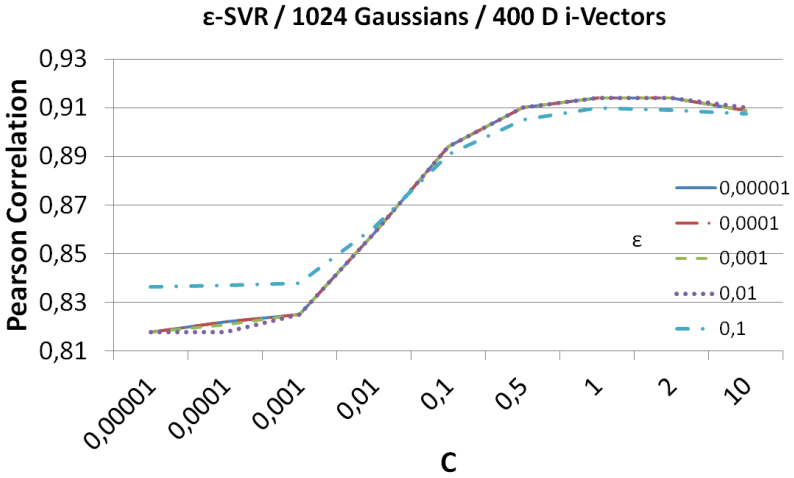


Figure 14.2: Results on intelligibility assessment with ϵ-SVR - r obtained with a system using 1024 Gaussian components and 400-dimension *i-Vectors*.

	User Data in Train	User Data not in Train
r	0.91	0.74
RMSE	0.14	0.23
error_rate$_{12.5\%}$	0.33	0.61

Table 14.3: Results on intelligibility assessment with ν-SVR using and not using data of the evaluated speaker for training - r, RMSE, and error_rate$_{12.5\%}$ when we had user data available for training (middle column) and when we did not have user data available for training (right column). System using 1024 Gaussian components, 400-dimension *i-Vectors* and ν-SVR with $C = 1$ and $\nu = 1$.

range of values of $C = 0.5 - 10$ and $\nu = 0.1 - 1$, and the results were not very different within these intervals. Thus, we used ν-SVR with $C = 1$ and $\nu = 1$ for the rest of the experiments. For comparison, $r = 0.94$ and RMSE $= 0.19$ were obtained in Falk et al. [2012] on data of ten spastic speakers of the UASpeech database, with a system using a set of six features representing atypical vocal source excitation, temporal dynamics, and prosody, and also data of the evaluated speaker was available during training. Note that in our system we used PLP features with energy and derivatives.

Then, our system for the rest of the experiments was configured with 1024 Gaussian components, 400-dimension *i-Vectors*, and ν-SVR with $C = 1$ and $\nu = 1$. In the next experiment, reflected in Table 14.3, we can compare the performance of the system when

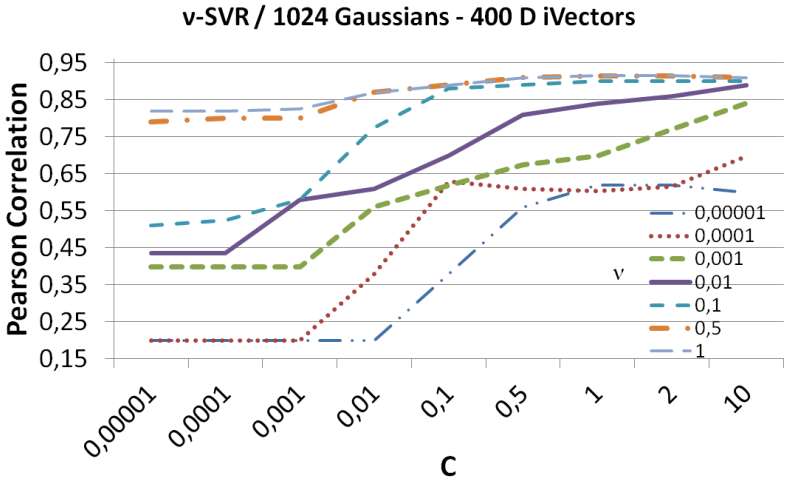


Figure 14.3: Results on intelligibility assessment with ν-SVR - r obtained with a system using 1024 Gaussian components and 400-dimension *i-Vectors*.

we had and we did not have available data of the application user during the training phase. These results clearly show that it was very helpful to count with data of the application user at the time of training the system. As we can see, the reduction in correlation and increase in RMSE and $\text{error\_rate}_{12.5\%}$ were high in the more challenging scenario without data of the application user during training. Two possible causes were responsible for this behavior, and both arose because of the limitation of the UASpeech database. First, the system was not able to predict with the same accuracy intelligibility ratings not seen in training. Given that we only had 13 different intelligibility labels in the training dataset, if we removed one, the system was not able to interpolate with the rest properly. Second, in the case where the application user was included for training, the system was learning not only intelligibility information, but also the speaker identity. We had very few speakers, and every speaker was uniquely associated to a single label, therefore, each label identified uniquely to that speaker (except for speakers F04 and M11, and speakers M10 and M08, who shared intelligibility rating).

In order to analyze the results of each speaker individually, the mean and standard deviation of each speaker for the case where we used data of that speaker for training are plotted in Figure 14.4, and the same metrics for the case where we did not use data of that speaker for training are plotted in Figure 14.5. We clearly see that the second curve deviated further from the x=y line (dash-dot curve). This was more dramatic

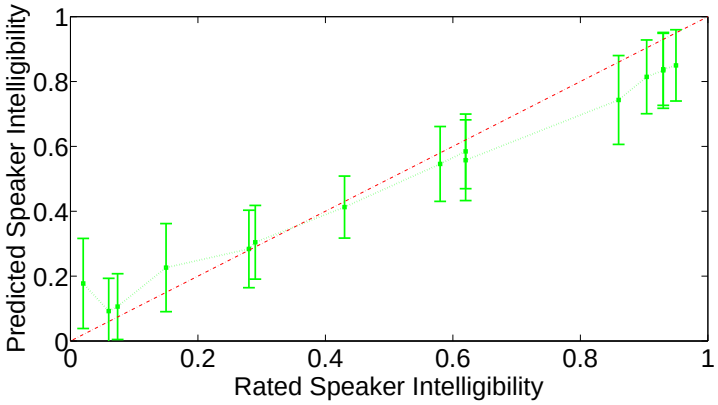


Figure 14.4: Individual results per speaker when there were user data available for training - Mean and standard deviation of automatic intelligibility assessments for each speaker (straight green), and x=y line (dash-dot red).

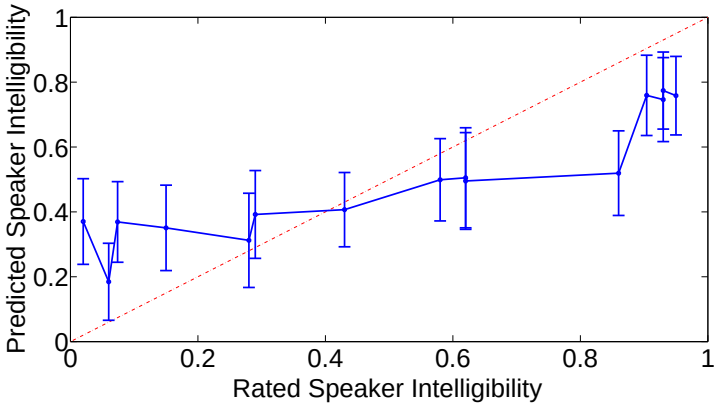


Figure 14.5: Individual results per speaker when there were no user data available for training - Mean and standard deviation of automatic intelligibility assessments for each speaker (straight blue), and x=y line (dash-dot red).

for speakers with very low intelligibility. One could think that this was due to having a UBM and an *i-Vector* extractor trained without any dysarthric speaker. Then, the less intelligible speakers should have had the least accurate predictions. However, the fact that mid intelligible speakers were better modeled than high intelligible speakers contradicted this hypothesis. In the future, we would like to try a mid-solution between training the UBM only with WSJ1 and only with UASpeech, for example a MAP adaptation of the WSJ1 UBM with dysarthric speakers of UASpeech.

We must also take into account the data scarcity issue, and when we did not include

the evaluated speaker for training, we lost a non-negligible part of the training data. The consequences of this were even more important for the least and most intelligible speakers, for whom the system could not interpolate with any other speakers to learn their associated intelligibility. Therefore, including the test speaker in training can be thought as an ideal scenario, where all the speakers were perfectly represented in the training dataset. This yielded optimal results, as observed.

In spite of all these problems, the correlation obtained for the case where we did not have data of the application user for training was about 0.74, which can still be interpreted as high correlation. One could think that the comparison of Table 14.3 is unfair because in the case where we had data of the evaluated speaker for training, we trained the predictor with more data compared to the case where there was no information about the evaluated speaker in the training dataset. To investigate this, we carried out a control experiment including information about the application user but where the amount of data was reduced to match that available for the case with no data of the application user in the training part. The results showed no difference, and $r = 0.91$, and $RMSE = 0.14$, were obtained in this last scenario.

One interesting point would be to see if there were some words better predicted than others, and to analyze if the best predictions came from particular word patterns. We measured the difference between the rated and automatically predicted intelligibility for each of the 300 tested words per speaker, and most of these differences were in the range 0.14-0.24. The worst predicted words, with a mean difference computed over all speakers higher than 0.25, were *behavior, employment, scissors, aloft, booth, buffoon, fishing, swoon*, and *ahead*. The best predicted words, with a mean difference over all speakers under 0.13, were *Pennsylvania, advantageous, bloodshed*, and *designate*. We did not find any phonetic cue indicating that some patterns were better rated than others. However, it seems that shorter words were more difficult to predict, although there was no strong evidence about it.

14.1.2.2 Intelligibility Assessment by Classification

The problems associated with regression caused by having each speaker associated to a unique intelligibility rating should be alleviated in a classification problem, because several speakers with different ratings are grouped into the same class. Also, the interpolation of extreme intelligibility ratings should not be as crucial, and it will be

Label (↓)\Decision (→)	Very Low	Low	Mid	High
Very Low	31.76	57.69	10.19	0.37
Low	19.55	61.65	18.67	0.13
Mid	4.12	46.88	44.38	4.62
High	0.23	10.50	44.86	44.41

Table 14.4: Confusion matrix for the 4-class intelligibility assessment classification problem - Confusion matrix in % of words.

Label (↓)\Decision (→)	Low	High
Low	85.94	14.06
High	25.89	74.11

Table 14.5: Confusion matrix for the 2-class intelligibility assessment classification problem - Confusion matrix in % of words.

sufficient if the system learns that the *i-Vector* associated to an utterance is close to others of the same class. The classification problem was conducted only for the case where there was no data of the application user for training.

The results of the 4-class classification problem are given in Table 14.4 in the form of a confusion matrix. Encouragingly, confusions were mainly made with neighbor classes. This fact assures that there was no overtraining of any class. The problem was difficult though, especially for the *very low* class, for which only 31.76% of the files were correctly classified, and almost 60% of the files were confused with the *low* class. The average weighted precision was 0.60 and the average weighted recall was 0.44.

Given that there were still many confusions with neighbor classes, a simpler and more reliable solution for real applications would be to just discriminate between *high* and *low* intelligibility. As we can see in Table 14.5, the results of this scenario were much better, but also note that the classes were twice as wide as for the 4-class problem, and hence, the information given by the system was not as precise as the one given by the 4-class classification system. Both weighted precision and recall were 0.80.

14.2 ASR Word *Accuracy* Rate Assessment

The interest of this application lies in obtaining confidence measures of ASR systems that guarantee a successful usage. ASR has the potential to be a very important human-

Nr	Spk Label	Speech Intelligibility (%)	*Accuracy* in [Christensen et al., 2012]	Mean *Accuracy* Assessment User Data in Train	Mean *Accuracy* Assessment User Data Not in Train
1	M04	very low (2%)	8.30%	25.61%	49.41%
2	F03	very low (6%)	23.00%	23.37%	30.53%
3	M12	very low (7.4 %)	11.70%	15.18%	44.81%
4	M01	very low (15%)	29.80%	34.08%	42.15%
5	M07	low (28%)	66.90%	55.12%	35.43%
6	F02	low (29%)	36.90%	37.99%	43.79%
7	M16	low (43%)	49.30%	49.61%	48.82%
8	M05	mid (58%)	53.40%	53.12%	51.95%
9	F04	mid (62%)	65.60%	61.71%	53.47%
10	M11	mid (62%)	53.00%	52.07%	54.30%
11	M09	high (86%)	81.50%	72.39%	54.65%
12	M14	high (90.4%)	74.90%	72.39%	71.59%
13	M10	high (93%)	86.20%	78.19%	69.50%
14	M08	high (93%)	81.80%	76.32%	70.95%
15	F05	high (95%)	89.60%	80.77%	68.97%
Total			54.10 %	52.69%	52.69%

Table 14.6: Labels and results per speaker for the ASR *Accuracy* prediction system - In the first two columns, we have the speaker identification number and label. In the third column, we have the speech intelligibility ratings given in the UASpeech database. In the fourth column, we have the *Accuracy* labels obtained in Christensen et al. [2012]. In the fifth column, we have the *Accuracy* predictions obtained by our system when there was user data available for training. In the sixth column, we have the *Accuracy* predictions obtained by our system when there was not user data available for training. In the last row, we have the averages obtained for the fourth, fifth, and sixth columns.

computer interaction mechanism for people with limited range of movements, as it happens with many people affected by dysarthria. Again, we compared the cases where data of the application user were and were not available in advance to train the system. The only change with respect to the intelligibility assessment experiment was the use of different labels to train the system. Instead of the intelligibility ratings given by the UASpeech database, we used the *Accuracy* results obtained by the reference speech recognizer, which was the *mapSI2* system presented in Christensen et al. [2012]. That was the best-performing system among 11 systems presented in that paper evaluating the same dysarthric speakers as we did. In that work, the authors used data of the evaluated speakers to build the recognizer, and made a MAP adaptation to the final user.

	User Data in Train	User Data not in Train
r	0.89	0.55
RMSE	0.12	0.22
error_rate$_{12.5\%}$	0.26	0.56

Table 14.7: Results on ASR *Accuracy* prediction with ν-SVR using and not using data of the evaluated speaker for training - r, RMSE, and error_rate$_{12.5\%}$ for the *Accuracy* prediction system when we had user data available for training (middle column) and we did not have user data available for training (right column).

14.2.1 Word *Accuracy* Rate Assessment by Regression

In general, this task was more complicated than intelligibility assessment, especially for the *very low* intelligible speakers, for whom better *Accuracy* than the real ones were obtained. Observe in Table 14.6 that there were only three speakers with *Accuracy* below 25% (M04, F03, and M12), therefore it was very hard for the regressor to learn the lower part of the *Accuracy* scale. Note that in this case, the labels were the result of the filtering process committed by the speech recognizer, which was not error-free, so the labels might contain noise introduced by this system. Despite this problem, when data of the application user were used for training, a correlation about 0.90 was obtained, as it can be seen in the middle column of Table 14.7. When no data of the application user were used for training, the drop in the correlation was dramatic, and the RMSE and error_rate$_{12.5\%}$ increased significantly, as observed in the right column of the same table. However, RMSE and error_rate$_{12.5\%}$ were smaller than in the intelligibility assessment task. This result could be misleading because, if we observe the individual results for each speaker in Table 14.6, although the *very low* and *very high* predictions were less accurate, we had less speakers with such labels, and this caused a smaller value of RMSE and error_rate$_{12.5\%}$. Consequently, the behavior of the intelligibility assessment system was preferred, because the system did not predict some intervals better or worse than others. In Table 14.6, the results of the *Accuracy* predictions for each speaker are shown for the cases where the user data were and were not present in the training dataset.

Label (↓)\Decision (→)	Very Low	Low	Mid	High
Very Low	9.95	64.18	25.62	0.25
Low	3.35	64.68	31.97	0.00
Mid	3.89	37.05	49.03	0.10
High	0.00	8.91	68.82	22.27

Table 14.8: Confusion matrix for the 4-class ASR *Accuracy* classification problem - Confusion matrix in % of words.

Label (↓)\Decision (→)	Low	High
Low	71.07	28.93
High	26.81	73.19

Table 14.9: Confusion matrix for the 2-class ASR *Accuracy* classification problem - Confusion matrix in % of words.

14.2.2 Word *Accuracy* Rate Assessment by Classification

The same problem as for the regression approach was observed in *Accuracy* classification. The *very low* class was not well modeled, and only 10% of the utterances were well classified in the 4-class classification problem. In general, the system was biased to predict the *low-mid* interval, as can be seen by the confusions in Table 14.8. For example, for the *very low* class, there were more words classified as *mid* than as *very low*. An average weighted precision of 0.45 and a weighted recall of 0.37 were obtained. Remember that the classification problem was conducted only for the case where data of the application user was not included for training.

A more reliable solution was the system that classifies just between *low* and *high*. In Table 14.9, we have the confusion matrix of this 2-class classification problem. In this case, both average weighted precision and recall were 0.72. These results confirm that ASR *Accuracy* prediction was more challenging than intelligibility assessment. In our opinion, one of the main reasons was that the labels were noisier due to the filtering process that the speech recognizer made.

15

Analysis of *i-Vectors*

Contents

In the final chapter of this part of the Thesis, we study the potential that *i-Vectors* have in future applications, and we analyze if the proposed method is better than other techniques previously used in the literature. Specifically, we studied the goodness of *i-Vectors* in the sense of proximity between those obtained for the same and different classes, and we tried to understand why we obtained the reported results. Then, we compared the results with two systems without *i-Vectors*. In the first, we extracted the mean of the PLPs of each utterance, and assessed intelligibility directly with the resulting set of coefficients. In the second, we used the supervectors of 1st order sufficient statistics calculated with the UBM directly to make the assessments, similarly to Bocklet et al. [2012].

15.1 Goodness of *i-Vectors*

In this section, we analyzed the goodness of *i-Vectors* to do intelligibility assessments. By goodness we mean similarity between *i-Vectors* extracted for all intelligibility ratings. The basic idea was to see if *i-Vectors* of the same rating were similar, and how similar they were, and if *i-Vectors* of different ratings differed, and how much they differed. The metric used to measure this similarity was CSS as defined in eq. (5.14), hence higher CSS means more similar *i-Vectors*.

First, we computed CSS between all possible pairs of *i-Vectors* extracted from the UASpeech database (including training and test datasets). That is, we measured the similarity between all the words included in the database. Then, for every possible intelligibility pair, we averaged all the CSSs calculated with all *i-Vector* pairs corresponding to speakers with those two intelligibility ratings. The result can be seen in Figure 15.1. We obtained a matrix where the average CSS among *i-Vectors* belonging to the intelligibility rating i and the intelligibility rating j is shown in row ith and column jth. Therefore, in the main diagonal, it is plotted the CSS among *i-Vectors* belonging to the same intelligibility rating. Note that there were 15 speakers and 13 ratings, because there were 2 pairs of speakers sharing the same rating (F04 and M11, and M10 and M08). The matrix is upper triangular to avoid replicating the information twice. The lower part would be the upper part transposed.

It can be observed that the similarity between high intelligibility pairs was higher (higher CSS) than that of the low intelligibility pairs, that is, the left upper part of Fig-

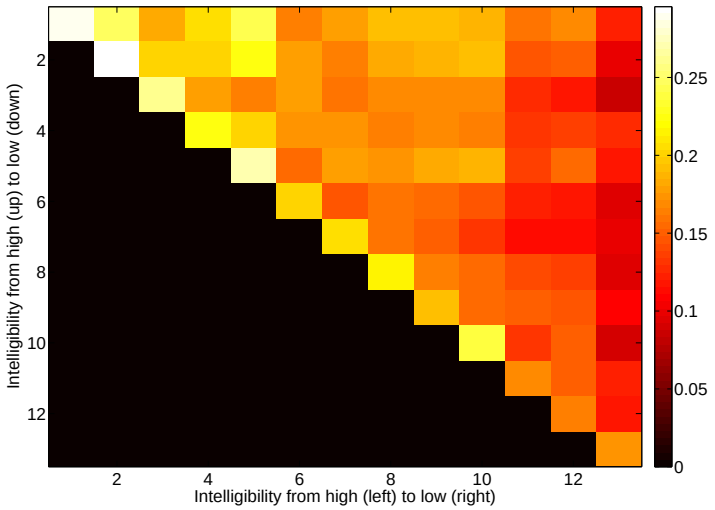


Figure 15.1: Matrix of average CSS calculated from *i-Vectors* among all possible intelligibility pairs - Each cell of the matrix shows the average CSS of the intelligibility rating pair indicated by the corresponding row and column. The whiter the cell, the closer *i-Vectors* are.

ure 15.1 is lighter than the right lower part, which is darker. Also, *i-Vectors* belonging to high intelligibility ratings were not similar to those of low intelligibility ratings, and the right upper part is dark red. In addition, it can be seen that the main diagonal decreases progressively from white to orange. This means that *i-Vectors* belonging to high intelligibility ratings were more similar among themselves than *i-Vectors* belonging to low intelligibility ratings. This is a normal behavior because very severe dysarthric speakers can produce very different sounds even when they want to say the same word or sentence, and it shows that the sound variability of very low intelligible dysarthric speakers is higher than that of more intelligible dysarthric speakers. Therefore, the progressive decrease of the CSS indicated that *i-Vectors* variability increased gradually as we passed from high to low intelligibility. Ideally, we would like to have a main diagonal as white as possible, indicating that *i-Vectors* belonging to the same intelligibility rating were very similar. In conclusion, *i-Vectors* behaved as expected, and they have the potential to be good features for intelligibility assessment. The results will improve if we are able to obtain *i-Vectors* more similar when they belong to the same intelligibility rating, and less similar when they belong to different intelligibility

	User Data in Train	User Data not in Train
r	0.83	0.28
RMSE	0.19	0.34
error_rate$_{12.5\%}$	0.45	0.77

Table 15.1: Results for the intelligibility assessment system based on PLP means - Results in terms of r, RMSE, and error_rate$_{12.5\%}$, when we had user data available for training (mid column) and when we did not have user data available for training (right column).

ratings. In short, the gradual variation from high to low intelligibility arose as a natural consequence of very low intelligible speakers being less consistent in their realizations.

15.2 Intelligibility Assessment with PLP Means

The simplest approach to assess intelligibility we could think of was to compute the mean of the PLPs of each file and assess intelligibility with the resulting set of coefficients. In this way, every word was represented by the mean of the PLPs, and that mean was the input to the regressor. This allowed us to check if *i-Vectors* were really keeping the important information while compressing the acoustic parameters. As we can see in Table 15.1, for the case where user data were available for training, the results with this method fell with respect to the *i-Vector* system, but they were still good. However, for the case where we did not include data of the user data for training, the decrease in performance was dramatic. This confirmed the success of *i-Vectors*.

15.3 Intelligibility Assessment with Supervectors of 1st Order Statistics

Another interesting comparison with *i-Vectors* was to assess intelligibility directly with the supervectors of 1st order statistics extracted with the UBM, as defined by eqs. (2.38) and (2.41). In this approach, the *i-Vector* extractor block was removed from our scheme, but we did not get the same big compression rates. The results are in Table 15.2 for a UBM with 1024 components. As we can see, *i-Vectors* allowed increasing the system performance, and even most important, they allowed a big reduction in computational time and simplicity. The dimension of supervectors was very high ($1024 \cdot$

	User Data in Train	User Data not in Train
r	0.90	0.71
RMSE	0.15	0.24
$\text{error\_rate}_{12.5\%}$	0.35	0.60

Table 15.2: Results for the intelligibility assessment system based on first order statistic supervectors - Results in terms of r, RMSE, and $\text{error\_rate}_{12.5\%}$, when we had user data available for training (mid column) and when we did not have user data available for training (right column). Supervectors of statistics computed with a 1024-component UBM.

$39 = 39936$), and the regressor had more problems to learn the important information. In short, *i-Vectors* removed noisy information from supervectors and kept the most important one, what turned into better results.

One interesting observation is that the system with PLP means as input behaved well when there was user data available for training but when there was not, the results dropped dramatically. This might be an indication that when data of the evaluated user was included in training, the system learned speaker information that the PLP means efficiently collected, instead of intelligibility information. However, the *i-Vector* and supervector of statistics systems did not suffer such a dramatic drop, what might indicate that these two approaches really learned intelligibility information.

16

Conclusions

Working with dysarthric voices is very challenging due to the different types of dysarthrias that can be found and to the great variability in the pronunciations of people with this affection. Dysarthric speakers normally have a lack of control of the speech production mechanism that derives in irregular voice patterns with sudden and unexpected voice sounds or breaks. Two interesting assistive applications for people with dysarthric speech based on *i-Vectors* were proposed in this part of the Thesis: intelligibility assessment and ASR *Accuracy* prediction. After studying the state of the art in these fields, we can realize that the investigations are probably in an early stage. One reason that can have reduced the speed of progress compared to other fields in speech technology is the lack of available databases in the community. Data acquisition is an important and delicate issue that requires coordination between research groups and clinical institutions. In the last years, some groups have made important efforts to release databases open to everybody, and the numbers of works has growth considerably. We have found interesting studies in the literature proposing different approaches, but we can not state that there is a strong solution to the problem yet. In this Thesis, we proposed a solution for intelligibility assessment and ASR accuracy prediction, which opens a new research direction. In addition, we reported promising results, which were analyzed in depth.

The first application had the goal of making automatic intelligibility assessments of dysarthric speech. In our application, the intelligibility rating of the person was made from a set of words not seen during the training process. The importance of making automatic intelligibility assessment for clinicians monitoring the progress of

their patients is huge. It would allow making objective assessments that are easily replicated. Furthermore, practitioners involuntarily get used to the speech of their patients, and such a tool would avoid this problem. Unlike humans, our application is free of the negative effect that in this case this continuous learning (that happens every time that the patient speaks) has.

Speech recognizers have a great potential to be used by disabled people with a limited range of movements, as is the case for many dysarthric speakers. They can serve as a human-computer interface when other common devices such as keyboards or mice can not be used. One important problem for the extension of voice interfaces for dysarthric speakers is that they cannot use standard applications. Dysarthric speech contains a high degree of variability, and such complicated inputs are very difficult to handle by current ASR systems. Probably, the cheapest solution for this would be to convert the dysarthric speech into a more intelligible one, so that current ASR systems could understand it [Hosom et al., 2003; Kain et al., 2007]. However, a technologically easier option is to train specific systems for people with this type of disability. At the present time, some progress has been made in this direction [Hamidi et al., 2010; Christensen et al., 2012]. Nevertheless, we have to bear in mind that, in the short term, the extension of these applications will be limited to the individuals for whom it was trained, and that they require a recording process, which is often hard and very exhausting for people with disabilities. Our second application was designed to predict the *Accuracy* that a speech recognizer will obtain for dysarthric speakers. If predictions of how the speech recognizer will perform for each user could be known beforehand, health costs and abandoned usage rates would be diminished. As for the intelligibility assessment case, the user only had to utter a set of words to obtain a prediction with our application.

A very relevant conclusion of our work is that it is very helpful to include pre-collected data of the application user for training our applications. In a clinical environment, it is common to know who will use the application in advance. However, this is an ideal scenario that is not always possible. Unfortunately, when data of the application user is not included for training, the performance of our systems dropped significantly. The implications of these results are of great impact among researchers and practitioners. As we said above, these results show a handicap for a global exten-

sion of voice interfaces among dysarthric people in the short term, and more effort is needed to improve results in such situations.

For the case of intelligibility assessment, the correlation between intelligibility perceptual ratings and the automatic intelligibility assessments made by our application fell from about 0.90, when we had user data available in the training dataset, down to about 0.74, when we did not, whereas the RMSE increased from 0.14 to 0.23. The assessments were worse for the *very low* and *high* intelligible speakers than for the *mid* intelligible ones. One important reason was data scarcity, especially for the speakers with extreme intelligibility ratings, because the regressor had no information about lower or higher intelligibility ratings to interpolate with.

For the case of *Accuracy* predictions, the correlation between the true scores obtained with the speech recognizer and the automatically predicted *Accuracy* values made by our application fell from about 0.89, when we had user data available in the training dataset, down to 0.55, when we did not, whereas the RMSE increased from about 0.12 to 0.22. In this case, worse results were also obtained for the speakers with *very low* and *high* intelligibility, but especially for the *very low*, because there were only three speakers in the database labeled with *very low Accuracy*, and the system did not have enough information to model them properly. Moreover, for this application the ground truth labels came from the evaluation of a speech recognizer, which was not error-free. Hence it is likely that these labels were not completely accurate.

i-Vectors were used as a method to compress the acoustic parametrization of the signal. They capture many aspects of the speaker‘s speech in a reduced set of measures, in our work 400, instead of 39 PLP coefficients extracted every 10 ms. Note that with PLPs, we would have 390 parameters in only 100 ms of speech. Their ability to capture intelligibility information and the *Accuracy* that a speech recognizer would obtain was shown with the experiments presented in this part of the Thesis. *i-Vectors* in this scenario were computed in the same way as for LID. It only varied the database used to train the UBM and FA extractor, and the parametrization of the speech signal. Also, different optimal values were found for the number of Gaussian components in the UBM and for the dimension of the *i-Vectors*. Given that each audio recording was represented by a single *i-Vector*, simple predictors could be used, such as ν-SVR.

Regarding the usefulness of *i-Vectors*, it was shown that they fulfilled the desired conditions to capture intelligibility information. One interesting observation was that

i-Vectors belonging to *low* intelligible speakers were more different among them than *i-Vectors* belonging to *high* intelligible speakers. This is due to the variability in the speech production of severe dysarthric speakers, whose utterances can vary a lot from realization to realization, even if they say the same word. Finally, *i-Vectors* were compared with other approaches to assess intelligibility. The conclusion was that *i-Vectors* kept the important information to make intelligibility assessments and *Accuracy* predictions better than the rest of studied techniques while performing a more efficient compression. We believe that this work opens an interesting research direction in this field

In the present work, the acoustic information was captured with the PLP coefficients. These features are not especially designed for intelligibility assessment, and we think that the results could be improved by adding other features more specific for this task. However, our results were competitive and comparable to other works in the literature, like Falk et al. [2012], where $r = 0.94$, and $RMSE = 0.186$ were obtained over only ten spastic speakers from the UASpeech database, using six features representing atypical vocal source excitation, temporal dynamics, and prosody. In that work, the information about the application user was also included in training.

Part IV

Conclusions of the Thesis and Quality Metrics

17

Conclusions of the Thesis

Linear Gaussian models play a central role in machine learning. The assumption of this type of models is that high dimensional data can be explained by a reduced set of dimensions where most information is contained. In these models, the observed variables are factorized in one or more terms, which span these low-dimension subspaces. Effectively, subspaces in linear Gaussian models allow us to train Gaussian- or GMM-based distributions with a more flexible covariance structure with less parameters than a conventional full covariance matrix. Thus, we avoid some of the problems of the curse of dimensionality, because we train fewer number of parameters more robustly, and we reduce the risk of overtraining. In addition, we saw that this data factorization can bring other advantages, depending on how the factors are related with each other. These relationships are commonly seen in graphical models.

Mixture models like PPCA, FA, MPPCA, or MFA, assume that there is a linear factor that explains the generation of each feature in a sequence independently. However, TMFA or its approximation, JFA, consider that the same linear factor is responsible for the generation of a sequence of features. This adds a very interesting potential to mixture models, because a different component is responsible for each observed feature, and given that the hidden factor is common to all observations, we can learn the correlations between the different components of the mixture. For example, if we assume that a sequence was generated by the same channel, the hidden factor can be used to model the channel. Then, once this channel factor is calculated using the correlations among the activations of different mixture components, the model can be used to compensate for channel mismatches, as we did in this Thesis. Likewise, other entities,

like the speaker, can be modeled with this technique. The explanation of this type of models is the main topic of Part I of the Thesis. In Part II and Part III, we have investigated linear Gaussian models applied on two different fields: LID and dysarthria intelligibility assessment, respectively.

In Part II, we speak about the application of linear Gaussian models on LID. We presented two different approaches depending on the type of features used: acoustic features and prosodic with formant information features. In the first case, the use of SDC as spectral information is very common among researchers of the LID community. These are stacked delta MFCC features of future frames. In this way, long term information of the signal is captured. In addition, MFCC features are also stacked to include the spectral information of the current frame. Meanwhile, prosodic features try to extract information at other communication level. Rhythm, stress and intonation are captured with pitch, energy, and duration features. All this information is conveyed in a longer time span than the spectral information reflected in the acoustic features. In order to capture this information along time, we used Legendre polynomials to model contours of 200 ms long. In such a way, we work with, not only the current value of the feature, but with the evolution of this parameters in this time interval. Additionally, we included formant information to these features, and those were modeled in the same way. Formants are very useful to disambiguate vowels, and its evolution along time might even be helpful to model prosody.

In the literature, the most successful classification technique for acoustic features had been GMM with channel compensation, also known as eigenchannel modeling, or JFA, if we follow the same notation as for speaker recognition. However, unlike speaker recognition, where JFA included eigenvoices to adapt the speaker, and eigenchannels to compensate the channel, only one factor was used to compensate the channel for LID, and the language was modeled via MAP from the UBM. Regarding prosodic features, no experiments had been reported with JFA for LID, although they had been reported for speaker recognition with success [Dehak et al., 2007b,a]. The closest technique previously tested with prosodic features for LID were GMM without channel compensation.

One major contribution of the Thesis is the proposal of using JFA as feature extractor instead of classifier, similarly to speaker recognition. In this approach, a unique JFA model is built with a single low-dimension hidden factor. This factor is tied to full

recordings and it includes all sources of variability. Thus, the subspace spanning this variable is called total variability subspace, and the vectors lying in this subspace are the well-known *i-Vectors*. Hence, the fixed-length and low-dimension *i-Vectors* were the new features to do classification. Note that each utterance was represented by one *i-Vector*. Several classifiers were built on top of *i-Vectors*, both generative and discriminative. The best-performing one was a Gaussian classifier, where each language is modeled with a single Gaussian distribution, with covariance matrix equal to the WC covariance matrix of the training data, which is shared among all classes. Nonetheless, recent experiments have shown that it is possible to learn a different covariance matrix for each language, and even different modes have been found within each language, if the amount of training data is large enough [Lopez-Moreno et al., 2014]. Note that the *i-Vectors* contain all sources of information, including language, but also others undesired like channel. Surprisingly, no compensation techniques helped to improve performance.

The architecture proposed with *i-Vectors* and the Gaussian classifier was successful for acoustic and prosodic features. The experiments in this Thesis showed improvements of *i-Vectors* over the JFA for all duration conditions. We checked that *i-Vectors* are more unreliable for short durations than for long durations, because their uncertainty is grater in this case (it directly depends on the number of frames used to compute it). The performance of acoustic features was higher for acoustic than for prosodic features, but they are complementary, and the fusion allowed reducing the number of errors with regard to the acoustic system alone.

Today, *i-Vector*-based classifiers are the state of the art for LID. Moreover, the Gaussian classifier is one of the preferred options selected by most of the researchers due to its simplicity and accuracy. Lately, the use of DNN has been shown to be very promising and they could become the new generation of LID classifiers. They have shown to be especially competitive for short durations [Lopez-Moreno et al., 2014].

The Part III of the Thesis is devoted to the use of *i-Vectors* with dysarthric speech. Concretely, we have tested two different applications with great interest for clinicians. The goal of the first application is to assess the intelligibility of dysarthric speakers from a set of words. Intelligibility assessment is important to perform objective assessments and measure the progress of the patients without the subjective influence of speech therapists. Our hypothesis was that *i-Vectors* are a set of measures that include

all sources of speech variability (in this case obtained from PLP features), including intelligibility. We studied two situations differing in the availability or not, for training, of data of the person that will use the application. The results were very different, because if we had access to data of the application user for training, the accuracy of the automatic predictions was much higher. In numbers, in the case of having data of the user for training, and using 400-dimension *i-Vectors*, we achieved a Pearson correlation of 0.91 and a RMSE of 0.14, while in the case where data of the user was not available for training, these metrics fell to 0.74 and 0.23, respectively.

For making the assessments, we compared linear regression and SVR with a radial basis function kernel, and better results were obtained with the second technique. At this point, we also have to take into account that current databases for experimentation were limited, and the set of speakers and recordings was reduced. We used UASpeech database and we think that this lack of data affected especially to the low intelligibility ratings. To overcome this problem, at least partially, we also performed a classification task, instead of prediction. By grouping the speaker into groups, we had more information for training a determined intelligibility interval. Note that, whereas in the assessment task the intelligibility rating is between 0 and 100, from very low to high intelligibility ratings, in the classification task, the output of the system is coarser, and e.g for a 4-class problem, it only says if the intelligibility is low, very low, medium or high.

We realized that the same configuration used for intelligibility assessment could be easily adapted to the problem of predicting the accuracy that an ASR system would obtain with the dysarthric speakers. Thus, ASR accuracy prediction is the second application that we have investigated. This task is very beneficial for clinicians who want to use an ASR system with their patients for different purposes, because they can know beforehand if the system will be useful. Moreover, it would increase confidence in this technology and it would diminish health related costs. As in the case of intelligibility assessment, our hypothesis was that *i-Vectors* also contain this kind of information. The experimental methodology was also similar to the previous task, and predictions were made after the speakers uttered a set of words. However, the performance was not as good. If we used data of the application user for training, we obtained a Pearson correlation of 0.89 and an RMSE of 0.12. However, if we did not include those data for training, we obtained 0.55 and 0.22 for the same metric, respectively.

In our experiments, the *i-Vector* subspace was trained using only non-dysarthric data. First, for practical reasons, we needed big amount of data to robustly estimate the subspace via ML. Second, our idea was that if the speech was intelligible, the estimated *i-Vector* would fit very well in this subspace, whereas the more unintelligible the speech, the further the estimated *i-Vector* from the intelligible *i-Vectors*. However, it might happen that including dysarthric speech in the *i-Vector* subspace training dataset helped to estimate more robust and reliable *i-Vectors* for dysarthric speakers. This is an interesting future research line. We checked that training the subspace only with the dysarthric speech contained in the UASpeech database is very harmful, but intermediate solutions, like MAP adaptation from the models without dysarthric speech, might be successful.

Finally, we analyzed the goodness of *i-Vectors* for intelligibility assessment, and we found that they are following the expected behavior. We can say that at least, results are promising, and we have opened a new research direction.

18

Thesis Quality Metrics

Contents

In this chapter, we list all the publications made during the development of this Thesis. Also, we include some metrics that guarantee that the Thesis fulfill criteria of high quality standards, like the h-factor or number of article citations and downloads in common research networks.

18.1 Publications

We list all the publications in chronological order. Three papers were included in the *Best Student Paper Award List* of the corresponding conference.

JOURNALS

- **Intelligibility Assessment and Speech Recogniser Word Accuracy Rate Prediction for Dysarthric Speakers in a Factor Analysis Subspace**. David Martínez, Eduardo Lleida, Phil Green, Heidi Christensen, Alfonso Ortega, Antonio Miguel. Accepted to be published in *ACM Transactions on Accessible Computing*, special issue on Speech and Language Interaction for Daily Assistive Technology. 2015.

- **On the use of Deep Feedforward Neural Networks for Automatic Language Identification**. Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, David Martínez, Oldřich Plchot, Joaquin Gonzalez-Rodriguez, Pedro J. Moreno. Submitted to *IEEE Transactions on Audio, Speech, and Language Processing*. 2015.

BOOK CHAPTERS

- **Unsupervised Accent Modeling for Language Identification**, David Martínez, Jesús Villalba, Eduardo Lleida, Alfonso Ortega. *Advances in Speech and Language Technologies for Iberian Languages*, Lecture Notes in Computer Science Volume 8854, pp. 49-58. Springer 2014. Presented in *Iberspeech* Las Palmas de Gran Canaria, Spain. 2014. [Martínez et al., 2014b].

- **Voice Pathology Detection on the Saarbrücken Voice Database with Calibration and Fusion of Scores Using MultiFocal Toolkit**, David Martínez, Eduardo Lleida, Alfonso Ortega, Antonio Miguel, Jesús Villalba. *Advances in*

Speech and Language Technologies for Iberian Languages, Communications in Computer and Information Science Volume 328, pp. 99-109. Springer 2012. Presented in *Iberspeech* 2012, Madrid, Spain. 2012. **BEST STUDENT PAPER AWARD LIST**. [Martínez et al., 2012d].

- **Score Level versus Audio Level Fusion for Voice Pathology Detection on the Saarbrücken Voice Database**, David Martínez, Eduardo Lleida, Alfonso Ortega, Antonio Miguel. *Advances in Speech and Language Technologies for Iberian Languages*, Lecture Notes in Computer Science Volume 8854, pp. 110-120. Springer 2012. Presented in *Iberspeech* 2012, Madrid, Spain. 2012. **BEST STUDENT PAPER AWARD LIST**. [Martínez et al., 2012c].

INTERNATIONAL CONFERENCES

- **Unscented Transform for iVector-Based Noisy Speaker Recognition**, David Martínez, Lukáš Burget, Themos Stafylakis, Yun Lei, Patrick Kenny, Eduardo Lleida. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4042-4046, Florence, Italy. 2014. [Martínez et al., 2014a].

- **Automatic Language Identification Using Deep Neural Networks**, Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, Oldřich Plchot, David Martínez, Joaquin Gonzalez-Rodriguez, Pedro Moreno. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5337-5341, Florence, Italy. 2014. [Lopez-Moreno et al., 2014].

- **Suprasegmental Information Modelling for Autism Disorder Spectrum and Specific Language Impairment Classification**, David Martínez, Dayana Ribas, Eduardo Lleida, Alfonso Ortega, Antonio Miguel. *INTERSPEECH*, pp. 195-199, Lyon, France. 2013. [Martínez et al., 2013c].

- **Dysarthria Intelligibility Assessment in a Factor Analysis Total Variability Space**, David Martínez, Phil Green, Heidi Christensen. *INTERSPEECH*, pp. 2133-2137, Lyon, France. 2013. [Martínez et al., 2013a].

- **Prosodic Features and Formant Modeling for an iVector-Based Language Recognition System**, David Martínez, Eduardo Lleida, Alfonso Ortega, Antonio Miguel. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 6847-6851, Vancouver, Canada. 2013. [Martínez et al., 2013b].

- **The BLZ Submission to the NIST 2011 LRE: Data Collection, System Development and Performance**, L.J. Rodríguez, M. Penagarikano, A. Varona, M. Díez, G. Bordel, A. Abad, David Martínez, J. Villalba, A. Ortega, E. Lleida. *INTERSPEECH*, pp. 38-41, Portland, USA. 2012. [Rodríguez-Fuentes et al., 2012].

- **iVector-Based Prosodic System for Language Identification**, David Martínez, Lukáš Burget, Luciana Ferrer, Nicolas Scheffer. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4861-4864, Kyoto, Japan. 2012. [Martínez et al., 2012a].

- **The BLZ Systems for the 2011 NIST Language Recognition Evaluation**, L.J. Rodríguez, M. Penagarikano, A. Varona, M. Díez, G. Bordel, A. Abad, D. Martínez, J. Villalba, A. Ortega, E. Lleida. *NIST 2011 LRE Workshop*, Atlanta, USA. 2011. [Rodríguez-Fuentes et al., 2011a].

- **I3A Language Recognition System Description for NIST LRE 2011**, David Martínez, Jesús Villalba, Alfonso Ortega, Eduardo Lleida. *NIST 2011 LRE Workshop*, Atlanta, USA. 2011. [Martínez et al., 2011c].

- **Multi-Site Heterogeneous System Fusions for the Albayzin 2010 Language Recognition Evaluation**, L.J. Rodríguez, M. Penagarikano, A. Varona, M. Díez, G. Bordel, D. Martínez, J. Villalba, A. Miguel, A. Ortega, E. Lleida, A. Abad, O. Koller, I. Trancoso, P. López, L. Docio, C. García, R. Saeidi, M. Soufifar, T. Kinnunen, T. Svendsen, P. Fränti. *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 377-382, Waikoloa, USA. 2011. [Rodríguez-Fuentes et al., 2011b]

- **Hierarchical Audio Segmentation with HMM and Factor Analysis in Broadcast News Domain**, Diego Castán, Carlos Vaquero, Alfonso Ortega,

David Martínez, Jesús Villalba, Eduardo Lleida. *INTERSPEECH*, pp. 421-424, Florence, Italy. 2011. [Castán et al., 2011].

- **Language Recognition in iVectors Space**, David Martínez, Oldřich Plchot, Lukáš Burget, Ondřej Glembek, Pavel Matěka. *INTERSPEECH*, pp. 861-864, Florence, Italy. 2011. **BEST STUDENT PAPER AWARD LIST**. [Martínez et al., 2011b].

- **I3A Language Recognition System for Albayzin 2010 LRE**, David Martínez, Jesús Villalba, Antonio Miguel, Alfonso Ortega, Eduardo Lleida. *INTERSPEECH*, pp. 849-852, Florence, Italy. 2011. [Martínez et al., 2011a].

NATIONAL CONFERENCES

- **Albayzin 2012 LRE @ ViVoLab**, David Martínez, Eduardo Lleida, Alfonso Ortega. *Iberspeech* 2012, Madrid, Spain. 2012. [Martínez et al., 2012b]

- **ViVoLab UZ Language Recognition System for Albayzin 2010 LRE**, David Martínez, Jesús Villalba, Antonio Miguel, Alfonso Ortega, Eduardo Lleida. *Fala, VI Jornadas en Tecnología del Habla and II SLTech Workshop*, Vigo, Spain. 2010. [Martinez et al., 2010]

18.2 Evaluations

- **INTERSPEECH 2013 Computational Paralinguistics Challenge (ComParE) - Autism Subchallenge**, David Martínez, Dayana Ribas, Eduardo Lleida, Alfonso Ortega, Antonio Miguel. *INTERSPEECH*, Lyon, France, 2013. Description in Martínez et al. [2013c]. 2nd position.

- **Albayzin 2012 Language Recognition Evaluation**, David Martínez, Eduardo Lleida, Alfonso Ortega. *Iberspeech* 2012, Madrid, Spain. 2012. Description in Martínez et al. [2012b]. 2nd position.

- Double participation in **NIST Language Recognition Evaluation**, Atlanta, 2011:

  - I3A Team: David Martínez, Jesús Villalba, Alfonso Ortega, Eduardo Lleida. Description in Martínez et al. [2011c].
  - BLZ Team: L.J. Rodríguez, M. Penagarikano, A. Varona, M. Díez, G. Bordel, A. Abad, D. Martínez, J. Villalba, A. Ortega, E. Lleida. Description in Rodríguez-Fuentes et al. [2011a]. A secondary submission was the best system in all the evaluation.

- **Albayzin 2010 Language Recognition Evaluation**, David Martínez, Jesús Villalba, Antonio Miguel, Alfonso Ortega, Eduardo Lleida. *Fala, VI Jornadas en Tecnología del Habla and II SLTech Workshop*, Vigo, Spain. 2010. Description in Martinez et al. [2010]. 1st position.

18.3 Reviewer

The author is reviewer of the following conferences and journals

- INTERSPEECH, ISCA
- Odyssey: The Speaker and Language Recognition Workshop, ISCA
- Mathematical Problems in Engineering, Hindawi Publishing Corporation

18.4 Session Chair

The author was session chair of the session Dialect and Accent Identification (Tue-Ses2-O1) at conference INTERSPEECH 2011.

18.5 Google Scholar Metrics

Google Scholar[1] has become a standard to track the number of citations of the articles of researchers in all fields. It also computes their h-index and their i10-index (definitions can be seen in the Google scholar website). We must be critical with this source of information though. The computation of citations is made by automatic robots that crawls for paper from all across the web, and it is known that it is not 100% accurate.

[1] `http://scholar.google.com`

Title	Year	Nr. of citations
Language Recognition in iVectors Space	2011	91
Automatic Language Identification Using Deep Neural Networks	2014	15
iVector-based Prosodic System for Language Identification	2012	14
Hierarchical Audio Segmentation with HMM and Factor Analysis in Broadcast News Domain	2011	8
Voice Pathology Detection on the Saarbrücken Voice Database with Calibration and Fusion of Scores Using MultiFocal Toolkit	2012	7
Prosodic Features and Formant Modeling for an iVector-Based Language Recognition System	2013	6
The BLZ Submission to the NIST 2011 LRE: Data Collection, System Development and Performance	2012	6
Multi-Site Heterogeneous System Fusions for the Albayzin 2010 Language Recognition Evaluation	2011	6
I3A Language Recognition System for Albayzin 2010 LRE	2011	4
Unscented Transform for iVector-Based Noisy Speaker Recognition	2014	2
I3A Language Recognition System Description for NIST LRE 2011	2011	2
Suprasegmental Information Modelling for Autism Disorder Spectrum and Specific Language Impairment Classification	2013	1
Dysarthria Intelligibility Assessment in a Factor Analysis Total Variability Space	2013	1
ViVoLab UZ Language Recognition System for Albayzin 2010 LRE	2010	1

Table 18.1: Google Scholar article citations - In first column, article title; in second column, year of publication; in third column, number of citations at May 30th, 2015.

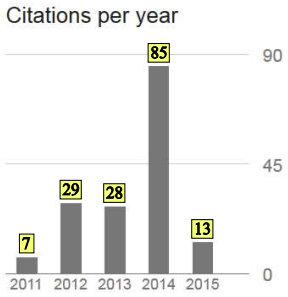


Figure 18.1: Number of citations per year in Google Scholar - Bar diagram to show the number of citations per year of the author at May 30th, 2015.

It takes into account unpublished works, and sometimes it counts citations twice and it gets confused with other person's work with the same name as you. Probably, these are not the only failures. Despite all the known inconveniences, we think that is a good site to gather an idea of how relevant your works is. Nonetheless, it is being constantly improved, searching in the whole web also has the advantage of having an enormous source of information, and it is free.

In Table 18.1, we have collected the citations of each of the author's articles at date May 30th, 2015. In Figure 18.1, we have included the total number of citations per year. Finally, in Figure 18.2, we show a screenshot of the author's Google Scholar site that captures the total number of citations, h-index and i10-index at that date.

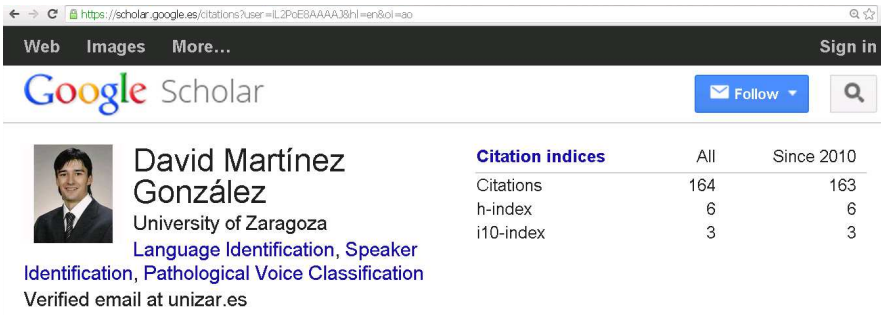


Figure 18.2: Citation indeces in Google Scholar - Screenshot of Google Scholar to show the author's citation index metrics: total number of citations, h-index, and i10-index. Data taken at May 30th, 2015.

18.6 ResearchGate Metrics

ResearhGate is a social network for researchers[1]. In this site, researchers can list and upload their publications and other researchers can easily access to them. In addition, there is a count of the number of times that each of your publications is downloaded and cited. However, it is warned that the count is not exhaustive and it is based on the publications that are available in the site's database, and the final metrics depend on how active you are in the network and how many contacts you have. In Figure 18.3, we have made a screenshot of the author's site overview on ResearchGate. In general, it seems less accurate than Google Scholar, but given that the number of researchers with a profile is high, it can be a good indicator of how many researchers are interested in your work.

[1]https://www.researchgate.net

Figure 18.3: ResearchGate Statistics - Screenshot of ResearchGate of the author's statistics overview site. It is shown number of publications, number of profile views, number of publication downloads, number of citations, number of impact points (a ResearchGate metric that counts the impact factors of the conferences and journals where your articles are published), and RG Score (a ResearchGate own metric which is based on the publications in your profile and how other researchers interact with your content on ResearchGate) to the right of the green bar.

APPENDIX A

Linear Gaussian Models

Contents

In this Appendix we present some mathematical formulations of the linear Gaussian models described in Chapter 2.

A.1 Principal Component Analysis

A.1.1 Mathematical Formulation of PCA

PCA goal is to maximize the variance of the projected data while keeping the dimensions uncorrelated. This means that the covariance matrix of the projected data must be diagonal, with values as large as possible. This can be done by maximizing the trace of the projected covariance matrix. Consider a set of data $\mathbf{O}$, of dimension DxN, where N is the number of samples, and D the dimension of each sample. We want to find the projection $\mathbf{P}$, such that $\mathbf{X} = \mathbf{P}^\intercal\mathbf{O}$. Then, our objective function is

$$L = \max(Tr(\mathbf{C}_x))) = \max(Tr(\mathbf{P}^\intercal\mathbf{C}_o\mathbf{P})). \tag{A.1}$$

We also want that $\mathbf{P}^\intercal\mathbf{P} = \mathbf{I}$, to avoid that the projection matrix adds variance in the projected space that did not exist in the original one. Thus, our projection matrix must be orthonormal. To fulfill this constraint we use the method of Lagrange multipliers. The solution is the optimal $\mathbf{P}$ that maximizes the trace

$$\begin{aligned} \underset{\mathbf{P}^*}{\operatorname{argmax}}(Tr(\mathbf{C}_x)) &= \underset{\mathbf{P}^*}{\operatorname{argmax}}(Tr(\mathbf{P}^\intercal\mathbf{C}_o\mathbf{P} - \Lambda(\mathbf{I} - \mathbf{P}^\intercal\mathbf{P}))) \\ &= \underset{\mathbf{P}^*}{\operatorname{argmax}} \sum_{i=1}^{D} [\mathbf{p_i}^\intercal\mathbf{C}_o\mathbf{p_i} + \lambda_i(1 - \mathbf{p_i}^\intercal\mathbf{p_i})] \end{aligned} \tag{A.2}$$

where $\mathbf{\Lambda}$ is a diagonal matrix, with Lagrange multipliers λ_i in the diagonal, and we have used of the definition of trace in the last equality.

We derive the objective function with respect to each $\mathbf{p_i}$, and we make it equal to 0:

$$\frac{dL}{d\mathbf{p_i}} = 2\mathbf{C}_x\mathbf{p_i} - 2\lambda_i\mathbf{p_i} = 0 \Rightarrow \mathbf{C}_x\mathbf{p_i} = \lambda_i\mathbf{p_i} \tag{A.3}$$

As we can see, this is an eigenvector problem. Thus, the solution is that the basis vectors are the eigenvectors of the covariance matrix of the original data. The variance of the rotated data in each of those directions is the eigenvalue of the corresponding eigenvector

$$\lambda_i = \mathbf{p_i}^\intercal C_x\mathbf{p_i}. \tag{A.4}$$

Alternatively, it can be shown that the eigenvectors of the covariance matrix can also be obtained through a singular value decomposition (SVD) of the data $\mathbf{X}$ [Shlens, 2014].

A.2 Probabilistic Principal Component Analysis

In this Section we present two different ML approaches to compute the parameters of a PPCA model. The first is based on direct differentiation of the log-likelihood function and the second is based on the EM algorithm.

A.2.1 Maximum Likelihood Formulation

The first approach to obtain the ML estimation of the model parameters is to differentiate the log-likelihood function, and make the derivatives with respect to the parameters equal to 0. The log-likelihood is defined by taking the logarithm of eq. (2.8). If we have N samples, $\mathbf{O} = \mathbf{o}_1 ... \mathbf{o}_N$, we obtain

$$\begin{aligned} \ln p(\mathbf{O}|\mu, \mathbf{W}, \sigma^{\mathbf{2}}) &= \sum_{n=1}^{N} p(\mathbf{o}|\mu, \mathbf{W}, \sigma^{\mathbf{2}}) \\ = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{o}_n - \mu)^{\intercal}\mathbf{C}^{-1}(\mathbf{o}_n - \mu) \\ &= -\frac{N}{2}[D\ln(2\pi) + \ln|\mathbf{C}| + \mathrm{Tr}(\mathbf{C}^{-1}\mathbf{S})] \end{aligned} \tag{A.5}$$

with $\mathbf{S}$ the sample covariance matrix of the data, $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{o}_n - \mu)(\mathbf{o}_n - \mu)^{\intercal}$.

We derive this equation with respect to the parameters μ, $\mathbf{W}$, and $\sigma^{\mathbf{2}}$, and equal the derivative to 0. Then, we arrive to the solution for the mean

$$\mu = \frac{1}{N}\sum_{n=1}^{N}\mathbf{o}_n. \tag{A.6}$$

The ML solution for $\mathbf{W}$ can be shown to be [Roweis, 1997; Tipping and Bishop, 1999b; Bishop, 2006]

$$\mathbf{W} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \tag{A.7}$$

with $\mathbf{U}_M$ a DxM matrix, whose columns are the M eigenvectors of $\mathbf{S}$ with highest eigenvalues, $\mathbf{L}_M$ is a MxM diagonal matrix with the M corresponding eigenvalues, and $\mathbf{R}$ is an arbitrary MxM orthogonal matrix, which can be chosen to be the identity

matrix for simplicity. It can be shown that the maximum solution occurs when $\mathbf{U}_M$ contains the M leading eigenvectors. Otherwise we will be located in a saddle point of the log-likelihood function. If we sort the eigenvectors by decreasing value of their eigenvalues, the matrix $\mathbf{W}$ will correspond to the same subspace found by PCA (but scaled by the variance parameters $\mathbf{L}_M - \sigma^2\mathbf{I}$).

And the ML solution for σ^2 is

$$\sigma^2 = \frac{1}{D-M}\sum_{j=M+1}^{D}\lambda_j, \tag{A.8}$$

where λ are the eigenvalues of the covariance matrix. As we can see, the sum goes over the eigenvalues not included in $\mathbf{W}$, so σ^2 accounts for the lost variance due to the dimensions not included in the transformation matrix. Given that the covariance of $\mathbf{o}$ given $\mathbf{x}$ is isotropic, we lose the individual variance in each dimension. The individual variances will be contained in the subspace $\mathbf{W}$, which also models the correlations among different dimensions. That's the reason why PPCA does not distinguish between variance and covariance.

A.2.2 EM Formulation

It may seem pointless to look for an EM solution when we have found a closed-form ML solution to the problem. However, in high dimensionality spaces, there may be a big computational load reduction by using an EM solution.

The EM states that the maximization of the data log-likelihood, $\ln p(\mathbf{O}|\mathbf{\Theta})$, can be done by maximizing the expectation of the logarithm of the joint distribution of the observed and latent variables, $\ln p(\mathbf{O}, \mathbf{X}|\mathbf{\Theta})$, with respect to the probability of the latent variables given the observed variables and the parameters in the previous iteration, $p(\mathbf{X}|\mathbf{O}, \mathbf{\Theta}_{old})$. The logarithm of the joint distribution can be expressed, using the *product rule* of probability, as

$$\ln p(\mathbf{O}, \mathbf{X}|\mu, \mathbf{W}, \sigma^2) = \sum_{n=1}^{N}[\ln p(\mathbf{o}_n|\mathbf{x}) + \ln p(\mathbf{x})]. \tag{A.9}$$

Then, our objective function $\mathcal{Q}$, can be expressed as

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\Theta}, \boldsymbol{\Theta}_{old}) &= \mathbb{E}_{\mathbf{X}|\mathbf{O}}[\ln p(\mathbf{O}, \mathbf{X}|\mu, \mathbf{W}, \sigma^{\mathbf{2}})] \\ = -\sum_{n=1}^{N}[\frac{D}{2}\ln(2\pi\sigma^2) + \frac{M}{2}\ln(2\pi) + \frac{1}{2}\operatorname{Tr}(\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal]) + \frac{1}{2\sigma^2}||\mathbf{o}_n - \mu||^2 \\ -\frac{1}{\sigma^2}\mathbb{E}[\mathbf{x}_n]^\intercal\mathbf{W}^\intercal(\mathbf{o}_n - \mu) + \frac{1}{2\sigma^2}\operatorname{Tr}(\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal]\mathbf{W}^\intercal\mathbf{W})]\end{aligned} \tag{A.10}$$

where we have used eqs. 2.2 and 2.4.

In the E step, we evaluate the expectations of $\mathbf{X}$ given $\mathbf{O}$ using the old parameters,

$$\mathbb{E}[\mathbf{x}_n] = \mathbf{B}^{-1}\mathbf{W}^\intercal(\mathbf{o}_n - \mu), \tag{A.11}$$

$$\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal] = \sigma^2 B^{-1} + \mathbb{E}[\mathbf{x}_n]\mathbb{E}[\mathbf{x}_n]^\intercal, \tag{A.12}$$

where we have used the relationship $\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal] = cov(\mathbf{x}_n) + \mathbb{E}[\mathbf{x}_n]\mathbb{E}[\mathbf{x}_n]^\intercal$, and the expression $\mathbb{E}[\mathbf{x}_n]$ indicates the expectation of hidden variable $\mathbf{x}$ at time n.

In the M step, new parameters are calculated by deriving eq. (A.10) with respect to $\mathbf{W}$ and $\sigma^{\mathbf{2}}$ while keeping $\mathbb{E}[\mathbf{x}_n]$ and $\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal]$ fixed, and setting the derivative equal to 0. We obtain the following update equations

$$\mathbf{W_{new}} = [\sum_{n=1}^{N}(\mathbf{o}_n - \mu)\mathbb{E}[\mathbf{x}_n]^\intercal][\sum_{n=1}^{N}\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal]]^{-1}, \tag{A.13}$$

$$\sigma^{\mathbf{2}}_{\mathbf{new}} = \frac{1}{ND}\sum_{n=1}^{N}[||\mathbf{o}_n - \mu||^2 - 2\mathbb{E}[\mathbf{x}_n]^\intercal\mathbf{W_{new}}^\intercal(\mathbf{o}_n - \mu) + \operatorname{Tr}(\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal]\mathbf{W_{new}}^\intercal\mathbf{W_{new}})], \tag{A.14}$$

and the solution for μ is the one obtained by ML in eq. (A.6).

A.3 Factor Analysis

A.3.1 EM Formulation

The maximum likelihood solution for μ is given by the sample mean of the data, $\mu = \frac{1}{N}\sum_{n=1}^{N} o_n$, with N the number of data points. However, for $\mathbf{W}$ and $\boldsymbol{\Psi}$ there is no closed-form solution. The solution must be found iteratively, for example with the EM algorithm. The formulation process is analogous to the EM for PPCA. In the E

step, we calculate the expectation of the logarithm of the joint probability distribution of observed and unobserved variables, which requires computing

$$\mathbb{E}[\mathbf{x}_n] = \mathbf{G}\mathbf{W}^\intercal\Psi^{-1}(\mathbf{o}_n - \mu), \tag{A.15}$$

$$\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal] = \mathbf{G} + \mathbb{E}[\mathbf{x}_n]\mathbb{E}[\mathbf{x}_n]^\intercal, \tag{A.16}$$

with $\mathbf{G} = (\mathbf{I} + \mathbf{W}^\intercal\Psi^{-1}\mathbf{W})^{-1}$.

In the M step, we derive the objective function with respect to the parameters, and make it equal to 0, to obtain

$$\mathbf{W_{new}} = [\sum_{n=1}^{N}(\mathbf{o}_n - \mu)\mathbb{E}[\mathbf{x}_n]^\intercal][\sum_{n=1}^{N}\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal]]^{-1}, \tag{A.17}$$

$$\mathbf{\Psi_{new}} = \text{diag}\{\mathbf{S} - \mathbf{W_{new}}\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[\mathbf{x}_n](\mathbf{o}_n - \mu)^\intercal\} \tag{A.18}$$

where $\mathbf{S}$ is the sample covariance matrix, and the diag operator builds a matrix with all off-diagonal elements set to 0.

Because of the distinction between variance and covariance in FA, the maximum likelihood estimates of the columns of $\mathbf{W}$ do not generally correspond to the eigenvectors of the sample covariance matrix of the data. In addition, the basis found in a K-factor model are not necessarily the same that the basis found in a $(K-1)$-factor model, whereas in PPCA they are the same, because in PPCA the basis corresponds to the K and $K-1$ eigenvectors with highest associated eigenvalue. Another difference between FA and PPCA is the behavior under some transformations. In PPCA, if we rotate the coordinate system, we obtain the same fit to the data but with different $\mathbf{W}$, which will be rotated. The analogous property to rotation in FA is scaling. In FA, if we make a component-wise re-scaling of the data, the elements of $\mathbf{\Psi}$ will be scaled.

As in PPCA, if we rotate the $\mathbf{W}$ matrix with an orthogonal matrix $\mathbf{R}$, we obtain the same model. However, the matrix $\mathbf{R}$ can not be recovered, and then, there is no uniqueness in the solution. This is one of the major criticisms of FA, because it is a model that tries to explain correlations, but there is no uniqueness in the solution of the matrix that explains those correlations.

A.4 Gaussian Mixture Model

A.4.1 EM for GMM

In order to apply the EM algorithm, first we have to define a K-dimension binary random latent variable, $\mathbf{z}$, having a $1-of-K$ representation, in which a particular z_k element is equal to 1 and the rest are 0 if the Gaussian component k generated the data. Thus if

$$p(z_k = 1) = \omega_k, \tag{A.19}$$

then

$$p(\mathbf{z}) = \prod_{k=1}^{K} \omega_k^{z_k}. \tag{A.20}$$

The conditional distribution of $\mathbf{o}$ given that a particular component k is active, $z_k = 1$, is Gaussian,

$$p(\mathbf{o}|\mathbf{z}) = \mathcal{N}(\mathbf{o}|\mu_k, \mathbf{\Sigma}_k)^{z_k}. \tag{A.21}$$

And the marginal distribution of $\mathbf{o}$ takes the form seen in equation 2.16. The conditional probability $z_k = 1$ given $\mathbf{o}$ will also play an important role. It is defined as the *responsibility* of component k, and can be found using Bayes' theorem,

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{o}) = \frac{p(z_k = 1)p(\mathbf{o}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{o}|z_j = 1)} = \frac{\omega_k \mathcal{N}(\mathbf{o}|\mu_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \omega_j \mathcal{N}(\mathbf{o}|\mu_j, \mathbf{\Sigma}_j)}. \tag{A.22}$$

Similarly, for a dataset with N samples we define the KxN variable $\mathbf{Z}$, where the value at row k and column n is z_{kn}, the latent variable at time n for component k. The corresponding log-likelihood function is defined in 2.17. Now we have all the ingredients to run the EM algorithm on the GMM log-likelihood function. The process is as follows:

1. Establish the number of Gaussian components K. Maybe, you can make this decision if you have some previous information about the number of modes in your data, or empirically by running different experiments on a separate partition of your training data. Once you know K, you can train the all components from the beginning simultaneously, or run a splitting strategy, where you start with one Gaussian, train the model with this component, split this Gaussian in H components in the directions of maximum variance, train the resulting model, split each Gaussian component in H components in the directions of maximum

variance, and repeat this process until you reach the desired number of Gaussians. Alternatively, there are methods in which K is calculated stochastically from the data, like Dirichlet process [Ferguson, 1973], but this is out of the scope of this work.

2. Initialize the means μ_k, covariances $\mathbf{\Sigma}_k$ and weights ω_k, and evaluate the current log-likelihood.

3. E step. Calculate the responsibilities $\gamma(z_k)$ with the current parameter values.

4. M step. Re-estimate the parameters using the current responsibilities

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_n(z_k) \mathbf{o}_n. \tag{A.23}$$

For full-covariance models

$$\mathbf{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_n(z_k)(\mathbf{o}_n - \mu_k^{new})(\mathbf{o}_n - \mu_k^{new})^\intercal. \tag{A.24}$$

For diagonal covariance models

$$\mathbf{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_n(z_k) \operatorname{diag}((\mathbf{o}_n - \mu_k^{new})^2). \tag{A.25}$$

where the diag operator builds a diagonal matrix with all elements off-diagonal set to 0.

For isotropic diagonal models

$$\mathbf{\Sigma}_k^{new} = \frac{1}{DN_k} \sum_{n=1}^{N} \gamma_n(z_k) ||(\mathbf{o}_n - \mu_k^{new})||^2. \tag{A.26}$$

$$\omega_k^{new} = \frac{N_k}{N}, \tag{A.27}$$

where

$$N_k = \sum_{n=1}^{N} \gamma_n(z_k). \tag{A.28}$$

5. Evaluate the log-likelihood function in 2.17. If convergence is not satisfied, return to step 3 with updated parameters.

A.5 Mixture of PPCAs

A.5.1 EM Algorithm for MPPCA

In this Section we summarize the E and M steps of the EM algorithm for MPPCA.

- E step:
  1. Use current parameters ω_k, μ_k, $\mathbf{W}_k$, and σ_k^2.
  2. Calculate expectations $\mathbb{E}[z_k|\mathbf{o}_n] = \gamma_n(z_k)$ as per eq. (A.22), the responsibilities of each Gaussian at time n.
- M step
  1. Obtain new Gaussian proportions
  $$\tilde{\omega}_k = \frac{N_k}{N}, \tag{A.29}$$
  where
  $$N_k = \sum_{n=1}^{N} \gamma_n(z_k). \tag{A.30}$$
  2. Obtain new means
  $$\tilde{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_n(z_k)\mathbf{o}_n. \tag{A.31}$$
  By merging the result of the E step with the update formulas for $\mathbf{W}_k$ and σ_k^2, we obtain the simplified M step formulas given next.
  3. First, we need to compute the responsibility weighted covariance matrix
  $$\mathbf{S}_k = \frac{1}{\tilde{\omega}_k N} \sum_{n=1}^{N} \gamma_n(z_k)(\mathbf{o}_n - \tilde{\mu}_k)(\mathbf{o}_n - \tilde{\mu}_k)^\intercal. \tag{A.32}$$
  4. Then, the update equation for $\mathbf{W}_k$ is
  $$\tilde{\mathbf{W}}_k = \mathbf{S}_k\mathbf{W}_k(\sigma_k^2\mathbf{I} + \mathbf{B}_k^{-1}\mathbf{W}_k^\intercal\mathbf{S}_k\mathbf{W}_k)^{-1}, \tag{A.33}$$
  with $\mathbf{B}_k = \mathbf{W}_k^\intercal\mathbf{W}_k + \sigma_k^2\mathbf{I}$.
  5. Finally, the update equation for σ_k^2 is
  $$\sigma_k^2 = \frac{1}{D}\operatorname{Tr}(\mathbf{S}_k - \mathbf{S}_k\mathbf{W}_k\mathbf{B}_k^{-1}\tilde{\mathbf{W}}_k). \tag{A.34}$$

A.6 Mixture of FAs

A.6.1 EM for MFA

In this Section we summarize the E and M steps of the EM algorithm for MFA.

- E step

  1. Use current parameters ω_k, μ_k, $\mathbf{W}_k$, and $\mathbf{\Psi}_k$.
  2. Estimate the responsibilities given by $\mathbb{E}[z_k|\mathbf{o}_n] = \gamma_n(z_k)$ according to eq. (A.22).
  3. Calculation of the following statistics,

$$\mathbb{E}[\mathbf{x}_n, z_k|\mathbf{o}_n] = \gamma_n(z_k)\beta_k(\mathbf{x}_n - \mu_k), \tag{A.35}$$

  where $\beta_k = \mathbf{W}_k^\intercal(\mathbf{\Psi}_k + \mathbf{W}_k\mathbf{W}_k^\intercal)^{-1}$.

$$\mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal, z_k|\mathbf{o}_n] = \gamma_n(z_k)(\mathbf{I} - \beta_k\mathbf{W}_k + \beta_k(\mathbf{o}_n - \mu_k)(\mathbf{o}_n - \mu_k)^\intercal\beta_k^\intercal). \tag{A.36}$$

  4. The estimation of μ and $\mathbf{W}$ will be made jointly, so we define $\tilde{\mathbf{x}} = [\mathbf{x}; 1]$, and

$$\mathbb{E}[\tilde{\mathbf{x}}_n, z_k|\mathbf{o}_n] = [\mathbb{E}[\mathbf{x}_n, z_k|\mathbf{o}_n]; \gamma_n(z_k)], \tag{A.37}$$

$$\mathbb{E}[\tilde{\mathbf{x}}_n\tilde{\mathbf{x}}_n^\intercal, z_k|\mathbf{o}_n] = \begin{bmatrix} \mathbb{E}[\mathbf{x}_n\mathbf{x}_n^\intercal, z_k|\mathbf{o}_n] & \mathbb{E}[\mathbf{x}_n, z_k|\mathbf{o}_n] \\ \mathbb{E}[\mathbf{x}_n, z_k|\mathbf{o}_n]^\intercal & \gamma_n(z_k) \end{bmatrix}. \tag{A.38}$$

- M step

  1. Joint computation of means and factor loading matrices

$$[\tilde{\mathbf{W}}_\mathbf{k}\ \tilde{\mu}_\mathbf{k}] = \tilde{\mathbf{\Lambda}}_k = (\sum_{n=1}^{N} \mathbf{o}_n\mathbb{E}[\tilde{\mathbf{x}}_n, z_k|\mathbf{o}_n]^\intercal)(\sum_{n=1}^{N} \mathbb{E}[\tilde{\mathbf{x}}_n\tilde{\mathbf{x}}_n^\intercal, z_k|\mathbf{o}_n])^{-1}. \tag{A.39}$$

  2. Computation of diagonal covariances,

$$\tilde{\mathbf{\Psi}}_k = \text{diag}\{\sum_{n=1}^{N}(\gamma_n(z_k)\mathbf{o}_n - \tilde{\mathbf{\Lambda}}_k\mathbb{E}[\tilde{\mathbf{x}}_n, z_k|\mathbf{o}_n])\mathbf{o_n}^\intercal\}/N_k, \tag{A.40}$$

  with $N_k = \sum_{n=1}^{N}\gamma_n(z_k)$.

  3. Computation of weights,

$$\tilde{\omega}_k = N_k/N. \tag{A.41}$$

A.7 Tied Mixture of Factor Analyzers

A.7.1 EM Algorithm for TMFA

In this Section we present the EM algorithm for the exact computation of TMFA. Our objective $\mathcal{Q}$ function of the EM algorithm is

$$
\begin{aligned}
\mathcal{Q}(\Theta,\Theta^{old}) &= \int \sum_{\bar{\mathbf{Z}}} p(\bar{\mathbf{Z}},\mathbf{X}|\bar{\mathbf{O}},\Theta^{old}) \ln p(\bar{\mathbf{Z}},\mathbf{X},\bar{\mathbf{O}}|\Theta) d\mathbf{X} \\
&= \int \sum_{s=1}^{K^N} p(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}}) p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) \ln[p(\bar{\mathbf{O}}|\bar{\mathbf{Z}}_s,\mathbf{X}) p(\mathbf{X}|\bar{\mathbf{Z}}_s) p(\bar{\mathbf{Z}}_s)] d\mathbf{X} \\
&= \int \sum_{s=1}^{K^N} p(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}}) p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) \\
&\cdot \ln[\prod_n \prod_k (\mathcal{N}(\mathbf{o}_n|\mu_k + \mathbf{W}_k\mathbf{x},\mathbf{\Sigma}_k))^{\delta_{k,s_n}} \mathcal{N}(\mathbf{X}|\mathbf{0},\mathbf{I}) \prod_n \prod_k (\omega_k)^{\delta_{k,s_n}}] d\mathbf{X} \\
&= \int \sum_{s=1}^{K^N} p(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}}) p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) \\
&\cdot \ln[\prod_n \prod_k (\omega_k \mathcal{N}(\mathbf{o}_n|\mu_k + \mathbf{W}_k\mathbf{x},\mathbf{\Sigma}_k))^{\delta_{k,s_n}} \mathcal{N}(\mathbf{X}|\mathbf{0},\mathbf{I})] d\mathbf{X} \\
&= \int \sum_{s=1}^{K^N} p(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}}) p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) [\sum_n \sum_k \delta_{k,s_n} \\
&\cdot \ln(\omega_k \mathcal{N}(\mathbf{O}_n|\mu_k + \mathbf{W}_k\mathbf{x},\mathbf{\Sigma}_k)) + \ln \mathcal{N}(\mathbf{X}|\mathbf{0},\mathbf{I})] d\mathbf{X} \\
&= \int \sum_{s=1}^{K^N} p(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}}) p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) [\sum_n \sum_k \delta_{k,s_n} \\
&\cdot \ln(\omega_k) + \sum_n \sum_k \delta_{k,s_n} \ln \mathcal{N}(\mathbf{O}_n|\mu_k + \mathbf{W}_k\mathbf{x},\mathbf{\Sigma}_k) + \ln \mathcal{N}(\mathbf{X}|\mathbf{0},\mathbf{I})] d\mathbf{X} \\
&= \sum_{s=1}^{K^N} p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) [\sum_n \sum_k \delta_{k,s_n} \ln(\omega_k) + \sum_n \sum_k \delta_{k,s_n} \\
&\cdot \ln \mathcal{N}(\mathbf{O}_n|\mu_k + \mathbf{W}_k \mathbb{E}(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}}),\mathbf{\Sigma}_k) + \ln \mathcal{N}(\mathbb{E}(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}})|\mathbf{0},\mathbf{I})] \\
&= \sum_{s=1}^{K^N} p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) \sum_n \sum_k \delta_{k,s_n} \ln(\omega_k) \\
&+ \sum_{s=1}^{K^N} p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) \sum_n \sum_k \delta_{k,s_n} \ln \mathcal{N}(\mathbf{O}_n|\mu_k + \mathbf{W}_k \mathbb{E}(\mathbf{X}|\bar{\mathbf{Z}}_s,\bar{\mathbf{O}}),\mathbf{\Sigma}_k) + \ln \mathcal{N}(\mathbb{E}(\mathbf{X}|\bar{\mathbf{O}})|\mathbf{0},\mathbf{I}),
\end{aligned}
\tag{A.42}
$$

where δ_{k,s_n} is the Kronecker delta, which is equal to 1 if $k = s_n$. We have taken into account that the expectation over the discrete hidden variable $\bar{\mathbf{Z}}$ exploits in a combinatorial term where we have to consider all possible permutations of the K components over the N frames of each file, and that given $\mathbf{X}$, the probability of

the observable variable at different frames becomes independent, and $p(\bar{\mathbf{O}}|\bar{\mathbf{Z}}_s, \mathbf{X}) = \prod_n \prod_k (\mathcal{N}(\mathbf{O}_n|\mu_k + \mathbf{W}_k\mathbf{x}, \mathbf{\Sigma}_k))^{\delta_{k,s_n}}$.

The next step is to substitute the normal function by its mathematical expression given in eq. (2.1). Then, we are in position to run the E step, where we compute the expected values of the hidden variable, $\mathbf{X}$, and of $\mathbf{X}\mathbf{X}^\intercal$, given the current parameters and the observations, and to run the M step, where we compute the new values of the model parameters. This process is summarized below.

- E step

  1. Computation of the first and second order statistics of the posterior distribution of $\mathbf{X}$ given the observations, which follows the following distribution,

$$p(\mathbf{X}|\bar{\mathbf{O}}) = \mathcal{N}(\mathbf{x}; \mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}], \mathbf{\Lambda}^{-1}). \tag{A.43}$$

  2. To compute the expectations, we apply the sum and product rules of probability to marginalize $\mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}]$ and $\mathbb{E}[\mathbf{X}\mathbf{X}^\intercal|\bar{\mathbf{O}}]$ over the K^N permutations [Miguel et al., 2014]

$$\mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}] = \sum_{s=1}^{K^N} p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}})\mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}, s], \tag{A.44}$$

$$\mathbb{E}[\mathbf{X}\mathbf{X}^\intercal|\bar{\mathbf{O}}] = \sum_{s=1}^{K^N} p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}})\mathbb{E}[\mathbf{X}\mathbf{X}^\intercal|\bar{\mathbf{O}}, s]. \tag{A.45}$$

  3. The expectations given the sequence s are calculated as

$$\mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}, s] = \mathbf{\Lambda}_s^{-1}\bar{\mathbf{W}}_s^\intercal\bar{\mathbf{\Psi}}_s^{-1}(\bar{\mathbf{O}} - \bar{\mu}_s), \tag{A.46}$$

$$\mathbb{E}[\mathbf{X}\mathbf{X}^\intercal|\bar{\mathbf{O}}, s] = \mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}, s]\mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}, s]^\intercal + \mathbf{\Lambda}_s^{-1}, \tag{A.47}$$

     with $\mathbf{\Lambda}_s^{-1} = (\mathbf{I}_{M\text{x}M} + \bar{\mathbf{W}}_s^\intercal\bar{\mathbf{\Psi}}_s^{-1}\bar{\mathbf{W}})^{-1}$.

  4. The variance of the posterior distribution in eq. (A.43) could be computed as

$$\mathbf{\Lambda}^{-1} = \sum_{s=1}^{K^N} p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}})\mathbf{\Lambda}_s^{-1}, \tag{A.48}$$

     but we do not need it for the EM algorithm.

5. Finally, the posteriors of each of the sequences given the data are computed as

$$p(\bar{\mathbf{Z}}_s|\bar{\mathbf{O}}) = \frac{\bar{\omega}_s \mathcal{N}(\bar{\mathbf{O}}; \bar{\mu}_s, \bar{\mathbf{\Sigma}}_s)}{\sum_{r=1}^{K^N} \bar{\omega}_r \mathcal{N}(\bar{\mathbf{O}}; \bar{\mu}_r, \bar{\mathbf{\Sigma}}_r)} = \frac{\bar{\omega}_s \mathcal{N}(\bar{\mathbf{O}}; \bar{\mu}_s, \bar{\mathbf{\Sigma}}_s)}{p(\bar{\mathbf{O}})}, \tag{A.49}$$

6. The formulas to obtain $\mathbf{W}_k$ and means μ_k in the M step are coupled. Hence, we must calculate them simultaneously. For that, we obtain the following expanded expectations,

$$\mathbb{E}[\tilde{\mathbf{X}}|\bar{\mathbf{O}}, s] = \begin{bmatrix} \mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}, s]| \\ 1 \end{bmatrix}, \tag{A.50}$$

$$\mathbb{E}[\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\intercal|\bar{\mathbf{O}}, s] = \begin{bmatrix} \mathbb{E}[\mathbf{X}\mathbf{X}^\intercal|\bar{\mathbf{O}}, s] & \mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}, s] \\ \mathbb{E}[\mathbf{X}|\bar{\mathbf{O}}, s] & 1 \end{bmatrix}. \tag{A.51}$$

This expansion makes the model in eq. (2.28) equivalent to

$$\mathbf{o}_n = \tilde{\mathbf{W}}_k \tilde{\mathbf{x}} + \epsilon_k, \tag{A.52}$$

where $\tilde{\mathbf{W}}_k = \begin{bmatrix} \mathbf{W}_k & \mu_k \end{bmatrix}$ and $\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$.

- M step

1. The first step is to create an indicator matrix $\mathbf{Q}_s$ of size KxN for each sequence s, where the rows are entries for the K Gaussian components, and the columns are entries for the N samples generated by the same hidden variable $\mathbf{X}$. In each column, there will only be one entry set to 1, corresponding to the active component at that time for combination s. Then we accumulate the indicator matrices weighted by the probability of their sequence s,

$$\mathbf{Q} = \sum_{s=1}^{K^N} \mathbf{Q}_s p(s|\bar{\mathbf{O}}). \tag{A.53}$$

2. The rest of steps are made individually for each Gaussian component k. We weight the observation at time n by the accumulated posterior over sequences at that time calculated in the previous step,

$$\mathbf{o}_{qnk} = \mathbf{o}_n \mathbf{Q}[n, k]. \tag{A.54}$$

3. We obtain sum the previous weighted data over time,

$$\mathbf{o}_{qk} = \sum_{n=1}^{N} \mathbf{o}_{qnk}. \tag{A.55}$$

4. We compute four accumulators,

$$\mathbf{A}_k = \mathbf{o}_{qk}\mathbb{E}[\tilde{\mathbf{X}}|\bar{\mathbf{O}}, s]. \tag{A.56}$$

$$\mathbf{B}_k = \sum_{n=1}^{N} Q[n,k]\mathbb{E}[\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\intercal}|\bar{\mathbf{O}}, s] \tag{A.57}$$

$$\mathbf{C}_k = \sum_{n=1}^{N} \mathbf{Q}[n,k]\mathbf{o}_n^2 \tag{A.58}$$

$$D_k = \sum_{n=1}^{N} \mathbf{Q}[n.k] \tag{A.59}$$

5. Up to now we have considered only one file, and our four accumulators only contain results for that file. If we have J files for training, the E step would be computed individually for each file, whereas in the M step we would compute four accumulators for each file, and the final accumulators would be the sum of the accumulators of all the files. Then, if the accumulators for file j are $\mathbf{A}_k^j$, $\mathbf{B}_k^j$, $\mathbf{C}_k^j$, and D_k^j, our final accumulators would be

$$\mathbf{A}_k = \sum_{j=1}^{J} \mathbf{A}_k^j, \tag{A.60}$$

$$\mathbf{B}_k = \sum_{j=1}^{J} \mathbf{B}_k^j, \tag{A.61}$$

$$\mathbf{C}_k = \sum_{j=1}^{J} \mathbf{C}_k^j, \tag{A.62}$$

$$D_k = \sum_{j=1}^{J} D_k^j. \tag{A.63}$$

6. Finally, the joint update of the matrix subspace $\mathbf{W}_k$ and the mean μ_k is

$$\tilde{\mathbf{W}}_k = \mathbf{A}_k\mathbf{B}_k^{-1}. \tag{A.64}$$

7. The update of the covariance matrix of the noise $\mathbf{\Psi}_k$ is

$$\mathbf{\Psi}_k = (\mathbf{C}_k - \text{diag}(\tilde{\mathbf{W}}_k \mathbf{A}_k^{-1}))/D_k. \tag{A.65}$$

8. And the update of the weights is

$$\omega_k = \frac{D_k}{N}. \tag{A.66}$$

In case of having J files, we would have $\omega_k = \frac{D_k}{\sum_{j=1}^{J} N_j}$, where the number of samples of file j is N_j.

TMFA has the potential to include different hidden variables to model simultaneously different aspects of the speech, like the speaker and the channel, as it happens in the field of speaker recognition. The main change occurs in the structure of the covariance matrix, which must include the correlations among the observed variables captured by each hidden variable. Then, the supervector of observations $\bar{\mathbf{O}}$ will include all data points that are affected by those hidden variables. For example, if we model the speaker, all the files of the same speaker will be concatenated in a single vector. If simultaneously we model the channel, the covariance matrix will reflect how the different observations are correlated taking into account that we have one speaker and several files of the same speaker. In this Thesis we are focused only in channel compensation, so we will not explore the case with more than one hidden variable further, and all our computations will be with only one. More information of this scenario with multiple hidden variables can be seen in Miguel et al. [2014].

A.8 Joint Factor Analysis

In this Section we detail the EM algorithm for JFA.

A.8.1 EM for JFA

JFA model parameters are estimated iteratively with the EM algorithm, which is solved differently to TMFA, due to the approximations explained before. First, we calculate the probability of the supervector of observations in sequence, $p(\bar{\mathbf{O}}|\mathbf{X})$, using eq. (B.2). Note that given $\mathbf{X}$, the dependency among observations is broken, and $\ln p(\bar{\mathbf{O}}|\mathbf{X}) = \ln p(\mathbf{O}|\mathbf{X})$. Then, by setting the q function in equation B.2 to the true

posterior distribution of the hidden variables, in this case to the true frame alignment that generated each frame or true responsibilities of each Gaussian, the KL divergence vanishes, and we can express the conditional log-likelihood, $\ln p(\mathbf{O}|\mathbf{X})$, according to eq. (B.3),

$$\begin{aligned}
\ln p(\mathbf{O}|\mathbf{X}) &= \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \omega_k p(\mathbf{o}_n|z_k, \mathbf{X}) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} p(z_k|\mathbf{o}_n, \mathbf{X}) \ln p(\mathbf{o}_n, z_k|\mathbf{X}) - \sum_{n=1}^{N}\sum_{k=1}^{K} p(z_k|\mathbf{o}_n, \mathbf{X}) \ln p(z_k|\mathbf{o}_n, \mathbf{X}) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} p(z_k|\mathbf{o}_n) \ln p(\mathbf{o}_n|z_k, \mathbf{X}) - \sum_{n=1}^{N}\sum_{k=1}^{K} p(z_k|\mathbf{o}_n) \ln \frac{p(z_k|\mathbf{o}_n)}{p(z_k)} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_n(z_k) \ln p(\mathbf{o}_n|z_k, \mathbf{X}) + \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_n(z_k) \ln \frac{\gamma_n(z_k)}{\omega_k} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_n(z_k) \ln p(\mathbf{o}_n|z_k, \mathbf{X}) + const_1,
\end{aligned} \tag{A.67}$$

where the constant reflects the terms independent of $\mathbf{O}$, and the file index, j, has been omitted for clarity. However, it is important to see that we do not really use the true responsibilities, but the fixed alignments given by the UBM. Note that they are independent of $\mathbf{X}$. This makes the KL divergence greater than 0 and $\ln p(\mathbf{O}|\mathbf{X})$ will therefore be an approximation to the true conditional log-likelihood. Actually, it will be a lower bound because the KL divergence is always non-negative. Nevertheless, if alignments are accurate enough, as it is normally the case, this approximation is good enough to obtain good results.

Now, we will define the EM auxiliary $\mathcal{Q}$ function for JFA, according to eq. (B.6), with variables $\mathbf{Z} = \mathbf{X}$, $\mathbf{X} = \mathbf{O}$, and $\Theta_l = \{\omega_l, \mathbf{t}_l, \mathbf{\Sigma}_l, \mathbf{U}_l\}$. Note the dependency on l, which is the class for which the parameters are calculated. In our work, JFA is used for LID, and our classes are languages. In practice, the means, $\mathbf{t}_l$, are obtained with relevance MAP and they are not adapted. Also the weights and the covariances are normally kept fixed and equal to the UBM weights and covariances, and in case they are adapted, they are initialized to the UBM values. Normally, a single $\mathbf{U}$ is trained for all languages, but we will start with the general case, where all the hyperparameters are adapted and there is a different $\mathbf{U}$ for each language. Particularities will be seen

later. With j being the index of the file, N_j being the number of frames in file j, k being the Gaussian component index, and n the index of the frame, $\mathcal{Q}_l$ for language l is

$$
\begin{aligned}
\mathcal{Q}_l(\Theta, \Theta^{old}) &= \sum_Z p(\mathbf{Z}|\mathbf{X}, \Theta^{old}) \ln p(\mathbf{X}, \mathbf{Z}|\Theta) \\
&= \int p(\mathbf{X}|\mathbf{O}, \Theta^{old}) \ln p(\mathbf{O}, \mathbf{X}|\Theta) d\mathbf{X} = \sum_{j \in l} \int q_j(\mathbf{X}) \ln p(\mathbf{O}(j), \mathbf{x}) d\mathbf{x} \\
= \sum_{j \in l} \int q_j(\mathbf{X}) \ln\{p(\mathbf{O}(j)|\mathbf{x}) p(\mathbf{x})\} d\mathbf{x} &= \sum_{j \in l} \int q_j(\mathbf{X}) \{\ln p(\mathbf{O}(j)|\mathbf{x}) + \ln p(\mathbf{x})\} d\mathbf{x} \\
= \sum_{j \in l} \int q_j(\mathbf{X}) \{\sum_{n=1}^{N_j} \sum_{k=1}^{K} \gamma_n^j(z_k) \ln \mathcal{N}(\mathbf{o}_n(j); \mathbf{t}_{lk} + \mathbf{U}_{lk}\mathbf{x}, \mathbf{\Sigma}_{lk}) + \ln \mathcal{N}(\mathbf{x}; 0, I)\} d\mathbf{x} & \\
= \sum_{j \in l} \sum_{k=1}^{K} N_k(j) \ln |\mathbf{\Sigma}_{lk}|^{-\frac{1}{2}} - \frac{1}{2} \operatorname{Tr}(\sum_{k=1}^{K} \mathbf{S}_k(j) \mathbf{\Sigma}_{lk}^{-1}) & \\
+ \frac{1}{2} \sum_{k=1}^{K} \mathbf{F}_k(j)^\intercal \mathbf{\Sigma}_{lk}^{-1} \mathbf{t}_{lk} + \frac{1}{2} \operatorname{Tr}(\mathbb{E}_X[\mathbf{x}_j] \sum_{k=1}^{K} \mathbf{F}_k(j)^\intercal \mathbf{\Sigma}_{lk}^{-1} \mathbf{U}_{lk}) & \\
+ \frac{1}{2} \sum_{k=1}^{K} \mathbf{t}_{lk} \mathbf{\Sigma}_{lk}^{-1} \mathbf{F}_k(j) + \frac{1}{2} \operatorname{Tr}(\mathbb{E}_X[\mathbf{x}_j] \sum_{k=1}^{K} \mathbf{U}_{lk} \mathbf{\Sigma}_{lk}^{-1} \mathbf{F}_k(j)) & \\
- \frac{1}{2} \sum_{k=1}^{K} N_k(j) \mathbf{t}_{lk}^\intercal \mathbf{\Sigma}_{lk}^{-1} \mathbf{t}_{lk} - \frac{1}{2} \operatorname{Tr}(\mathbb{E}_X[\mathbf{x}_j] \sum_{k=1}^{K} N_k(j) \mathbf{t}_{lk}^\intercal \mathbf{\Sigma}_{lk}^{-1} \mathbf{U}_{lk}) & \\
- \frac{1}{2} \operatorname{Tr}(\mathbb{E}_X[\mathbf{x}_j]^\intercal \sum_{k=1}^{K} N_k(j) \mathbf{U}_{lk}^\intercal \mathbf{\Sigma}_{lk}^{-1} \mathbf{t}_{lk}) - \frac{1}{2} \operatorname{Tr}(\mathbb{E}_X[\mathbf{x}_j \mathbf{x}_j^\intercal] \sum_{k=1}^{K} N_k(j) \mathbf{U}_{lk}^\intercal \mathbf{\Sigma}_{lk}^{-1} \mathbf{U}_{lk}) & \\
- \frac{1}{2} \operatorname{Tr}(\mathbb{E}_X[\mathbf{x}_j \mathbf{x}_j^\intercal]) + const_2. &
\end{aligned}
\tag{A.68}
$$

In the E step we calculate the posterior distribution of the hidden variable given the observation file j in language l, $p(\mathbf{X}|\mathbf{O})$,

$$\ln p(\mathbf{X}|\mathbf{O}) = \ln p(\mathbf{O},\mathbf{X}) - \ln p(\mathbf{O})$$
$$\ln p(\mathbf{O}(j),\mathbf{x}) - \ln p(\mathbf{O}(j)) = \ln p(\mathbf{O}(j)|\mathbf{x})p(\mathbf{x}) + const_3$$

$$= \sum_{n=1}^{N_j}\sum_{k=1}^{K}\gamma_n^j(z_k)\ln p(\mathbf{O}(j)|k,\mathbf{x}) + \ln p(\mathbf{x}) + const_3$$
$$= \sum_{n=1}^{N_j}\sum_{k=1}^{K}\gamma_n^j(z_k)\ln \mathcal{N}(\mathbf{o}_n(j);\mathbf{t}_{lk}+\mathbf{U}_{lk}\mathbf{x},\mathbf{\Sigma}_{lk}) + \ln\mathcal{N}(\mathbf{x};0,I) + const_3$$
$$= \sum_{k=1}^{K}\frac{1}{2}\mathbf{F}_k(j)^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbf{x} + \sum_{k=1}^{K}\frac{1}{2}\mathbf{x}^\intercal\mathbf{U}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{F}_k(j)$$
$$-\sum_{k=1}^{K}\frac{1}{2}N_k(j)\mathbf{x}^\intercal\mathbf{U}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{t}_{lk} - \sum_{k=1}^{K}\frac{1}{2}N_k(j)\mathbf{t}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbf{x} \tag{A.69}$$
$$-\sum_{k=1}^{K}\frac{1}{2}(\mathbf{U}_{lk}\mathbf{x})^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbf{x} - \frac{1}{2}\mathbf{x}^\intercal\mathbf{x} + const_4$$
$$= \sum_{k=1}^{K}\{\mathbf{x}^\intercal\mathbf{U}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{F}_k(j) - N_k(j)\mathbf{x}^\intercal\mathbf{U}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{t}_{lk}$$
$$-\frac{1}{2}(\mathbf{U}_{lk}\mathbf{x})^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbf{x}\} - \frac{1}{2}\mathbf{x}^\intercal\mathbf{x} + const_4$$
$$= \mathbf{x}^\intercal\{\sum_{k=1}^{K}\mathbf{U}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}(\mathbf{F}_k(j) - N_k(j)\mathbf{t}_{lk})\}$$
$$-\frac{1}{2}\mathbf{x}^\intercal\{\sum_{k=1}^{K}(N_k(j)\mathbf{U}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}) + I\}\mathbf{x} + const_4,$$

where we can identify a Gaussian. and

$$p(\mathbf{X}|\mathbf{O}) = \mathcal{N}(\mathbb{E}_X[\mathbf{x}],\mathbf{L}^{-1}), \tag{A.70}$$

which for utterance j has the following form,

$$\mathbb{E}_X[\mathbf{x}_j] = \mathbf{L}_j^{-1}\sum_{k=1}^{K}\{\mathbf{U}_{lk}^\intercal\mathbf{\Sigma}_{lk}^{-1}(\mathbf{F}_k(j) - N_k(j)\mathbf{t}_{lk})\}, \tag{A.71}$$

with symbol E_X referring to the expectation taken with respect to the latent variable

$\mathbf{X}$,

$$\mathbf{L}_j = \sum_{k=1}^{K} (N_k(j)\mathbf{U}_{lk}^{\intercal}\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}) + \mathbf{I}, \tag{A.72}$$

and we define

$$\mathbb{E}_X[\mathbf{x}_j\mathbf{x}_j^{\intercal}] = \mathbf{L}_j^{-1} + \mathbb{E}_X[\mathbf{x}_j]\mathbb{E}_X[\mathbf{x}_j]^{\intercal}. \tag{A.73}$$

In the M step, the $\mathcal{Q}$ function is derived with respect to the parameters, Θ_l, and made equal to zero. Then, we obtain an expression for each of the parameters. The weights are kept constant and equal to the UBM weights. The means, $\mathbf{t}_{lk}$, are adapted as

$$\frac{\partial}{\partial \mathbf{t}_{lk}}\mathcal{Q}(\mathbf{t}_{lk}, \mathbf{t}_{lk}^{old}) = \sum_{j\in l} \mathbf{\Sigma}_{lk}^{-1}\mathbf{F}_k(j) - N_k(j)\mathbf{\Sigma}_{lk}^{-1}\mathbf{t}_{lk} - N_k(j)\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbb{E}_X[\mathbf{x}_j] = 0, \tag{A.74}$$

$$\mathbf{t}_{lk} = \frac{1}{J_l}\sum_{j=1}^{J_l}(\frac{\mathbf{F}_k(j)}{N_k(j)} - \mathbf{U}_{lk}\mathbb{E}_X[\mathbf{x}_j]), \tag{A.75}$$

being J_l the number of utterances of language l. Nonetheless, the means are rarely adapted. The update of the subspace matrix $\mathbf{U}_{lk}$ is

$$\begin{aligned}\frac{\partial}{\partial \mathbf{U}_{lk}}\mathcal{Q}(\mathbf{U}_{lk}, \mathbf{U}_{lk}^{old}) = \\ \sum_{j\in l} \mathbf{\Sigma}_{lk}^{-1}\mathbf{F}_k(j)\mathbb{E}_X[\mathbf{x}_j]^{\intercal} - N_k(j)\mathbf{\Sigma}_{lk}^{-1}\mathbf{t}_{lk}\mathbb{E}_X[\mathbf{x}_j]^{\intercal} - N_k(j)\mathbf{U}_{lk}\mathbb{E}_X[\mathbf{x}_j\mathbf{x}_j^{\intercal}]^{\intercal} = 0\end{aligned} \tag{A.76}$$

$$\mathbf{U}_{lk} = \mathbf{C}_{lk}\mathbf{A}_{lk}^{-1}, \tag{A.77}$$

where

$$\mathbf{C}_{lk} = \sum_{j\in l} (\mathbf{F}_k(j) - N_k(j)\mathbf{t}_{lk})\mathbb{E}_X[\mathbf{x}_j]^{\intercal}, \tag{A.78}$$

$$\mathbf{A}_{lk} = \sum_{j\in l} \mathbb{E}_X[\mathbf{x}_j\mathbf{x}_j^{\intercal}]N_k(j). \tag{A.79}$$

An alternative commonly used is to estimate a unique $\mathbf{U}_k$ for all languages. The update formula is a weighted average of the $\mathbf{U}_{lk}$ of the individual languages

$$\mathbf{U}_k = (\sum_l \mathbf{C}_{lk})(\sum_l \mathbf{A}_{lk})^{-1}. \tag{A.80}$$

Finally, we obtain the following update formula for $\mathbf{\Sigma}_{lk}$, which in practice is seldom adapted

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{\Sigma}_{lk}} \mathcal{Q}(\mathbf{\Sigma}_{lk}, \mathbf{\Sigma}_{lk}^{old}) \qquad \text{(A.81)}\\
&= -\frac{1}{2}\sum_{j\in l} N_k(j)\mathbf{\Sigma}_{lk}^{-1} + \sum_{j\in l}\{\mathbf{\Sigma}_{lk}^{-1}\mathbf{S}_k(j)\mathbf{\Sigma}_{lk}^{-1} - \frac{1}{2}I \circ (\mathbf{\Sigma}_{lk}^{-1}\mathbf{S}_k(j)\mathbf{\Sigma}_{lk}^{-1})\}\\
&\quad -\frac{1}{2}\sum_{j\in l}\mathbf{\Sigma}_{lk}^{-1}\mathbf{F}_k(j)\mathbf{t}_{lk}^{\intercal}\mathbf{\Sigma}_{lk}^{-1} - \frac{1}{2}\sum_{j\in l}\mathbf{\Sigma}_{lk}^{-1}\mathbf{F}_k(j)\mathbb{E}_X[\mathbf{x}_j]^{\intercal}\mathbf{U}_{lk}^{\intercal}\mathbf{\Sigma}_{lk}^{-1}\\
&\quad -\frac{1}{2}\sum_{j\in l}\mathbf{\Sigma}_{lk}^{-1}\mathbf{t}_{lk}\mathbf{F}_k(j)^{\intercal}\mathbf{\Sigma}_{lk}^{-1} - \frac{1}{2}\sum_{j\in l}\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbb{E}_X[\mathbf{x}_j]\mathbf{F}_k(j)^{\intercal}\mathbf{\Sigma}_{lk}^{-1}\\
&\quad +\frac{1}{2}\sum_{j\in l} N_k(j)\mathbf{\Sigma}_{lk}^{-1}\mathbf{t}_{lk}\mathbf{t}_{lk}^{\intercal}\mathbf{\Sigma}_{lk}^{-1} + \frac{1}{2}\sum_{j\in l} N_k(j)\mathbf{\Sigma}_{lk}^{-1}\mathbf{t}_{lk}\mathbb{E}_X[\mathbf{x}_j]^{\intercal}\mathbf{U}_{lk}^{\intercal}\mathbf{\Sigma}_{lk}^{-1}\\
&+\frac{1}{2}\sum_{j\in l} N_k(j)\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbb{E}_X[\mathbf{x}_j]\mathbf{t}_{lk}^{\intercal}\mathbf{\Sigma}_{lk}^{-1} + \frac{1}{2}\sum_{j\in l} N_k(j)\mathbf{\Sigma}_{lk}^{-1}\mathbf{U}_{lk}\mathbb{E}_X[\mathbf{x}_j\mathbf{x}_j^{\intercal}]\mathbf{U}_{lk}^{\intercal}\mathbf{\Sigma}_{lk}^{-1} = 0,
\end{aligned}$$

where the symbol $\circ$ means the *Hadamard* or entry-wise product, and we have applied the following identities

$$\frac{\partial}{\partial \mathbf{C}} \ln|\mathbf{C}| = (\mathbf{C}^{-1})^{\intercal} \qquad \text{(A.81)}$$

$$\frac{\partial}{\partial \mathbf{C}} \mathrm{Tr}(\mathbf{C}^{-1}\mathbb{E}[\mathbf{x}\mathbf{x}^{\intercal}]) = -2\mathbf{C}^{-1}\mathbb{E}[\mathbf{x}\mathbf{x}^{\intercal}]\mathbf{C}^{-1} + I \circ (\mathbf{C}^{-1}\mathbb{E}[\mathbf{x}\mathbf{x}^{\intercal}]\mathbf{C}^{-1}) \qquad \text{(A.82)}$$

$$\frac{\partial}{\partial \mathbf{C}} tr(\mathbf{A}\mathbf{C}^{-1}\mathbf{B}) = -(\mathbf{C}^{-1}\mathbf{B}\mathbf{A}\mathbf{C}^{-1})^{\intercal} = -\mathbf{C}^{-\intercal}\mathbf{A}^{\intercal}\mathbf{B}^{\intercal}\mathbf{C}^{-\intercal}. \qquad \text{(A.83)}$$

Then, the update formula for $\mathbf{\Sigma}_{lk}$ is

$$\begin{aligned}
\mathbf{\Sigma}_{lk} = \frac{1}{J_l}\sum_{j\in l} &\frac{\mathbf{S}_k(j) - \mathbf{F}_k(j)(\mathbf{t}_{lk}^{\intercal} + \mathbb{E}_X[\mathbf{x}_j]^{\intercal}\mathbf{U}_{lk}^{\intercal}) - (\mathbf{t}_{lk} + \mathbf{U}_{lk}\mathbb{E}_X[\mathbf{x}_j])\mathbf{F}_k(j)^{\intercal}}{N_k(j)}\\
&+\mathbf{t}_{lk}(\mathbf{t}_{lk}^{\intercal} + \mathbb{E}_X[\mathbf{x}_j]^{\intercal}\mathbf{U}_{lk}^{\intercal}) + \mathbf{U}_{lk}(\mathbb{E}_X[\mathbf{x}_j]\mathbf{t}_{lk}^{\intercal} + \mathbb{E}_X[\mathbf{x}_j\mathbf{x}_j^{\intercal}]\mathbf{U}_{lk}^{\intercal}),
\end{aligned} \qquad \text{(A.84)}$$

and we have assumed that the off-diagonal terms of $\mathbf{S}_k(j)\Sigma_{lk}^{-1}$ are much smaller than the diagonal terms, and the approximation

$$I \circ (\mathbf{\Sigma}_{lk}^{-1}\mathbf{S}_k(j)\mathbf{\Sigma}_{lk}^{-1}) \approx \mathbf{\Sigma}_{lk}^{-1}\mathbf{S}_k(j)\mathbf{\Sigma}_{lk}^{-1} \qquad \text{(A.85)}$$

is applied. In practice, we use the UBM covariance matrices, $\mathbf{\Sigma}_k$, for all the languages.

A.8.2 EM with Minimum Divergence for JFA

MD is an alternative way of maximizing the lower bound of the EM algorithm [Kenny, 2006; Brümmer, 2009b,a, 2010]. The lower bound can be expressed as the difference between the expectation of the logarithm of the probability of the observations given the hidden variable, and the KL distance between the posterior distribution of the hidden variables given the observations, and the prior distribution of the hidden variables. Then, we can differentiate two steps. A fist one to maximize the first term, and a second one to minimize the divergence. The minimization gives name to the algorithm, and makes the posterior and prior distributions of the hidden variables match as much as possible. The final operation of the algorithm is to transform the model obtained in the minimization step, with arbitrary prior distribution over the hidden variable, in such a way that the prior distribution becomes the originally assumed in the maximization step, which is normally standard Gaussian. Let's see the formulation of the algorithm.

The EM with MD is another approach to implement the EM algorithm. It is derived by expressing equation B.3 as

$$\begin{aligned}
\mathcal{L}(q,\Theta) = \sum_{\mathbf{X}} q(\mathbf{X}) \ln \frac{p(\mathbf{O},\mathbf{X}|\Theta)}{q(\mathbf{X})} &= \sum_{\mathbf{X}} q(\mathbf{X}) \ln \frac{p(\mathbf{O}|\mathbf{X},\Theta_1)p(\mathbf{X}|\Theta_2)}{q(\mathbf{X})} \\
&= \sum_{\mathbf{X}} q(\mathbf{X}) \ln p(\mathbf{O}|\mathbf{X},\Theta_1) - \sum_{\mathbf{X}} q(\mathbf{X}) \ln \frac{q(\mathbf{X})}{p(\mathbf{X}|\Theta_2)} \\
&= \mathbb{E}_{q(\mathbf{X})}[\ln p(\mathbf{O}|\mathbf{X},\Theta_1)] - D_{KL}(q(\mathbf{X})||p(\mathbf{X}|\Theta_2)) \\
&= \mathbb{E}_{p(\mathbf{X}|\mathbf{O},\Theta^{old})}[\ln p(\mathbf{O}|\mathbf{X},\Theta_1)] - D_{KL}(p(\mathbf{X}|\mathbf{O},\Theta^{old})||p(\mathbf{X}|\Theta_1)) \\
&= \mathbb{E}_{p(\mathbf{X}|\mathbf{O},\Theta^{old})}[\ln p(\mathbf{O}|\mathbf{X},\Theta_2)] - D_{KL}(p(\mathbf{X}|\mathbf{O},\Theta^{old})||p(\mathbf{X}|\Theta_2)),
\end{aligned} \tag{A.86}$$

where Θ_1 and Θ_2 are two disjoint subset of parameters, and the model is said to be *overparametrized* [Brümmer, 2009b], because $\Theta_1 \equiv \Theta 2$, and $\mathcal{L}(\Theta_1) = \mathcal{L}(\Theta_2)$, that is, there exists redundant parametrizations which are equivalent. In general, two models are equivalent if

$$p(\mathbf{O}|\Theta_1) = p(\mathbf{O}|\Theta_2). \tag{A.87}$$

In our case, $\Theta_{l1} = \{\mathbf{t}_{l1}, \mathbf{\Sigma}_{l1}, \mathbf{U}_{l1}, \Pi_1\}$, where $\Pi_1 = \{\mu_{X_1} = 0, \mathbf{\Sigma}_{X_1} = I\}$ are the hyperparameters of the prior distribution $p(\mathbf{X}|\Pi_1)$, and $\Theta_{l2} = \{\mathbf{t}_{l2}, \mathbf{\Sigma}_{l2}, \mathbf{U}_{l2}, \Pi_2\}$, where $\Pi_2 = \{\mu_{X_2}, \mathbf{\Sigma}_{X_2}\}$ is also updated. Hence we have a transformation of the distribution $p(\mathbf{X}_2|\Theta_2)$ to obtain $p(\mathbf{X}_1|\Theta_1)$, and the variables $\mathbf{X}_1$ with standard normal distribution

and $\mathbf{X}_2$ are related as

$$\mathbf{X}_1 = \phi(\mathbf{X}_2) = P^{-1}(\mathbf{X}_2 - \mu_{X_2}), \tag{A.88}$$

or equivalently

$$\mathbf{X}_2 = \phi^{-1}(\mathbf{X}_1) = P(\mathbf{X}_1 + \mu_{X_2}), \tag{A.89}$$

where $\mathbf{\Sigma}_{X_2} = \mathbf{P}\mathbf{P}^\intercal$, that is, $\mathbf{X}_2$ is normalized by its mean and covariance's square root to obtain $\mathbf{X}_1$. According to the fundamental theorem of Calculus and to the chain rule, their probability density functions are related as

$$f_{X_1|\Pi_1}(\mathbf{X}_1) = f_{X_2|\Pi_2}(\phi^{-1}(\mathbf{X}_1))\frac{\partial\phi^{-1}(\mathbf{X}_1)}{\partial\mathbf{X}_1} = p_{X_2}(\phi^{-1}(\mathbf{X}_1|\Pi_2))\mathbf{P}. \tag{A.90}$$

This is one of the two sufficient conditions for equivalence. The other is

$$p(\mathbf{O}|\mathbf{X}_1, \mathbf{U}_1) = p(\mathbf{O}|\phi^{-1}(\mathbf{X}_1), \mathbf{U}_2), \tag{A.91}$$

which follows from expanding equation A.87, as shown in Brümmer [2009b].

This type of EM has 2 alternated M steps:

- Log-likelihood maximization of $\mathbb{E}_{p(\mathbf{X}|\mathbf{O},\Theta_l^{old})}[\ln p(\mathbf{O}|\mathbf{X}, \Theta_{l1})]$. In this case, the update formulas of the parameters are the same as in equations A.75, A.77, A.80, and A.84. The reason why they are the same is that the only difference between $\mathbb{E}_{p(\mathbf{X}|\mathbf{O},\Theta_l^{old})}[\ln p(\mathbf{O}|\mathbf{X}, \Theta_{l1})]$ and equation A.68, which is the one that we maximized before, is the term $\sum_{\mathbf{X}} q(\mathbf{X}) \ln p(\mathbf{X})$, which is present in the second but not in the first. However, when deriving these functions with regard to the parameters, this term does not affect, because it does not depend on the parameters Θ_{l1}, and the update equations are the same.

- Minimization of $D_{KL}(p(\mathbf{X}|\mathbf{O}, \Theta_l^{old})||p(\mathbf{X}|\Theta_{l2}))$. This step can be divided in another two:

  1. The minimization itself is carried out with respect to $\Pi_2 = \{\mu_{X_2}, \mathbf{\Sigma}_{X_2}\}$, because we let the prior distribution $p(\mathbf{X})$ to be a non-standard Gaussian. Then, $p(\mathbf{X}) = \mathcal{N}(\mu_X, \mathbf{\Sigma}_X)$. And $p(\mathbf{X}|\mathbf{O}, \Theta_l^{old})$ was defined in A.70. Then, the KL divergence between two Gaussians is defined as

$$\begin{aligned} & D_{\mathrm{KL}}(\mathcal{N}_0\|\mathcal{N}_1) \\ = \frac{1}{2}(\mathrm{Tr}\left(\mathbf{\Sigma}_1^{-1}\mathbf{\Sigma}_0\right) + (\mu_1 - \mu_0)^\intercal \mathbf{\Sigma}_1^{-1}(\mu_1 - \mu_0) & \\ & -D - \ln\left(\frac{|\mathbf{\Sigma}_0|}{|\mathbf{\Sigma}_1|}\right)), \end{aligned} \tag{A.92}$$

with D the feature dimensionality. Substituting $\mathcal{N}_0$ by $p(\mathbf{X}|\mathbf{O}, \Theta_l^{old})$, $\mathcal{N}_1$ by $p(\mathbf{X}|\Pi_2)$, and considering a total of J_l training files, we reach the following expression for the divergence

$$\begin{aligned} & D_{KL}(p(\mathbf{X}|\mathbf{O}, \Theta_l^{old})||p(\mathbf{X}|\Theta_{l2})) \\ = \sum_{j=1}^{J_l} \frac{1}{2} (\mathrm{Tr}\left(\mathbf{\Sigma}_X^{-1}\mathbf{L}_j^{-1}\right) + (\mu_X - \mathbb{E}_X[\mathbf{x}_j])^\intercal \mathbf{\Sigma}_X \mathbf{X}^{-1} (\mu_X - \mathbb{E}_X[\mathbf{x}_j]) \\ & -D - \ln\left(\frac{|\mathbf{L}^{-1}|}{|\mathbf{\Sigma}_X|}\right)). \end{aligned} \tag{A.93}$$

Deriving with respect to μ_X and making the derivative equal to zero, we obtain an expression for μ_X

$$\frac{\partial}{\partial \mu_X} D_{KL} = J_l 2\mu_x^\intercal \mathbf{\Sigma}_X^{-1} - 2\mathbf{\Sigma}_X^{-1} \sum_{j=1}^{J_l} \mathbb{E}_X[\mathbf{x}_j] = 0 \tag{A.94}$$

$$\mu_X = \frac{1}{J_l} \sum_{j=1}^{J_l} \mathbb{E}_X[\mathbf{x}_j] \tag{A.95}$$

Deriving with respect to $\mathbf{\Sigma}_X$ and making the derivative equal to zero, we obtain an expression for $\mathbf{\Sigma}_X$

$$\begin{aligned} \frac{\partial}{\partial \mathbf{\Sigma}_X} D_{KL} = & \sum_{j=1}^{J_l} -2\mathbf{\Sigma}_X^{-1}\mathbf{L}_j^{-1}\mathbf{\Sigma}_X^{-1} + \mathbf{I} \circ (\mathbf{\Sigma}_X^{-1}\mathbf{L}_j^{-1}\mathbf{\Sigma}_X^{-1}) \\ & -\mathbf{\Sigma}_X^{-1}(\mu_X - \mathbb{E}_X[\mathbf{x}_j])(\mu_X - \mathbb{E}_X[\mathbf{x}_j])^\intercal \mathbf{\Sigma}_X^{-1} + \mathbf{\Sigma}_X^{-1} \\ \approx \sum_{j=1}^{J_l} & -\mathbf{\Sigma}_X^{-1}\mathbf{L}_j^{-1}\mathbf{\Sigma}_X^{-1} - \mathbf{\Sigma}_X^{-1}(\mu_X - \mathbb{E}_X[\mathbf{x}_j])(\mu_X - \mathbb{E}_X[\mathbf{x}_j])^\intercal \mathbf{\Sigma}_X^{-1} + \mathbf{\Sigma}_X^{-1} = 0, \end{aligned} \tag{A.96}$$

where it is assumed that the off-diagonal terms of $\mathbf{\Sigma}_X^{-1}\mathbf{L}_j^{-1}$ are much smaller than the diagonal terms and the approximation

$$\mathbf{I} \circ (\mathbf{\Sigma}_X^{-1}\mathbf{L}_j^{-1}\mathbf{\Sigma}_X^{-1}) \approx \mathbf{\Sigma}_X^{-1}\mathbf{L}_j^{-1}\mathbf{\Sigma}_X^{-1} \tag{A.97}$$

is valid. Finally

$$\mathbf{\Sigma}_X = \sum_{j=1}^{J_l} \mathbf{L}_j^{-1} + (\mu_X - \mathbb{E}_X[\mathbf{x}_j])(\mu_X - \mathbb{E}_X[\mathbf{x}_j])^\intercal \tag{A.98}$$

2. At this point we have to obtain the equivalent model, because in the next iteration we will work again over Θ_{l1}. Therefore, the models with and without standard normal prior must be equivalent. As $p(\mathbf{X}_1) = \mathcal{N}(\mathbf{X}_1; \mathbf{0}, \mathbf{I})$ and $p(\mathbf{X}_2) = \mathcal{N}(\mathbf{X}_2; \mu_{X_2}, \mathbf{\Sigma}_{X_2})$,

$$\begin{aligned} \mathbf{t}_{l1} + \mathbf{U}_{l1}\mathbf{x}_1 = \mathbf{t}_{l2} + \mathbf{U}_{l2}\mathbf{x}_2 = \mathbf{t}_{l2} + \mathbf{U}_{l2}\phi^{-1}(\mathbf{x}_1) \\ = \mathbf{t}_{l2} + \mathbf{U}_{l2}(\mathbf{P}\mathbf{x}_1 + \mu_{X_2}) = (\mathbf{U}_{l2}\mu_{X_2} + \mathbf{t}_{l2}) + \mathbf{U}_{l2}\mathbf{P}\mathbf{x}_1 \end{aligned} \tag{A.99}$$

where, in this case, $\Pi^{old} = \{\mu_{X_2}, \mathbf{U}_{l2}\}$ are the hyperparameters from previous iteration. The update equation for $\mathbf{U}_l$ is

$$\mathbf{U}_l = \mathbf{U}_{l1} = \mathbf{U}_{l2}\mathbf{P}. \tag{A.100}$$

In practice, a single $\mathbf{U}$ is computed for all languages, as the average of all the $\mathbf{U}_l$, as in the case of the ML approach seen in eq. (A.80). The update equation for $\mathbf{t}_l$ is

$$\mathbf{t}_l = \mathbf{t}_{l1} = \mathbf{t}_{l2} + \mathbf{U}_{l2}\mu_{X_2}, \tag{A.101}$$

but in our experiments the means $\mathbf{t}_l$ are not updated.

In summary, MD minimizes the divergence between the posterior distribution of the hidden variables and its prior distribution, and given that in this minimization process the hyperparameters of the prior distribution are modified, the model parameters are transformed in such a way that the initial prior distribution of the hidden variables is conserved. At the end of the KL minimization, we let the prior to be non-standard Gaussian, even when our assumption at the beginning was that it was normal. Later, we transform the model parameters to recover an equivalent model with standard prior. In this way of deriving the EM algorithm, we alternate between an ML and an MD step successively. In a general case, we could have integrated from the beginning the mean and covariance of the prior also in the ML step.

MD assures that the log-likelihood of the current iteration will increase unless we are in a maximum. However, it does not give higher global log-likelihood than EM only with the ML step, but as shown in Brümmer [2009b] the convergence is faster. In addition, some authors assure that EM without MD is more vulnerable to getting stuck in saddle-points and MD helps to avoid this [Kenny, 2006; Brümmer, 2009b].

A.9 Total Variability Subspace: *i-Vectors*

In this Section we review the EM algorithm for training the *i-Vector* subspace.

A.9.1 EM Algorithm for *i-Vectors*

The training procedure of JFA as feature extractor is the same as the one explained in section 2.11, with the following differences:

- the model is language-independent. As we can see in eq. (2.50), there is no MAP adaptation to the language and the model is centered at the UBM. The subspace is spanned by matrix $\mathbf{T}$, and the point in the subspace of utterance j is given by the latent variable $\mathbf{i}(j)$ (we just renamed $\mathbf{U}$ and $\mathbf{x}(j)$ of JFA, which were defined for the channel subspace).

- the same EM procedure defined for JFA can be used, substituting $\mathbf{M}_l(j)$ by $\mathbf{M}(j)$. In practice, we pool the files of all the languages and calculate a single JFA model.

In the E step we identify that the posterior distribution of the latent variable $\mathbf{i}$ is

$$p(\mathbf{i}|\mathbf{O}) = \mathcal{N}(\mathbb{E}_I[\mathbf{i}], \mathbf{L}^{-1}), \tag{A.102}$$

with the following expression for utterance j,

$$\mathbb{E}_I[\mathbf{i}_j] = \mathbf{L}_j^{-1} \sum_{k=1}^{K} \{\mathbf{T}_k^\intercal \mathbf{\Sigma}_k^{-1} (\mathbf{F}_k(j) - N_k(j)\mu)\} \tag{A.103}$$

being the MAP point estimate of the total variability factor $\mathbf{i}$ of utterance j, which is known as *i-Vector*. Note that μ, $\mathbf{T}$, and $\mathbf{\Sigma}$, are independent of the language. And

$$\mathbf{L}_j = \sum_{k=1}^{K} (N_k(j)\mathbf{T}_k^\intercal \mathbf{\Sigma}_k^{-1} \mathbf{T}_k) + \mathbf{I}. \tag{A.104}$$

In the M step, the update equations become

- for the means

$$\mu = \frac{1}{J} \sum_{j=1}^{J} (\frac{\mathbf{F}_k(j)}{N_k(j)} - \mathbf{T}_k \mathbb{E}_I[\mathbf{i}_j]), \tag{A.105}$$

  with J the total number of utterances in the training dataset,

- for the subspace $\mathbf{T}$ matrix

$$\mathbf{T}_k = \mathbf{C}_k \mathbf{A}_k^{-1}, \tag{A.106}$$

where

$$\mathbf{C}_k = \sum_j (\mathbf{F}_k(j) - N_k(j)\mu)\mathbb{E}_I[\mathbf{i}_j]^\intercal \tag{A.107}$$

$$\mathbf{A}_k = \sum_j \mathbb{E}_I[\mathbf{i}_j \mathbf{i}_j^\intercal] N_k(j), \tag{A.108}$$

- for the covariance matrix

$$\begin{aligned}\mathbf{\Sigma}_k = \frac{1}{J}\sum_j \frac{\mathbf{S}_k(j) - \mathbf{F}_k(j)(\mu^\intercal + \mathbb{E}_I[\mathbf{i}_j]^\intercal \mathbf{T}_k^\intercal) - (\mu + \mathbf{T}_k\mathbb{E}_I[\mathbf{i}_j])\mathbf{F}_k(j)^\intercal}{N_k(j)} \\ +\mu(\mu^\intercal + \mathbb{E}_I[\mathbf{i}_j]^\intercal \mathbf{T}_k^\intercal) + \mathbf{T}_k(\mathbb{E}_I[\mathbf{i}_j]\mu^\intercal + \mathbb{E}_I[\mathbf{i}_j\mathbf{i}_j^\intercal]\mathbf{T}_k^\intercal).\end{aligned} \tag{A.109}$$

In case of applying the EM with MD, the process is again the same as for JFA depicted in section A.8.2, and the update equations of the minimization of the KL divergence and model parameters transformation are

- Minimization step

$$\mu_I = \frac{1}{J}\sum_{j=1}^{J} \mathbb{E}_I[\mathbf{i}_j], \tag{A.110}$$

$$\mathbf{\Sigma}_I = \sum_{j=1}^{J} \mathbf{L}_j^{-1} + (\mu_I - \mathbb{E}_I[\mathbf{i}_j])(\mu_I - \mathbb{E}_I[\mathbf{i}_j])^\intercal. \tag{A.111}$$

- Equivalent model conversion step

$$\mathbf{T} = \mathbf{T}_1 = \mathbf{T}_2 \mathbf{P}_I, \tag{A.112}$$

$$\mu = \mu_1 = \mu_2 + \mathbf{T}_2 \mu_{I_2}. \tag{A.113}$$

Once the model is trained, *i-Vectors* of new utterances are computed as the expectation of the posterior distribution of $\mathbf{i}$ given $\mathbf{O}$, $p(\mathbf{i}|\mathbf{O})$, as per eq. (A.103).

In practical implementations, sufficient statistics are often normalized with the UBM mean and covariance, as suggested in Glembek et al. [2011]. This simplifies the rest of the model training and evaluation.

APPENDIX B

Expectation Maximization Algorithm

In this Appendix, we describe the EM algorithm. EM is one of the central algorithms in this Thesis, used to train most of our Gaussian linear model parameters. It is a ML optimization technique, thus the likelihood of the data is increased at each iteration. For an in-depth description, see Dempster et al. [1977]; Bishop [2006].

In short, the EM algorithm's goal is to maximize the likelihood of probabilistic models with latent variables. Consider a dataset of observed variables defined by $\mathbf{O}$, a set of hidden variables defined by $\mathbf{Z}$, and the parameters of our model given by Θ. The likelihood function is given by

$$p(\mathbf{O}|\Theta) = \sum_{\mathbf{Z}} p(\mathbf{O}, \mathbf{Z}|\Theta). \tag{B.1}$$

In this equation, we assume discrete latent variables, but the derivation is identical if they were continuous, or if we had a combination of discrete and continuous variables, but sums would be replaced by integrals.

Because the logarithm is a monotonic function, maximizing the likelihood is equivalent to maximizing the logarithm of the likelihood, or log-likelihood. So, let us express the log-likelihood function in eq. (B.1) as

$$\ln p(\mathbf{O}|\Theta) = \ln \sum_{\mathbf{Z}} p(\mathbf{O}, \mathbf{Z}|\Theta) = \mathcal{L}(q, \Theta) + KL(q||p), \tag{B.2}$$

where we have defined the *lower bound* of the log-likelihood function as

$$\mathcal{L}(q, \Theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{O}, \mathbf{Z}|\Theta)}{q(\mathbf{Z})}, \tag{B.3}$$

and

$$KL(q||p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z}|\mathbf{O},\Theta)}{q(\mathbf{Z})} \tag{B.4}$$

is the KL divergence between $q(\mathbf{Z})$, a distribution over the latent variables, and the true posterior distribution, $p(\mathbf{Z}|\mathbf{O})$. We can observe that for any choice of $q(\mathbf{Z})$, equation B.2 holds. Recall that $KL(q||p) \geq 0$, with equality if, and only if, $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{O},\Theta)$, and thus it follows that $\mathcal{L}(q,\Theta) \leq \ln p(\mathbf{O}|\Theta)$, so $\mathcal{L}(q,\Theta)$ is a lower bound of $\ln p(\mathbf{O}|\Theta)$. The same conclusion can be reached by making use of *Jensen's inequality*,

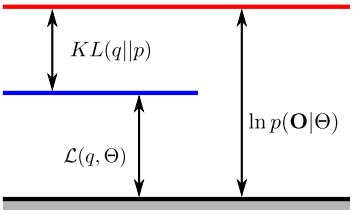
$$\begin{aligned} \ln p(\mathbf{O}|\Theta) &= \ln \sum_{\mathbf{Z}} p(\mathbf{O},\mathbf{Z}|\Theta) = \ln \sum_{\mathbf{Z}} q(\mathbf{Z}) \frac{p(\mathbf{O},\mathbf{Z}|\Theta)}{q(\mathbf{Z})} \\ &\geq \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{O},\mathbf{Z}|\Theta)}{q(\mathbf{Z})} = \mathcal{L}(q,\Theta), \end{aligned} \tag{B.5}$$

with equality only when $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{O},\Theta)$, and in such case, the KL divergence previously defined goes to zero.

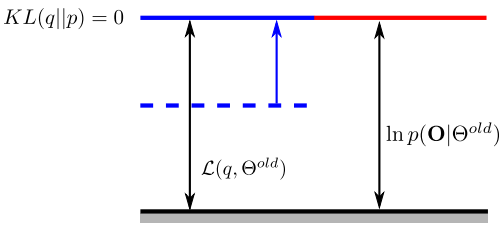
The main assumption of the EM algorithm is that the direct optimization of $p(\mathbf{O}|\Theta)$ is difficult, but the optimization of the complete-data likelihood function $p(\mathbf{O},\mathbf{Z}|\Theta)$ is easier. The EM is a two-stage iterative process that maximizes the data likelihood as follows:

- E step: in this stage we start from the previous values of Θ, Θ^{old}, and the lower bound $\mathcal{L}(q,\Theta)$ is maximized with respect to function q, holding Θ^{old} fixed. Given that $\ln p(\mathbf{O}|\Theta)$ does not depend on q, the maximum of $\mathcal{L}(q,\Theta)$ occurs when $q = p$ and then $KL(q||p) = 0$. At this point the lower bound is equal to the log-likelihood. This is illustrated in Figure B.1.

- M step: in this step the distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}$ is maximized with respect to Θ, to obtain a new estimate of the parameters, Θ^{new}. The maximization causes the lower bound to increase, which necessarily makes the log-likelihood of the incomplete dataset, $p(\mathbf{O}|\Theta)$, to increase. The reason is that the KL will also increase, because q is held fixed, but now it will not equal the new posterior distribution $p(\mathbf{Z}|\mathbf{O},\Theta^{new})$, and the KL will not be zero. The increase in the log-likelihood is greater than the increase in the lower bound, as can be seen in Figure B.2.



(a) Log-likelihood decomposition. The decomposition given by eq. (B.2) holds for any choice of distribution $q(\mathbf{Z})$. Because the KL divergence satisfies $KL(q||p) \geq 0$, we see that the quantity $\mathcal{L}(q,\Theta)$ is a lower bound on the log-likelihood function $\ln p(\mathbf{O}|\Theta)$. Figure taken from Bishop [2006].



(b) E step of the EM algorithm. The q distribution is set equal to the posterior distribution for the current parameter values Θ^{old}, causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing. Figure taken from Bishop [2006].

Figure B.1: Decomposition of log-likelihood function and E step of the EM algorithm.

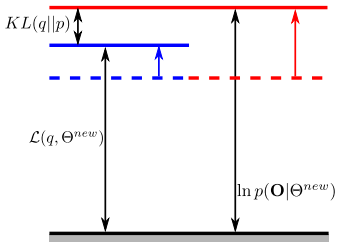


Figure B.2: M step of the EM algorithm - The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q,\Theta)$ is maximized with respect to the parameter vector Θ to give a revised value Θ^{new}. Because the KL divergence is nonnegative, this causes the log likelihood $\ln p(\mathbf{O}|\Theta)$ to increase by at least as much as the lower bound does. Figure taken from Bishop [2006].

In short, in the E step we calculate the posterior distribution $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{O},\Theta)$, and in the M step, we obtain new estimates of the parameters that maximize the likelihood of the observed data. If we substitute this amount into B.3,

$$\begin{aligned}\mathcal{L}(q,\Theta) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{O},\Theta^{old}) \ln p(\mathbf{O},\mathbf{Z}|\Theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{O},\Theta^{old}) \ln p(\mathbf{Z}|\mathbf{O},\Theta^{old}) \\ = \mathcal{Q}(\Theta,\Theta^{old}) + const,\end{aligned} \tag{B.6}$$

it can be seen that the maximization of the lower bound is equivalent to the maximization of the expectation of the complete data log-likelihood with respect to the posterior probability of $\mathbf{Z}$ given $\mathbf{O}$ and Θ^{old}, expressed as $\mathcal{Q}$, because the constant part

of B.6 is independent of Θ. Note that if the joint distribution $p(\mathbf{O}, \mathbf{Z}|\Theta)$ is a member of the exponential family, or product of such members, the logarithm will cancel the exponential and lead to an M step typically much simpler than the maximization of the incomplete data log-likelihood $p(\mathbf{O}|\Theta)$.

APPENDIX C

Confusion Matrices 2009 NIST LRE

In this Appendix, we include the confusion matrices of the experiments reported in Chapter 9 on the 2009 NIST LRE database, using a prior equal to 0.5, and a development dataset matched to the test dataset in length, meaning that, for each duration condition (3 s, 10 s, and 30 s tasks), the length of the utterances are as long as the test utterances or longer. Results for the acoustic, prosodic, and fusion systems are included. Observe that the results for the 3 s task, are the same for the experiments with matched and unmatched development datasets because for this task all development data were used in both cases. First, we include results for 3 s, 10 s, and 30 s tasks with the acoustic system, then for the three tasks with the prosodic system with formant information, and finally, for the three tasks with the fusion system.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	86.78	5.05	5.53	8.65	5.53	18.27	9.86	5.53	11.06	17.31	5.05	12.26	19.71	6.49	5.53	19.47	5.77	9.38	11.78	7.69	8.89	14.90	10.58
bosn	8.27	65.60	4.80	12.80	71.20	18.93	9.07	8.80	10.13	23.73	16.00	1.07	17.87	4.27	4.00	14.40	18.40	19.20	18.40	10.93	24.00	14.40	4.27
cant	2.87	0.86	87.68	3.72	0.57	9.46	16.62	10.32	9.46	14.33	2.58	3.72	10.60	7.45	26.93	5.16	2.87	4.30	10.32	1.43	6.88	7.16	30.09
creo	7.14	7.14	5.14	81.14	10.00	7.71	12.57	9.14	6.86	68.29	6.29	10.57	16.00	7.14	6.29	10.57	12.57	9.43	26.86	6.00	6.29	11.43	10.00
croa	6.35	66.50	3.55	7.36	74.37	18.27	10.41	4.82	10.91	17.77	9.14	2.54	14.97	2.03	2.54	15.99	16.24	18.53	17.26	9.90	26.65	11.68	2.54
dari	13.51	8.85	8.11	12.29	6.39	80.84	8.85	5.41	58.72	22.36	11.30	7.37	19.16	7.86	7.62	38.57	9.83	7.13	13.76	12.29	7.13	18.18	6.63
engi	9.59	6.26	8.81	4.50	5.48	17.42	84.34	49.71	12.33	18.00	5.48	5.09	33.66	3.72	7.05	9.59	12.33	9.00	16.24	7.24	9.00	26.22	8.61
engl	5.31	2.77	4.27	3.70	2.66	7.97	42.03	91.22	9.24	9.93	3.70	4.16	9.93	3.46	8.31	5.08	7.62	6.24	6.93	5.08	7.62	8.43	5.77
fars	6.49	7.01	5.19	9.61	5.97	62.86	9.35	7.01	86.49	17.66	7.79	6.49	12.73	4.42	7.27	18.18	6.49	5.45	7.79	14.03	4.16	12.47	4.94
fren	8.96	6.47	6.72	61.44	7.46	17.91	12.94	6.22	12.19	89.05	6.72	6.47	19.90	9.20	9.95	16.67	19.90	12.44	20.90	10.95	14.93	15.67	8.21
geor	12.16	13.90	2.73	11.41	21.34	11.91	6.45	6.20	10.17	18.61	81.14	3.23	17.37	6.70	5.21	16.38	10.67	17.12	12.90	10.67	16.87	12.66	4.47
haus	21.97	2.97	6.64	12.81	1.60	14.87	9.15	10.98	9.61	21.28	2.75	74.83	15.79	9.61	8.70	39.59	10.76	11.90	13.04	5.72	8.01	12.36	13.73
hind	11.56	5.78	7.81	7.34	6.09	18.44	29.06	9.84	12.03	13.12	9.84	7.81	82.66	4.84	6.88	23.91	8.12	10.00	17.34	9.22	8.44	78.12	8.59
kore	11.04	3.90	20.35	8.44	3.46	13.20	12.34	7.36	9.74	16.23	3.03	9.31	13.20	83.77	19.05	15.37	3.46	4.76	12.34	7.58	3.68	9.31	12.34
mand	4.40	2.15	23.23	5.42	2.46	7.57	11.57	8.70	10.24	15.97	2.66	8.60	8.80	13.92	87.92	7.88	5.63	5.73	5.42	4.61	6.55	6.55	11.05
pash	17.73	9.11	5.17	7.64	10.10	42.61	9.61	9.36	21.43	21.92	12.81	19.70	30.54	7.88	7.88	79.56	11.08	16.75	14.29	14.53	13.55	28.57	7.88
port	6.55	8.88	2.96	20.30	12.26	15.64	13.74	7.40	8.25	40.17	6.34	4.65	19.66	3.81	7.19	19.87	78.86	18.18	23.68	14.16	16.49	15.43	6.34
russ	5.79	5.17	2.89	9.92	12.19	9.30	12.81	10.33	5.58	19.01	9.50	4.96	12.60	3.51	8.06	12.40	16.94	87.40	11.78	9.71	58.47	9.30	5.99
span	7.21	11.19	5.47	14.43	16.17	10.45	10.20	5.47	6.97	19.90	3.98	3.73	15.42	2.49	2.99	8.96	10.95	5.97	88.81	3.98	5.97	10.70	8.46
turk	9.09	6.82	3.03	11.87	6.57	26.26	15.15	10.35	21.21	30.05	9.85	8.33	22.98	6.31	10.61	26.26	15.66	12.37	10.10	75.25	10.35	18.18	4.29
ukra	7.39	17.00	2.71	10.84	26.85	18.72	10.10	8.87	11.82	19.70	12.32	7.88	18.97	3.94	6.90	16.75	18.72	65.27	14.78	8.87	66.50	16.50	3.69
urdu	7.20	5.91	6.17	8.48	5.40	18.51	17.99	8.48	11.57	13.62	5.66	9.00	85.60	2.57	5.14	36.25	11.05	7.71	16.45	8.48	10.03	83.80	7.97
viet	6.71	1.77	34.98	2.83	2.12	10.60	18.37	12.37	7.07	12.72	1.77	8.83	10.95	7.07	14.13	8.48	5.65	3.89	12.72	3.53	2.83	9.19	87.28

Table C.1: Confusion matrix of the 3 s task of 2009 NIST LRE database for the acoustic system - Prior equal to 0.5. Numbers represent % of files. Note that these results are the same for the experiments with matched and unmatched development datasets.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	98.73	1.01	0.51	2.53	0.00	4.05	1.52	0.76	2.78	4.30	0.51	4.05	4.30	0.51	1.27	2.53	1.52	2.03	2.53	2.78	1.27	2.78	2.78
bosn	1.67	84.96	0.00	2.23	84.68	9.47	1.39	1.11	2.79	9.47	6.13	0.56	8.36	1.95	0.28	9.19	10.03	14.21	6.69	6.13	17.55	6.13	0.28
cant	0.29	0.00	97.41	0.86	0.00	1.15	4.90	1.73	1.15	2.88	0.00	0.58	2.02	2.59	8.36	0.86	1.15	0.29	2.31	0.29	0.29	1.44	6.34
creo	0.96	0.32	0.96	96.15	0.00	0.96	2.56	1.60	1.28	49.04	2.88	0.96	3.85	4.17	2.56	1.28	6.41	2.88	17.95	3.53	1.28	3.53	0.32
croa	1.89	76.76	0.27	2.70	88.11	8.65	1.35	0.54	1.62	5.41	4.05	0.00	7.03	0.54	0.27	5.68	12.16	11.35	7.57	3.24	16.49	4.05	0.00
dari	4.38	2.84	1.55	4.38	1.03	91.75	3.09	2.58	60.05	11.34	5.41	2.32	8.25	1.55	1.29	23.97	2.32	3.87	5.15	5.15	2.58	6.96	2.58
engi	2.06	0.19	2.63	2.06	0.00	5.44	94.93	41.09	3.75	6.38	1.31	1.31	24.39	0.75	2.63	4.69	3.94	2.06	4.88	1.31	2.06	15.57	5.44
engl	0.72	0.00	1.20	0.24	0.00	2.03	30.62	98.44	1.32	2.99	0.60	0.84	3.23	0.48	1.56	1.44	1.67	1.08	1.67	0.96	1.08	2.15	1.08
fars	1.57	0.26	0.52	1.83	0.78	45.17	1.83	0.26	99.48	4.44	1.31	0.52	2.87	1.04	1.04	3.39	0.78	0.78	2.61	3.66	1.04	2.35	0.52
fren	3.54	0.51	0.51	66.84	0.51	4.81	3.29	1.52	2.78	96.96	2.03	1.77	7.85	1.52	2.53	5.82	9.62	4.30	8.86	2.78	5.06	7.34	2.78
geor	1.77	5.56	0.25	3.28	5.30	3.54	2.53	1.52	3.54	7.32	96.46	0.76	8.08	0.76	0.76	6.31	3.03	6.82	3.03	3.03	5.30	5.56	0.25
haus	11.69	1.04	0.78	3.12	0.00	4.42	3.90	1.30	1.82	5.45	0.00	96.36	4.68	1.30	0.52	22.08	1.82	2.86	2.34	1.30	1.04	2.86	3.64
hind	5.05	0.00	2.28	2.12	0.16	6.03	14.17	1.63	1.95	3.09	3.26	2.28	96.91	1.14	1.47	11.07	2.93	2.93	7.00	3.26	2.28	93.32	1.95
kore	4.22	0.22	7.33	1.56	0.22	4.44	4.44	1.33	3.33	4.67	0.44	4.00	5.78	98.22	5.11	5.78	0.67	0.89	4.44	2.00	0.89	3.33	3.56
mand	0.51	0.10	6.90	0.82	0.10	0.93	3.09	0.93	1.13	3.60	0.31	2.68	1.96	4.22	98.15	2.57	1.03	0.62	0.31	0.72	0.72	1.34	2.47
pash	11.76	2.56	0.26	1.02	2.05	39.90	3.58	2.05	14.83	9.21	8.44	7.93	21.23	2.30	2.56	91.56	4.86	12.28	4.35	7.42	9.72	19.69	2.05
port	3.17	1.58	0.79	6.07	3.17	5.28	1.06	1.06	2.11	18.47	1.85	1.58	3.17	0.53	1.32	6.07	96.31	8.18	7.12	4.75	8.71	1.32	0.53
russ	0.42	0.84	0.21	0.84	1.04	2.51	2.30	1.46	1.46	3.13	2.71	1.04	2.09	0.84	1.25	1.67	4.18	98.54	0.21	2.09	42.17	1.25	0.42
span	1.57	2.61	0.00	4.96	3.66	2.35	2.87	1.57	0.78	6.79	0.26	0.26	2.61	0.52	0.00	2.61	4.44	1.04	98.43	1.31	2.35	1.04	0.78
turk	2.79	0.25	0.00	3.81	0.00	9.64	2.79	2.03	12.44	11.93	3.30	1.27	8.38	1.52	3.05	8.63	6.60	4.31	1.78	95.18	1.78	5.84	0.25
ukra	1.04	9.40	0.00	4.96	18.54	6.01	0.78	1.57	3.39	9.92	7.05	2.09	5.22	0.78	1.83	4.96	8.88	78.85	7.05	5.48	83.81	3.92	0.78
urdu	2.41	0.80	0.80	1.61	0.27	6.97	5.63	0.54	3.22	4.56	1.61	1.88	97.32	0.80	0.80	17.43	3.22	1.34	4.56	1.88	1.88	97.05	0.54
viet	1.43	0.71	15.71	1.07	0.00	2.14	5.71	3.57	2.14	3.21	0.36	1.43	2.50	1.79	2.86	0.71	0.00	0.71	0.71	0.36	1.07	2.14	97.14

Table C.2: Confusion matrix of the 10 s task of 2009 NIST LRE database for the acoustic system - Prior equal to 0.5 and development dataset with utterances at least 10 s long. Numbers represent % of files.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	98.69	1.04	0.00	0.00	0.00	0.52	0.52	0.26	0.26	0.52	0.00	0.52	0.26	0.00	0.00	0.78	0.26	0.00	0.00	0.00	0.00	0.00	0.00
bosn	0.30	96.98	0.00	0.00	95.77	1.81	0.30	0.00	0.00	1.51	2.11	0.00	1.81	0.00	0.00	2.11	2.11	6.65	0.60	0.30	10.88	0.91	0.00
cant	0.27	0.00	99.73	0.00	0.00	0.53	1.33	0.53	0.00	0.27	0.00	0.00	0.80	0.00	1.87	0.27	0.00	0.00	0.80	0.00	0.00	0.27	1.33
creo	0.65	0.00	0.33	99.02	0.00	0.33	0.00	0.00	0.00	28.99	0.33	0.00	0.33	0.33	0.00	0.65	1.30	0.33	7.17	0.65	0.33	0.00	0.00
croa	0.00	75.82	0.00	0.27	95.88	2.47	0.55	0.00	0.82	1.10	0.82	0.00	2.20	0.00	0.00	1.92	3.30	4.95	2.47	0.00	9.34	1.10	0.00
dari	1.08	1.08	0.00	0.81	0.00	92.47	0.81	0.00	60.48	2.96	1.34	1.61	3.76	0.27	0.00	11.02	0.27	0.81	0.54	0.81	1.34	3.49	0.27
engi	0.52	0.00	0.00	0.17	0.00	1.03	97.93	33.28	0.52	2.41	0.34	0.17	9.83	0.00	0.17	0.69	0.52	0.17	0.69	0.00	0.52	6.38	0.34
engl	0.44	0.22	0.33	0.00	0.00	0.66	25.08	98.90	0.22	1.10	0.00	0.22	1.20	0.00	0.22	0.66	0.44	0.44	0.44	0.11	0.33	0.55	0.44
fars	0.00	0.00	0.00	0.00	0.00	26.60	1.02	0.51	99.74	0.51	0.00	0.00	0.26	0.00	0.26	1.02	0.26	0.00	0.00	0.77	0.00	0.26	0.00
fren	1.29	0.00	0.00	65.72	0.00	0.26	0.77	0.00	0.26	99.23	0.52	0.26	0.52	0.77	0.77	1.03	4.64	0.52	3.35	0.52	2.32	0.52	0.00
geor	0.50	3.02	0.00	0.75	2.01	1.26	0.75	0.00	1.01	1.51	98.49	0.00	1.51	0.25	0.00	2.26	1.01	1.26	1.26	0.00	2.26	0.50	0.00
haus	8.12	0.00	0.00	0.29	0.00	0.87	0.29	0.58	0.58	0.58	0.00	99.13	0.00	0.29	0.29	8.12	0.29	0.00	0.58	0.00	0.00	0.00	0.29
hind	1.65	0.00	0.00	0.00	0.00	1.95	12.72	1.20	0.75	1.35	0.45	0.30	98.65	0.15	0.45	3.44	0.30	0.15	1.05	0.00	0.45	97.31	0.15
kore	0.88	0.00	1.10	0.00	0.00	0.44	0.44	0.00	0.44	0.44	0.00	0.00	0.44	100.00	0.44	0.22	0.00	0.00	0.44	0.00	0.00	0.22	0.00
mand	0.10	0.00	1.85	0.00	0.00	0.29	0.29	0.29	0.19	0.49	0.00	0.39	0.29	0.10	99.71	0.58	0.10	0.10	0.00	0.10	0.19	0.29	0.39
pash	10.82	0.52	0.00	1.03	0.00	31.70	0.26	0.00	8.25	4.38	2.58	2.84	7.73	0.52	0.26	97.94	1.55	5.67	2.06	1.03	4.38	5.93	0.26
port	0.00	0.00	0.00	1.47	0.29	1.18	0.00	0.00	0.59	6.19	0.00	0.00	1.18	0.29	0.00	2.36	100.00	1.47	2.06	0.59	1.77	0.88	0.00
russ	0.19	0.00	0.00	0.19	0.00	0.00	0.57	0.19	0.00	0.57	0.19	0.38	0.57	0.00	0.00	0.19	0.38	99.81	0.00	0.00	25.62	0.19	0.00
span	0.00	0.27	0.00	0.54	0.27	0.27	0.27	0.27	0.00	0.81	0.00	0.00	0.00	0.00	0.00	0.27	0.54	0.00	100.00	0.00	0.00	0.00	0.00
turk	0.51	0.26	0.00	0.51	0.00	2.30	0.51	0.26	3.32	2.30	1.53	0.26	1.53	0.26	0.26	2.30	0.77	0.51	0.51	99.23	0.77	1.02	0.26
ukra	0.80	7.47	0.00	1.33	14.67	1.33	0.27	0.53	1.07	4.80	3.20	0.27	3.20	0.27	0.00	2.13	3.47	83.47	3.47	1.87	96.53	2.40	0.00
urdu	0.54	0.00	0.00	0.00	0.00	1.35	2.43	0.81	0.27	1.08	0.54	0.27	99.73	0.27	0.00	3.77	0.54	0.00	0.54	0.00	0.00	99.46	0.00
viet	0.63	0.00	9.21	0.00	0.00	0.63	5.08	0.95	0.00	3.17	0.00	0.00	0.95	0.00	1.59	0.63	0.00	0.00	0.63	0.00	0.00	0.95	97.78

Table C.3: Confusion matrix of the 30 s task of 2009 NIST LRE database for the acoustic system - Prior equal to 0.5 and development dataset with utterances at least 30 s long. Numbers represent % of files.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	70.19	13.22	8.17	18.03	19.47	28.61	23.08	20.91	24.28	26.68	24.04	15.87	24.76	15.38	12.98	18.27	13.94	24.04	25.00	17.55	6.25	12.98	18.75
bosn	10.40	64.00	5.87	16.80	70.40	28.80	17.07	27.73	24.00	18.93	20.00	11.73	24.53	12.00	14.40	26.67	28.00	34.13	30.40	10.93	20.80	18.67	7.73
cant	5.73	2.58	74.21	3.15	5.73	10.03	32.95	17.48	12.32	7.74	6.88	12.89	13.18	22.64	39.83	6.88	2.87	7.45	20.34	2.87	4.58	9.46	32.09
creo	22.57	19.14	5.14	64.00	21.71	23.71	16.57	16.29	24.29	48.57	19.14	14.29	25.43	14.00	16.00	24.00	31.71	28.86	32.86	15.14	9.43	16.00	7.43
croa	10.91	51.78	7.36	11.42	62.44	30.96	17.51	24.62	29.44	19.29	18.53	11.68	25.89	11.42	11.68	25.38	21.32	32.74	32.49	16.24	20.05	19.04	7.87
dari	16.71	17.20	11.06	11.55	22.36	66.09	25.80	23.34	57.99	23.34	20.88	17.20	25.06	15.97	14.25	42.26	14.50	20.15	20.15	12.04	8.35	19.66	15.97
engi	14.48	9.98	15.46	4.50	14.87	22.90	73.78	52.25	23.48	16.24	14.87	7.63	45.01	14.48	16.63	18.00	12.33	16.05	22.90	13.11	14.29	34.25	21.14
engl	9.35	12.82	12.70	6.58	18.01	19.05	56.58	77.02	19.63	13.39	16.63	10.85	19.98	10.51	17.32	15.94	15.36	18.59	17.78	10.39	10.39	17.44	14.67
fars	14.29	21.56	9.61	15.58	21.30	64.42	24.94	21.82	70.39	35.84	12.99	10.13	31.43	17.40	12.73	36.62	15.84	17.92	22.34	18.70	10.39	21.30	10.91
fren	20.90	17.16	4.98	41.79	19.40	32.34	18.66	21.89	28.36	69.15	14.18	9.45	23.13	12.19	12.44	24.13	32.34	28.36	26.87	21.14	10.95	15.42	10.20
geor	18.61	17.62	4.96	10.67	36.23	21.59	30.77	16.87	20.84	17.12	72.95	7.94	29.28	14.14	6.70	20.60	11.66	31.51	22.58	17.87	9.93	19.85	11.17
haus	20.82	13.27	11.21	11.44	22.20	29.75	22.88	27.92	23.57	20.37	13.50	67.96	25.86	14.87	20.59	45.08	25.40	28.60	31.12	5.03	5.49	16.70	18.08
hind	13.75	9.22	11.56	8.12	18.44	22.66	51.09	23.44	26.41	13.28	23.59	10.47	71.72	13.28	11.72	25.47	12.81	19.53	26.88	12.97	9.69	63.28	14.06
kore	18.61	12.77	16.45	7.14	17.53	33.77	34.42	17.75	24.68	17.53	13.85	11.47	22.08	61.26	24.03	29.87	10.61	20.35	37.23	8.66	6.71	15.58	17.53
mand	8.19	10.54	29.68	6.55	16.07	15.56	29.38	22.31	16.89	13.92	6.45	14.02	14.02	28.56	75.84	13.31	8.80	15.66	22.11	5.42	7.68	10.44	19.04
pash	19.70	21.43	5.42	14.29	23.40	48.52	17.73	22.66	39.41	14.53	21.43	22.66	30.79	15.02	8.62	68.72	15.02	28.33	22.91	10.59	16.01	25.86	9.36
port	14.59	22.83	6.13	20.08	25.58	39.32	13.53	27.06	24.52	36.79	13.11	16.70	14.38	11.42	19.87	34.46	74.63	31.08	31.92	9.94	11.21	8.03	11.84
russ	23.35	17.98	3.72	15.29	32.64	24.17	30.99	22.11	18.60	17.36	29.96	11.16	23.14	14.46	13.64	24.79	18.18	75.00	26.65	12.19	34.30	16.94	10.33
span	18.16	36.32	4.48	19.90	46.77	26.62	25.37	23.13	21.14	27.86	17.66	9.95	29.60	14.43	12.19	20.65	23.88	19.15	71.14	12.19	8.71	16.67	12.69
turk	22.98	15.66	4.29	18.94	17.68	28.54	19.44	18.69	33.08	35.10	23.48	6.31	23.48	14.39	10.35	25.25	18.94	28.03	18.43	67.42	8.33	20.20	11.87
ukra	13.05	34.48	3.20	16.26	43.35	22.91	16.26	20.94	18.47	19.70	20.20	11.33	17.73	11.82	10.10	27.34	25.37	67.24	25.12	12.32	45.57	15.52	8.87
urdu	13.11	19.02	9.25	13.11	20.82	26.99	37.02	17.74	29.31	14.14	19.28	11.05	71.72	10.03	13.88	28.28	15.68	15.94	19.02	11.57	11.31	64.78	12.08
viet	10.25	2.83	42.05	2.12	3.18	18.73	30.04	14.84	16.61	10.25	9.19	10.25	18.02	12.37	20.49	15.19	6.71	5.30	21.55	4.24	2.12	11.31	83.04

Table C.4: Confusion matrix of the 3 s task of 2009 NIST LRE database for the prosodic system with formant information - Prior equal to 0.5. Numbers represent % of files. Note that these results are the same for the matched and unmatched experiments.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	90.63	2.78	1.77	11.39	3.54	10.13	10.89	5.32	9.11	15.95	11.14	6.33	16.46	8.10	3.29	9.37	7.34	8.61	13.67	12.91	5.57	14.43	7.59
bosn	1.39	74.37	0.84	6.13	81.06	13.09	3.62	9.75	11.98	7.24	10.86	2.51	9.47	3.62	4.18	13.65	18.38	25.91	14.76	5.57	21.17	10.03	0.28
cant	1.44	0.00	92.51	0.86	0.29	0.58	10.37	5.19	1.15	1.73	3.17	2.02	4.32	10.66	25.07	0.86	1.15	0.86	3.75	0.00	0.86	3.75	17.87
creo	8.65	4.17	0.00	85.90	7.37	5.77	5.13	5.77	8.65	36.86	9.94	2.24	10.26	6.41	3.21	9.29	16.03	14.42	16.67	5.13	7.37	11.22	4.81
croa	2.70	65.41	0.81	6.49	82.16	18.38	4.59	10.54	11.89	10.27	11.62	4.05	12.16	3.51	2.97	13.78	18.92	26.49	18.11	9.73	24.05	13.51	1.08
dari	12.63	5.15	3.09	7.99	9.79	78.61	12.37	11.60	64.43	17.78	19.59	7.22	19.07	10.57	6.44	37.89	6.19	11.60	15.98	9.28	5.67	21.39	9.54
engi	9.94	1.88	5.82	2.25	4.69	10.32	85.18	54.03	12.95	9.94	7.69	4.13	43.90	9.19	10.32	9.57	6.19	12.01	14.26	8.82	8.07	35.46	14.26
engl	4.07	3.11	3.59	2.75	7.89	6.70	50.60	91.15	6.34	7.18	7.30	3.71	10.17	5.14	9.33	5.98	8.25	9.81	9.09	5.86	8.49	10.77	8.61
fars	6.79	7.05	2.61	15.14	9.14	65.80	14.88	6.53	84.86	28.20	8.09	5.48	24.54	8.62	5.22	19.06	9.92	8.88	13.58	14.88	6.01	24.02	4.70
fren	11.65	4.56	1.27	52.91	7.34	14.68	4.30	7.59	15.44	87.09	11.39	4.81	9.62	6.08	4.81	10.13	26.33	19.24	16.71	15.44	12.91	10.13	5.06
geor	10.61	7.58	1.26	8.33	22.47	13.13	12.12	8.08	11.87	8.59	88.64	2.53	15.40	7.07	2.53	11.36	5.30	21.72	12.63	7.83	9.09	16.16	2.78
haus	12.21	2.60	3.12	6.23	5.71	11.43	9.09	12.47	8.31	4.68	7.79	87.79	8.83	7.79	9.09	25.97	16.10	13.51	14.03	1.82	4.42	8.83	5.97
hind	7.82	2.61	5.21	5.37	6.03	7.17	33.22	8.79	9.77	6.51	15.64	4.56	87.13	7.65	5.37	12.70	6.03	8.47	14.01	7.00	5.05	85.34	7.65
kore	13.33	3.56	11.11	3.56	3.78	22.44	24.00	7.33	16.89	8.44	5.56	9.11	16.00	82.44	15.78	23.33	2.67	10.67	24.89	4.44	6.00	14.44	10.00
mand	1.85	2.06	22.45	2.27	3.60	4.43	12.67	8.75	5.87	6.08	2.27	7.52	6.28	22.04	93.92	3.09	2.47	6.59	8.34	2.16	3.30	4.94	9.78
pash	18.93	8.70	2.81	9.21	13.04	42.71	6.39	10.49	34.27	13.55	22.51	17.39	19.95	11.76	3.84	78.01	14.32	21.99	17.14	14.07	14.32	25.06	5.12
port	7.12	9.76	0.79	12.40	16.89	24.54	3.43	13.46	15.83	28.50	7.12	9.76	8.18	6.07	4.75	23.48	88.65	20.05	16.62	9.23	10.03	7.39	3.17
russ	12.53	5.64	0.84	8.56	14.82	10.02	16.08	8.98	7.52	10.23	17.95	6.26	12.32	7.10	7.10	13.57	12.73	88.73	11.48	6.26	51.15	16.49	3.76
span	10.44	28.72	2.09	16.45	40.47	14.62	8.09	8.36	10.70	15.93	11.49	3.13	12.53	8.88	4.44	11.49	15.93	6.79	84.33	5.48	7.05	7.57	4.18
turk	9.39	3.05	1.02	8.38	6.85	7.61	6.85	5.08	19.54	17.77	11.93	1.27	9.90	6.09	3.55	8.12	7.11	9.64	6.60	89.34	4.31	15.48	3.55
ukra	6.01	24.28	0.78	11.49	36.55	8.88	5.74	7.31	6.27	11.23	11.49	6.79	11.23	3.92	2.87	14.62	16.97	72.06	16.19	9.14	69.19	15.14	0.78
urdu	5.90	2.68	2.41	4.83	5.36	13.40	20.91	6.43	16.09	5.90	10.19	4.83	85.79	6.43	3.49	19.03	7.51	7.51	9.12	10.72	5.09	85.79	5.63
viet	3.21	0.36	26.79	1.43	0.36	7.14	13.93	6.07	5.71	3.93	3.57	5.71	6.43	5.36	6.07	4.64	1.79	1.07	8.21	1.07	0.71	6.43	94.64

Table C.5: Confusion matrix of the 10 s task of 2009 NIST LRE database for the prosodic system with formant information - Prior equal to 0.5 and development dataset with utterances at least 10 s long. Numbers represent % of files.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	96.87	0.52	0.78	1.83	0.52	4.44	4.44	0.78	4.18	5.48	3.92	2.09	5.22	2.09	0.00	7.05	3.39	4.18	4.70	6.79	0.00	2.09	2.35
bosn	0.00	87.31	0.30	2.11	89.73	7.85	1.81	1.81	4.53	1.51	5.14	1.21	4.23	0.60	0.00	6.95	11.48	15.11	8.46	1.21	10.57	2.72	0.00
cant	0.00	0.00	98.13	0.00	0.00	0.27	4.80	1.07	0.00	0.53	0.53	1.60	1.07	4.27	12.53	0.27	0.00	0.27	1.07	0.27	0.27	0.27	6.93
creo	3.26	0.33	0.33	94.46	2.28	1.63	2.93	1.95	2.28	21.17	2.28	0.33	3.58	1.30	0.98	2.28	5.54	4.56	13.03	0.65	2.28	2.93	0.65
croa	0.27	77.20	0.00	1.92	93.96	10.99	2.20	0.82	6.59	6.04	5.77	0.82	6.04	0.27	0.55	6.87	12.36	19.51	10.99	1.92	5.49	4.12	0.00
dari	7.53	4.57	2.15	5.38	6.45	87.90	6.18	2.96	76.88	14.78	13.98	2.96	13.44	4.30	1.88	24.19	1.88	6.72	12.63	2.96	1.34	9.68	2.96
engi	5.86	0.34	0.17	1.03	1.03	6.03	96.03	52.59	9.83	6.90	3.45	1.21	43.10	3.62	3.45	5.34	3.10	5.52	6.90	3.45	1.21	14.31	3.97
engl	1.31	1.42	1.42	0.44	2.52	3.50	44.47	94.74	4.93	3.40	3.07	1.53	8.76	1.86	2.74	3.29	4.93	4.38	4.27	1.42	2.63	3.83	3.07
fars	4.09	4.09	0.26	5.12	4.86	53.71	9.97	3.32	95.40	14.83	3.58	1.02	14.83	4.35	3.32	12.79	4.60	3.58	5.63	5.12	1.28	8.95	1.53
fren	4.64	1.80	0.00	50.26	4.90	5.67	2.06	2.32	9.02	96.39	6.19	1.29	6.96	2.06	1.55	5.67	15.46	7.99	7.73	4.64	1.29	3.35	1.29
geor	4.27	4.77	0.00	2.76	13.82	9.55	5.78	2.01	4.77	2.76	95.23	0.25	12.31	1.76	1.26	5.78	2.51	13.07	6.03	3.02	4.77	8.54	2.01
haus	5.80	1.45	0.58	2.32	2.61	3.48	2.32	2.32	3.19	2.32	3.77	94.20	4.93	4.06	2.03	15.65	7.25	8.12	4.64	0.29	2.61	3.48	1.74
hind	4.19	0.90	0.75	0.75	2.25	4.19	34.73	3.74	6.44	3.89	8.38	2.40	94.01	2.10	1.95	12.43	1.50	4.34	7.04	1.95	0.45	79.04	2.99
kore	7.28	1.77	2.65	1.32	1.10	12.36	10.15	0.66	5.52	3.31	0.88	2.21	6.62	93.82	8.83	14.35	0.66	5.08	15.89	1.32	1.99	7.73	3.75
mand	0.68	0.49	6.42	0.19	1.17	1.65	4.77	1.85	2.63	2.43	0.39	3.02	2.43	7.88	98.25	1.36	1.46	2.63	2.63	0.58	0.58	0.97	1.85
pash	15.72	3.35	0.77	2.58	5.15	46.13	2.32	3.35	29.90	7.73	17.78	9.28	17.27	5.15	2.06	93.81	8.25	17.78	9.02	3.61	5.67	22.68	1.55
port	1.77	3.83	0.00	6.19	10.62	12.68	1.77	2.06	6.19	17.40	1.77	3.24	4.13	1.18	0.88	10.62	97.35	10.91	11.21	3.24	3.83	2.95	0.59
russ	4.21	3.25	0.19	4.02	11.09	3.25	9.37	4.40	2.68	5.16	7.46	1.34	7.46	2.29	1.91	5.93	6.69	96.94	4.78	1.53	26.58	5.54	0.38
span	6.22	22.16	0.27	10.81	31.35	4.86	3.51	1.62	4.05	5.68	4.59	0.54	4.05	1.89	0.54	5.14	12.97	5.14	95.95	0.27	2.70	2.70	0.27
turk	3.57	0.51	0.26	1.02	0.26	3.57	1.02	0.51	8.67	6.63	5.61	0.00	3.06	1.53	0.51	3.32	1.28	3.57	1.79	97.70	0.51	3.83	0.26
ukra	2.40	18.93	0.27	4.00	35.47	4.53	1.33	1.33	3.73	8.00	7.20	1.87	5.33	3.20	1.33	11.73	10.93	78.93	8.00	5.07	63.47	5.60	0.00
urdu	1.62	1.35	0.27	2.16	1.08	4.85	12.40	1.62	8.09	1.08	6.47	1.35	94.34	1.89	0.81	14.29	2.43	2.43	3.77	2.70	1.08	91.11	1.08
viet	0.00	0.00	8.25	0.00	0.00	0.63	4.76	1.59	0.95	0.32	0.32	2.86	2.22	2.54	1.90	1.90	0.32	0.63	2.22	0.32	0.00	0.63	97.78

Table C.6: Confusion matrix of the 30 s task of 2009 NIST LRE database for the prosodic system with formant information - Prior equal to 0.5 and development dataset with utterances at least 30 s long. Numbers represent % of files.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	90.38	5.77	2.64	7.69	5.53	16.59	7.21	4.33	11.54	14.66	6.01	9.62	18.27	4.57	3.61	15.62	5.53	10.10	12.02	7.69	4.33	11.78	8.41
bosn	8.00	70.67	3.47	10.13	79.20	17.60	7.47	10.67	10.13	19.47	12.00	1.60	17.87	4.80	4.00	14.40	20.53	21.33	17.33	8.27	20.80	12.80	3.20
cant	1.43	0.29	88.54	2.87	1.15	7.74	16.62	8.88	5.44	8.60	1.72	3.72	7.45	6.30	24.07	2.58	1.72	3.44	9.74	0.86	2.29	4.58	24.36
creo	7.14	6.57	2.29	82.86	9.43	8.57	8.86	7.14	6.86	66.29	7.43	6.86	16.29	5.71	5.43	10.86	15.43	12.86	26.29	6.86	4.86	10.00	6.86
croa	4.82	68.02	2.54	6.35	77.66	17.01	8.12	5.84	9.64	15.23	7.61	2.03	12.18	2.28	1.78	13.71	15.99	19.04	15.99	8.88	21.57	9.14	3.30
dari	14.74	7.86	5.90	7.86	7.37	85.01	8.60	4.67	63.64	22.11	10.81	6.88	19.90	5.65	5.90	38.08	8.85	10.81	13.51	10.57	5.16	15.72	6.63
engi	7.63	3.91	5.87	3.13	3.52	12.92	86.30	47.16	11.15	14.87	4.50	2.94	35.42	2.74	5.48	8.22	9.20	9.00	14.29	4.70	7.24	24.66	8.61
engl	3.58	1.96	4.16	2.42	3.23	7.16	42.73	91.92	7.97	7.16	3.93	3.35	10.16	2.08	6.24	4.73	6.24	6.70	5.77	5.08	4.73	7.74	5.31
fars	3.38	5.97	2.86	8.57	4.68	65.45	10.39	7.01	89.61	21.56	6.23	5.71	14.55	4.94	5.45	17.40	7.01	4.68	8.05	12.73	3.90	11.95	3.64
fren	9.70	6.22	3.73	55.97	7.96	18.16	8.96	6.47	11.94	91.04	5.47	5.22	15.42	6.22	7.21	14.93	20.40	12.69	18.91	10.70	9.95	12.44	4.73
geor	10.17	11.17	1.99	8.44	20.84	11.66	8.44	5.96	11.41	16.38	86.35	2.73	16.87	5.21	3.72	13.40	7.44	18.36	12.16	9.68	10.17	10.42	3.72
haus	20.82	3.20	4.58	8.70	2.97	15.79	8.70	10.07	10.53	19.22	2.29	80.55	16.48	7.55	8.01	38.44	11.67	14.65	13.50	3.43	4.81	9.38	11.44
hind	9.84	3.44	7.03	3.75	5.47	14.37	30.94	10.31	12.66	9.22	9.22	4.38	86.09	4.53	5.62	19.53	6.88	8.75	15.47	6.88	7.19	80.47	6.25
kore	11.04	2.60	15.58	4.33	4.11	16.02	12.99	6.06	10.61	14.29	3.25	7.58	13.85	85.28	16.45	17.97	3.25	5.41	15.80	5.19	1.95	7.14	11.04
mand	3.48	2.05	19.55	2.46	2.46	7.06	11.46	7.47	8.50	12.38	1.74	6.14	7.57	13.61	90.28	6.65	4.40	6.04	6.45	2.66	4.40	4.81	9.31
pash	15.27	9.11	1.97	5.67	8.87	43.35	8.13	9.36	24.14	17.24	13.55	16.01	31.03	6.65	3.94	81.53	11.33	17.98	13.79	11.58	13.05	27.59	4.68
port	7.19	9.73	2.11	18.60	11.63	19.24	9.73	6.98	9.30	37.42	4.02	4.65	12.26	2.75	6.77	21.99	86.89	17.97	21.78	8.88	9.51	7.19	5.07
russ	5.17	5.79	0.62	7.23	10.74	8.47	11.36	7.23	5.37	15.91	8.88	3.10	10.95	2.89	6.20	10.54	13.22	88.43	10.74	8.88	49.38	8.06	3.31
span	6.47	11.44	2.99	11.44	19.15	9.95	9.95	5.97	6.22	18.91	4.48	3.98	13.18	2.74	2.99	9.45	12.19	7.21	88.81	3.48	5.22	7.96	6.22
turk	9.60	4.29	2.02	8.33	6.31	23.48	12.37	7.58	23.99	28.79	10.10	3.54	18.18	5.05	6.06	20.71	11.62	13.13	9.34	80.30	6.31	14.39	4.04
ukra	5.67	19.21	1.48	9.36	30.79	15.52	7.14	5.17	9.85	18.47	10.34	5.17	11.82	2.46	5.42	18.23	17.00	69.95	13.79	9.61	64.53	9.61	2.71
urdu	4.88	7.71	4.37	7.71	5.40	16.71	17.74	5.14	12.34	11.31	5.40	6.43	87.15	2.06	4.11	29.56	9.00	5.91	12.08	6.43	7.20	83.55	6.17
viet	3.89	1.77	29.33	1.77	1.77	8.48	16.96	8.83	5.30	8.13	1.41	4.95	9.89	4.59	9.89	5.65	2.83	2.12	9.89	1.41	1.06	5.30	90.81

Table C.7: Confusion matrix of the 3 s task of 2009 NIST LRE database for the fusion system - Prior equal to 0.5. Numbers represent % of files. Note that these results are the same for the matched and unmatched experiments.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	98.48	1.01	0.25	1.52	0.00	2.78	2.03	0.25	1.01	2.28	0.00	2.28	2.78	0.51	0.76	2.28	1.01	1.52	1.52	1.77	0.51	2.28	0.76
bosn	0.56	85.79	0.00	2.79	84.96	9.19	0.84	0.84	2.51	5.29	5.01	0.28	6.69	1.67	0.28	7.52	10.03	13.93	3.90	3.62	14.48	4.74	0.00
cant	0.00	0.00	97.98	0.00	0.00	0.29	2.59	1.44	0.00	0.29	0.29	0.29	0.86	1.73	7.49	0.58	0.00	0.00	0.86	0.00	0.00	0.86	5.48
creo	0.32	0.00	0.32	96.79	0.32	0.32	0.96	1.92	0.00	38.14	2.24	0.64	2.24	2.56	0.64	0.64	6.09	2.24	9.94	1.28	0.96	1.92	1.28
croa	0.54	75.68	0.00	1.08	89.19	7.30	1.35	0.81	2.70	4.59	4.32	0.00	4.05	0.00	0.00	5.68	10.81	12.97	7.03	2.70	13.51	2.70	0.00
dari	4.38	1.55	1.03	2.58	0.52	92.27	3.09	1.80	64.43	7.99	5.67	2.84	5.41	1.03	0.77	19.85	2.06	3.35	4.38	4.12	2.06	5.93	2.84
engi	1.31	0.19	2.44	1.13	0.00	4.50	94.56	40.53	2.63	4.69	0.75	0.56	24.20	0.56	2.25	2.63	2.06	2.44	3.75	0.94	1.69	14.82	3.00
engl	0.72	0.12	0.60	0.36	0.12	1.32	29.19	98.44	1.08	2.03	0.72	0.60	2.75	0.60	0.96	1.20	0.96	1.67	1.44	0.60	0.84	1.67	0.84
fars	1.04	0.78	0.52	1.04	0.52	43.60	2.61	1.04	99.48	6.01	1.31	0.26	4.18	0.26	0.78	3.13	1.04	0.78	2.35	3.13	0.52	2.61	0.26
fren	3.04	0.51	0.00	60.25	0.51	4.05	1.27	0.76	2.78	97.47	1.77	1.01	3.80	1.01	0.51	3.80	8.10	3.80	6.33	2.78	4.30	3.04	1.27
geor	0.76	4.04	0.25	1.77	4.80	2.78	1.52	1.26	2.53	4.29	97.22	0.76	4.80	1.26	0.51	4.55	1.26	6.57	2.78	2.53	3.79	2.78	0.25
haus	8.57	0.78	0.00	0.78	0.00	3.64	2.60	1.30	1.56	2.60	0.52	97.92	3.38	0.78	0.52	18.18	3.12	3.90	1.56	0.26	0.52	2.08	1.56
hind	3.42	0.00	1.63	1.14	0.16	3.91	15.80	1.47	1.47	1.63	3.42	1.63	97.88	1.30	0.81	6.19	1.63	1.47	5.21	2.44	1.47	94.14	0.98
kore	3.78	0.00	5.33	0.67	0.00	4.44	3.11	0.89	2.67	2.67	0.00	2.89	3.56	98.67	3.56	6.67	0.22	0.67	5.33	0.89	0.22	2.22	2.00
mand	0.21	0.00	5.46	0.10	0.00	0.51	2.16	0.51	0.62	1.24	0.21	1.44	0.93	3.91	99.18	1.24	0.31	0.62	0.51	0.51	0.00	0.62	1.24
pash	9.97	3.07	0.51	1.53	2.05	40.15	2.30	2.05	13.55	7.93	8.18	6.39	15.35	1.79	0.51	91.30	4.09	12.53	3.84	5.63	7.42	16.37	1.28
port	1.32	1.32	0.26	4.22	2.37	4.75	0.53	1.32	2.11	14.25	0.53	1.58	1.32	0.26	0.53	4.75	98.94	6.60	6.86	1.85	4.22	1.85	0.26
russ	0.63	0.21	0.00	0.21	1.04	1.67	2.30	1.25	0.84	2.30	1.67	0.63	0.84	0.84	1.46	1.25	2.92	99.58	0.63	1.04	36.33	0.63	0.42
span	0.78	4.18	0.00	4.70	5.22	2.09	2.35	1.31	0.52	4.96	0.26	0.00	2.87	0.26	0.00	3.13	5.22	2.09	99.48	0.26	0.78	1.57	0.26
turk	1.52	0.25	0.00	1.78	0.00	6.35	1.27	0.51	8.38	7.36	3.05	0.25	3.81	1.02	1.02	2.79	2.54	3.05	0.25	96.19	1.02	3.05	0.51
ukra	0.52	9.14	0.00	2.87	15.40	3.66	0.78	1.04	1.57	7.57	4.96	2.09	2.61	0.52	0.52	3.66	5.22	79.63	4.18	4.18	85.12	2.61	0.00
urdu	1.61	0.27	0.80	1.07	0.00	5.90	6.97	0.27	2.95	2.95	3.22	0.80	96.78	0.54	0.80	10.99	2.68	1.61	2.95	2.14	0.54	96.78	0.80
viet	0.00	0.00	12.86	0.36	0.00	1.79	3.93	1.07	2.14	2.14	0.00	1.43	1.79	0.00	2.50	1.79	0.36	0.71	1.43	0.71	0.00	1.07	98.93

Table C.8: Confusion matrix of the 10 s task of 2009 NIST LRE database for the fusion system - Prior equal to 0.5 and development dataset with utterances at least 10 s long. Numbers represent % of files.

	amha	bosn	cant	creo	croa	dari	engi	engl	fars	fren	geor	haus	hind	kore	mand	pash	port	russ	span	turk	ukra	urdu	viet
amha	98.69	1.04	0.00	0.26	0.00	0.52	0.78	0.00	0.26	0.00	0.00	0.26	0.52	0.00	0.00	0.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bosn	0.00	97.58	0.00	0.00	93.66	1.81	0.30	0.00	0.00	0.91	2.11	0.00	1.21	0.00	0.00	1.81	3.02	7.85	0.91	0.30	7.85	0.30	0.00
cant	0.27	0.00	99.73	0.00	0.00	0.00	0.80	0.00	0.00	0.27	0.00	0.00	0.27	0.00	1.87	0.00	0.00	0.00	0.27	0.00	0.00	0.00	0.80
creo	0.65	0.00	0.00	99.35	0.00	0.00	0.00	0.00	0.00	16.61	0.00	0.00	0.33	0.00	0.33	0.65	0.98	0.65	4.23	0.00	0.00	0.00	0.00
croa	0.00	74.45	0.00	0.00	96.43	1.92	0.55	0.00	0.82	1.10	0.55	0.00	1.10	0.00	0.00	1.65	4.67	6.59	2.20	0.00	7.69	0.00	0.00
dari	0.54	0.81	0.27	0.27	0.00	93.01	1.08	0.00	62.90	1.88	2.69	1.88	3.76	0.00	0.00	9.14	0.00	0.27	0.81	0.00	0.00	3.23	0.27
engi	0.52	0.00	0.00	0.00	0.00	0.52	97.76	30.69	0.52	2.76	0.00	0.00	10.34	0.00	0.34	0.69	0.17	0.17	0.52	0.00	0.00	3.97	0.17
engl	0.11	0.11	0.22	0.00	0.22	0.55	23.33	98.90	0.22	0.33	0.00	0.11	1.20	0.00	0.33	0.33	0.55	0.33	0.22	0.00	0.22	0.44	0.44
fars	0.00	0.00	0.00	0.00	0.00	23.02	0.51	0.26	100.00	0.77	0.26	0.00	0.51	0.00	0.26	0.77	0.00	0.00	0.00	0.26	0.00	0.00	0.00
fren	1.29	0.00	0.00	50.77	0.00	0.52	0.26	0.00	0.26	98.97	0.52	0.00	0.26	0.26	0.00	0.26	2.32	0.77	1.29	0.00	0.26	0.26	0.00
geor	0.75	1.76	0.00	0.25	1.76	1.51	0.75	0.25	0.50	1.01	99.50	0.00	2.01	0.00	0.00	2.51	0.75	2.26	0.75	0.25	0.75	0.75	0.00
haus	4.64	0.00	0.00	0.29	0.00	0.87	0.29	0.29	0.58	0.00	0.00	98.84	0.29	0.29	0.29	8.41	0.29	0.29	0.58	0.00	0.00	0.00	0.00
hind	0.75	0.00	0.00	0.00	0.00	0.60	14.22	1.20	0.15	0.75	0.15	0.00	98.20	0.15	0.75	2.25	0.00	0.15	0.45	0.15	0.15	94.76	0.30
kore	0.88	0.00	0.22	0.22	0.00	0.44	0.22	0.00	0.66	0.44	0.00	0.00	0.22	99.78	0.44	0.88	0.00	0.22	0.88	0.00	0.00	0.00	0.22
mand	0.00	0.10	0.97	0.00	0.00	0.19	0.19	0.10	0.00	0.10	0.00	0.19	0.29	0.10	100.00	0.58	0.10	0.00	0.10	0.10	0.00	0.19	0.19
pash	9.79	0.77	0.26	0.00	0.00	31.70	0.00	0.52	7.99	2.32	2.32	1.03	5.15	0.26	0.00	97.94	0.52	5.93	1.29	0.26	2.58	4.12	0.00
port	0.29	0.29	0.00	0.59	0.00	1.47	0.00	0.00	0.29	5.31	0.00	0.00	0.59	0.00	0.00	1.77	100.00	0.88	2.06	0.29	1.18	0.59	0.00
russ	0.19	0.00	0.00	0.19	0.00	0.00	0.38	0.19	0.00	0.57	0.19	0.00	0.76	0.00	0.00	0.19	0.19	100.00	0.19	0.00	11.66	0.19	0.00
span	0.00	0.81	0.00	0.81	0.27	0.27	0.27	0.00	0.00	0.81	0.00	0.00	0.00	0.00	0.00	0.00	0.54	0.00	100.00	0.00	0.27	0.00	0.00
turk	0.51	0.00	0.00	0.26	0.00	1.53	0.00	0.00	1.79	0.77	1.02	0.00	0.51	0.00	0.00	1.02	0.51	0.51	0.00	99.49	0.51	0.51	0.00
ukra	0.27	6.93	0.00	0.80	11.73	0.53	0.27	0.27	0.53	2.40	1.87	0.00	1.33	0.27	0.00	1.87	2.67	85.07	2.93	0.80	90.67	1.33	0.00
urdu	0.00	0.00	0.00	0.00	0.00	1.35	2.70	0.54	0.27	0.00	0.54	0.00	100.00	0.00	0.00	4.31	0.27	0.00	0.27	0.00	0.00	98.65	0.00
viet	0.32	0.00	5.40	0.00	0.00	0.00	1.90	0.63	0.00	1.27	0.00	0.00	0.63	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.32	99.05

Table C.9: Confusion matrix of the 30 s task of 2009 NIST LRE database for the fusion system - Prior equal to 0.5 and development dataset with utterances at least 30 s long. Numbers represent % of files.

APPENDIX D

2009 NIST LRE Datasets

In this Appendix we include the training, development, and testing data, used in the experiments with the 2009 NIST LRE dataset reported in Chapter 9.

Table D.1 contains target language data distribution of the 2009 NIST LRE test dataset according to the file duration.

Table D.2 shows the databases used for training with number of files and time of speech (after VAD) for each language.

Table D.3 shows the number of files and hours of speech (after VAD) of the databases and languages used in our experiments for development.

Language	3 s	10 s	30 s	Total
Amharic	416	395	383	1194
Bosnian	375	359	331	1065
Cantonese	349	347	375	1071
Creole Haitian	350	312	307	969
Croatian	394	370	364	1128
Dari	407	388	372	1167
Indian English	511	533	580	1624
American English	866	836	913	2615
Farsi	385	383	391	1159
French	402	395	388	1185
Georgian	403	396	398	1197
Hausa	437	385	345	1167
Hindi	640	614	668	1922
Korean	462	450	453	1365
Mandarin	977	971	1028	2976
Pashto	406	391	388	1185
Portuguese	473	379	339	1191
Russian	484	479	523	1486
Spanish	402	383	370	1155
Turkish	396	394	392	1182
Ukrainian	406	383	375	1164
Urdu	389	373	371	1133
Vietnamese	283	280	315	878
Total	10613	10196	10369	31178

Table D.1: Data distribution in 2009 NIST LRE database - Number of files belonging to the target languages of 2009 NIST LRE database among the 3 s, 10 s, and 30 s tasks.

	Total		CALLFR		OHSU		SRE04		SRE06		SRE08		SRE10		SWCHBR		LRE03		LRE05		LRE07t		LRE11d		LRE11e		VOA3	
Language	**#**	**hrs**	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs	#	hrs
Amharic	**528**	**4.43**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	528	4.43
Bosnian	**149**	**1.58**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	149	1.58
Cantonese	**624**	**21.26**	0	0	0	0	0	0	2	0.07	486	17.14	0	0	0	0	0	0	0	0	40	2.63	0	0	0	0	96	1.40
Creole Haitian	**255**	**4.30**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	4.30
Croatian	**59**	**1.15**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	1.15
Dari	**1283**	**4.49**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1283	4.49
Am. English	**6097**	**79.51**	240	40.76	174	11.43	0	0	256	8.62	0	0	0	0	0	0	2614	7.12	2464	6.96	0	0	0	0	0	0	349	4.60
In. English	**1008**	**9.93**	0	0	88	5.77	0	0	0	0	0	0	0	0	0	0	0	0	533	1.45	0	0	0	0	387	2.70	0	0
Farsi	**1445**	**31.27**	120	22.54	0	0	0	0	1	0.02	56	2.00	0	0	0	0	717	2.00	0	0	0	0	0	0	0	0	551	4.69
French	**1432**	**27.28**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	711	1.93	0	0	0	0	0	0	0	0	601	4.45
Georgian	**227**	**3.76**	0	0	0	0	0	0	0	0	1	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	3.72
Hausa	**760**	**4.30**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	760	4.30
Hindi	**1861**	**43.46**	120	20.58	72	4.72	0	0	23	0.83	288	9.84	0	0	0	0	705	1.89	430	1.20	0	0	0	0	0	0	223	4.38
Korean	**2638**	**42.62**	120	18.63	162	9.33	0	0	3	0.13	183	5.90	0	0	0	0	710	1.81	949	2.39	0	0	0	0	0	0	511	4.404
Mandarin	**7397**	**141.10**	240	42.68	324	21.94	302	10.92	1304	46.10	114	3.74	0	0	0	0	1178	3.21	2954	8.10	0	0	0	0	0	0	981	4.38
Pashto	**1321**	**4.43**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1321	4.43
Portuguese	**554**	**4.25**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	554	4.25
Russian	**1669**	**32.57**	0	0	0	0	326	11.59	120	4.10	256	9.11	0	0	0	0	240	0.67	0	0	40	2.64	0	0	0	0	687	4.44
Spanish	**3974**	**73.85**	239	45.93	130	8.41	145	5.14	37	1.24	6	0.20	0	0	0	0	1160	3.25	1851	5.11	0	0	0	0	0	0	406	4.52
Turkish	**541**	**6.64**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	276	1.86	265	4.77
Ukrainian	**606**	**2.42**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	557	1.69	49	0.72
Urdu	**1209**	**10.76**	0	0	0	0	0	0	8	0.28	0	0	0	0	0	0	0	0	0	0	40	2.68	0	0	478	3.23	683	4.56
Vietnamese	**1534**	**38.75**	120	21.28	0	0	0	0	0	0	350	11.52	0	0	0	0	707	1.86	0	0	0	0	0	0	0	0	357	4.07
Albanian	**268**	**2.74**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	268	2.74
Arabic	**1829**	**48.21**	120	19.68	0	0	382	14.80	173	6.29	9	0.34	0	0	0	0	705	1.88	0	0	40	2.72	400	2.47	0	0	0	0
Azerbaijani	**403**	**4.51**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	403	4.51
Bengali	**683**	**13.52**	0	0	0	0	0	0	129	4.31	51	1.78	0	0	0	0	0	0	0	0	40	2.86	0	0	0	0	463	4.55
Burmese	**1035**	**4.52**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1035	4.52
Czech	**100**	**1.32**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0.66	0	0	0	0
English Others	**34024**	**1168.38**	0	0	0	0	3361	118.11	10258	343.41	9163	299.75	5111	166.19	4856	236.44	0	0	0	0	0	0	0	0	0	0	1275	4.45
German	**1137**	**27.82**	120	22.27	42	2.84	0	0	0	0	0	0	0	0	0	0	718	2.01	257	0.69	0	0	0	0	0	0	0	0
Greek	**126**	**1.46**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	126	1.46
Indonesian	**394**	**4.58**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	394	4.58
Italian	**50**	**1.70**	0	0	0	0	0	0	0	0	50	1.70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Japanese	**2599**	**45.45**	120	20.03	184	11.55	0	0	0	0	240	8.56	0	0	0	0	954	2.39	1101	2.91	0	0	0	0	0	0	0	0
Khmer	**1086**	**4.34**	0	0	0	0	0	0	0	0	2	0.06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1084	4.27
Kru	**412**	**4.41**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	412	4.41
Kurdish	**515**	**4.41**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	515	4.41
Lao	**165**	**1.86**	0	0	0	0	0	0	0	0	5	0.15	0	0	0	0	0	0	0	0	0	0	93	0.50	0	0	67	0.69
Macedonian	**64**	**0.80**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	0.80
Ndebele	**374**	**4.50**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	374	4.50
Oromo	**432**	**4.48**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	432	4.48
Polish	**100**	**1.36**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0.68	0	0	0	0
Punjabi	**100**	**1.28**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0.64	0	0	0	0
Serbian	**404**	**3.20**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	404	3.20
Shanghai Wu	**182**	**8.41**	0	0	0	0	0	0	0	0	142	5.46	0	0	0	0	0	0	0	0	40	2.95	0	0	0	0	0	0
Shona	**371**	**4.65**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	371	4.65
Slovak	**100**	**1.34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0.67	0	0	0	0
Somali	**716**	**4.61**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	716	4.61
Min Chinese	**164**	**7.34**	0	0	0	0	0	0	0	0	124	4.54	0	0	0	0	0	0	0	0	40	2.79	0	0	0	0	0	0
Swahili	**523**	**4.52**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	523	4.52
Tagalog	**50**	**1.67**	0	0	0	0	0	0	0	0	50	1.67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tamil	**1455**	**28.57**	120	18.56	92	6.52	0	0	0	0	1	0.03	0	0	0	0	688	1.89	554	1.55	0	0	0	0	0	0	0	0
Thai	**830**	**21.46**	0	0	0	0	0	0	7	0.23	492	16.40	0	0	0	0	0	0	0	0	40	2.62	0	0	0	0	291	2.20
Tibetan	**534**	**4.38**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	534	4.38
Tigre	**302**	**4.32**	0	0	0	0	0	0	0	0	1	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	301	4.30
Uzbek	**617**	**4.6**	0	0	0	0	0	0	0	0	12	0.47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	605	4.28

Table D.2: Train data for 2009 NIST LRE database - Target languages in bold.

	Total		LRE07e		LRE11		VOA3	
Language	**#**	**hrs**	#	hrs	#	hrs	#	hrs
Amharic	**1500**	**4.13**	0	0	0	0	1500	4.13
Bosnian	**800**	**1.41**	0	0	0	0	800	1.41
Cantonese	**877**	**1.88**	240	0.90	0	0	637	0.98
Creole Haitian	**1345**	**3.23**	0	0	0	0	1345	3.23
Croatian	**678**	**1.21**	0	0	0	0	678	1.21
Dari	**2703**	**7.13**	0	0	1205	2.93	1498	4.19
Am. English	**3048**	**8.32**	240	0.88	1356	3.43	1452	4.00
In. English	**480**	**1.70**	480	1.70	0	0	0	0
Farsi	**2954**	**9.30**	240	1.03	1214	3.72	1500	4.53
French	**1739**	**4.99**	240	0.85	0	0	1499	4.13
Georgian	**1108**	**2.50**	0	0	0	0	1108	2.50
Hausa	**1500**	**4.04**	0	0	0	0	1500	4.04
Hindi	**3174**	**8.91**	479	1.67	1242	3.23	1453	4.00
Korean	**1596**	**4.22**	240	0.89	0	0	1356	3.33
Mandarin	**1973**	**5.91**	474	1.77	0	0	1499	4.14
Pashto	**1499**	**4.22**	0	0	0	0	1499	4.22
Portuguese	**1500**	**4.14**	0	0	0	0	1500	4.14
Russian	**3302**	**10.20**	480	1.69	1322	4.32	1500	4.18
Spanish	**2217**	**6.75**	720	2.60	0	0	1497	4.15
Turkish	**1276**	**3.09**	0	0	0	0	1276	3.09
Ukrainian	**1069**	**2.54**	0	0	557	1.69	512	0.84
Urdu	**3173**	**9.65**	240	0.86	1433	4.50	1500	4.28
Vietnamese	**1554**	**3.71**	480	1.58	0	0	1074	2.13
Albanian	**854**	**1.70**	0	0	0	0	854	1.70
Arabic	**240**	**0.75**	240	0.75	0	0	0	0
Azerbaijani	**1475**	**4.19**	0	0	0	0	1475	4.19
Bengali	**1740**	**5.11**	240	0.91	0	0	1500	4.19
Burmese	**1500**	**4.13**	0	0	0	0	1500	4.13
English Others	**1473**	**4.24**	0	0	0	0	1473	4.24
German	**240**	**0.82**	240	0.82	0	0	0	0
Greek	**770**	**1.31**	0	0	0	0	770	1.31
Indonesian	**1554**	**4.10**	240	0.78	0	0	1314	3.31
Italian	**240**	**1.04**	240	1.04	0	0	0	0
Japanese	**240**	**0.87**	240	0.87	0	0	0	0
Khmer	**1500**	**3.84**	0	0	0	0	1500	3.84
Kru	**1500**	**4.08**	0	0	0	0	1500	4.08
Kurdish	**1500**	**4.33**	0	0	0	0	1500	4.33
Lao	**360**	**0.41**	0	0	0	0	360	0.41
Macedonian	**716**	**1.11**	0	0	0	0	716	1.11
Ndebele	**1500**	**4.23**	0	0	0	0	1500	4.23
Oromo	**1499**	**4.30**	0	0	0	0	1499	4.30
Punjabi	**96**	**0.37**	96	0.37	0	0	0	0
Serbian	**798**	**1.54**	0	0	0	0	798	1.54
Shanghai Wu	**240**	**0.89**	240	0.89	0	0	0	0
Shona	**1499**	**4.38**	0	0	0	0	1499	4.38
Somali	**1500**	**4.13**	0	0	0	0	1500	4.13
Min Chinese	**240**	**0.83**	240	0.83	0	0	0	0
Swahili	**1500**	**4.22**	0	0	0	0	1500	4.22
Tagalog	**240**	**0.89**	240	0.89	0	0	0	0
Tamil	**480**	**1.60**	480	1.60	0	0	0	0
Thai	**892**	**2.00**	240	0.86	0	0	652	1.14
Tibetan	**1343**	**3.07**	0	0	0	0	1343	3.07
Tigre	**1459**	**3.84**	0	0	0	0	1459	3.84
Uzbek	**1398**	**3.47**	0	0	0	0	1398	3.47

Table D.3: Development data for 2009 NIST LRE database - Target languages in bold.

References

Adami, A. G. and Hermansky, H. (2003). Segmentation of Speech for Speaker and Language Recognition. In *INTERSPEECH*, pages 841–844, Geneva, Switzerland.

Ambikairajah, E., Li, H., Wang, L., Yin, B., and Sethu, V. (2011). Language Identification: A Tutorial. *IEEE Circuits and Systems Magazine*, 11(2):82–108.

Andersen, O., Dalsgaard, P., and Barry, W. (1994). On the Use of Data-Driven Clustering Technique for Identification of Poly- and Mono-Phonemes for Four European Languages. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/121–I/124 vol.1, Adelaide, Australia.

Armstrong, J. S. and Collopy, F. (1992). Error Measures for Generalizing About Forecasting Methods : Empirical Comparisons. *International Journal of Forecasting*, 8(1):69–80.

Bamberg, P. (1981). Vocal Tract Normalization. Technical report, Verbex.

Barras, C. and Gauvain, J.-L. (2003). Feature and Score Normalization for Speaker Verification of Cellular Data. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages II/49–II/52. Ieee.

Bartholomew, D. J., Knott, M., and Moustaki, I. (2011). *Latent Variable Models and Factor Analysis: A Unified Approach*. John Wiley & Sons, third edition.

Beech, J. R., Harding, L., and Hilton-Jones, D. (1993). *Assessment in Speech and Language Therapy*. Routledge, first edition.

Berkling, K. M. and Barnard, E. (1995). Theoretical Error prediction for a Language Identification System Using Optimal Phoneme Clustering. In *EUROSPEECH*, pages 351–354, Madrid, Spain.

Beukelman, D. R. and Yorkston, K. M. (1979). The Relationship Between Information Transfer and Speech Intelligibility of Dysarthric Speakers. *Journal of Communication Disorders*, 12(3):189–196.

Bielefeld, B. (1994). Language Identification Using Shifted Delta Cepstrum. In *Fourteenth Annual Speech Research Symposium*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Series in Information Science and Statistics, first edition.

Bocklet, T., Haderlein, T., Hönig, F., Rosanowski, F., and Nöth, E. (2009). Evaluation and Assessment of Speech Intelligibility on Pathological Voices Based Upon Acoustic Speaker Models. In *3rd Advanced Voice Function Assessment International Workshop*, pages 89–92, Madrid, Spain.

Bocklet, T., Riedhammer, K., Nöth, E., Eysholdt, U., and Haderlein, T. (2012). Automatic Intelligibility Assessment of Speakers After Laryngeal Cancer by Means of Acoustic Modeling. *Journal of Voice*, 26(3):390–397.

Boonsuk, S., Suchato, A., Punyabukkana, P., Wutiwiwatchai, C., and Thatphithakkul, N. (2014). Language Recognition Using Latent Dynamic Conditional Random Field Model with Phonological Features. *Mathematical Problems in Engineering*, 2014.

Brümmer, N. (2007). FoCal Multi-Class: Toolkit for Evaluation , Fusion and Calibration of Multi-Class Recognition Scores. Available from: `https://sites.google.com/site/nikobrummer/focalmulticlass`.

Brümmer, N. (2009a). EM4JFA. Technical report, AGNITiO. Available from: `https://sites.google.com/site/nikobrummer/EMforJFA.pdf`.

Brümmer, N. (2009b). The EM algorithm and Minimum Divergence. Technical report, AGNITiO. Available from: `https://sites.google.com/site/nikobrummer/EMandMINDIV.pdf`.

Brümmer, N. (2010). A Minimum Divergence Recipe for VBEM. Technical report, AGNITiO. Available from: `https://sites.google.com/site/nikobrummer/VBEMandMINDIV.pdf`.

Brümmer, N., Strasheim, A., Hubeika, V., Matejka, P., Burget, L., and Glembek, O. (2009). Discriminative Acoustic Language Recognition via Channel-Compensated GMM Statistics. In *INTERSPEECH*, pages 2187–2190, Brighton, UK.

Brümmer, N. and van Leeuwen, D. (2006). On Calibration of Language Recognition Scores. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, San Juan, Puerto Rico.

Burget, L., Matejka, P., and Cernocký, J. (2006). Discriminative Training Techniques for Acoustic Language Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/209–I/212, Toulouse, France.

Campbell, W., Gleason, T., Navratil, J., Reynolds, D., Shen, W., Singer, E., and Torres-Carrasquillo, P. (2006). Advanced language recognition using cepstra and phonotactics: MITLL system performance on the NIST 2005 language recognition evaluation. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, San Juan, Puerto Rico.

Campbell, W. M., Singer, E., Torres-Carrasquillo, P. A., and Reynolds, D. A. (2004). Language Recognition with Support Vector Machines. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, pages 285–288, Toledo, Spain.

Campbell, W. M., Sturim, D. E., Torres-Carrasquillo, P., and Reynolds, D. A. (2008). A Comparison of Subspace Feature-Domain Methods for Language Recognition. In *INTERSPEECH*, number 2, pages 309–312, Brisbane, Australia.

Carmichael, J. (2007). *Introducing Objective Acoustic Metrics for the Frenchay Dysarthria Assessment Procedure*. PhD thesis, University of Sheffield.

Carmichael, J. and Green, P. (2004). Revisiting Dysarthria Assessment Intelligibility Metrics. In *International Conference on Spoken Language Processing (ICSLP)*, Jeju Island, South Korea.

Castaldo, F., Colibro, D., Dalmasso, E., Laface, P., and Vair, C. (2007a). Acoustic Language Identification Using Fast Discriminative Training. In *INTERSPEECH*, pages 346–349, Antwerp, Belgium.

Castaldo, F., Colibro, D., Dalmasso, E., Laface, P., and Vair, C. (2007b). Compensation of Nuisance Factors for Speaker and Language Recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):1969–1978.

Castaldo, F., Cumani, S., Laface, P., and Colibro, D. (2009). Language Recognition Using Language Factors. In *INTERSPEECH*, number 2, pages 176–179, Brighton, UK.

Castaldo, F., Dalmasso, E., Laface, P., Colibro, D., and Vair, C. (2007c). Language Identification Using Acoustic Models and Speaker Compensated Cepstral-Time Matrices. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages IV/1013–IV/1016, Honolulu, HI, USA.

Castán, D., Vaquero, C., Ortega, A., Martínez, D., Villalba, J., and Lleida, E. (2011). Hierarchical Audio Segmentation with HMM and Factor Analysis in Broadcast News Domain. In *INTERSPEECH*, pages 421–424, Florence, Italy.

Chang, C.-C. and Lin, C.-J. (2002). Training v-Support Vector Regression: Theory and Algorithms. *Neural Computation*, 14(8):1959–1977.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27. Available from: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

Choi, D.-L., Kim, B.-W., Lee, Y.-J., Um, Y., and Chung, M. (2011). Design and Creation of Dysarthric Speech Database for Development of QoLT Software Technology. In *2011 International Conference on Speech Database and Assessments (Oriental COCOSDA)*, pages 47–50, Hsinchu, Taiwan.

Christensen, H., Aniol, M. B., Bell, P., Green, P., Hain, T., King, S., and Swietojanski, P. (2013). Combining In-Domain and Out-Of-Domain Speech Data for Automatic Recognition of Disordered Speech. In *INTERSPEECH*, pages 3642–3645, Lyon, France.

Christensen, H., Cunningham, S., Fox, C., Green, P., and Hain, T. (2012). A Comparative Study of Adaptive, Automatic Recognition of Disordered Speech. In *INTERSPEECH*, Portland, OR.

Cimarusti, D. and Ives, R. B. (1982). Development of an Automatic Identification System of Spoken Languages: Phase1. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1661–1663, Paris, France.

Comon, P. (1991). Independent Component Analysis. In *International Signal Processing Workshop on Higher Order Statistics*, pages 111–120, Chamrousse, France.

Cordoba, R., D'Haro, L. F., Fernandez-Martinez, F., Macias-Guarasa, J., and Ferreiros, J. (2007). Language Identification based on n-gram Frequency Ranking. In *INTERSPEECH*, pages 354–357, Antwerp, Belgium.

Cummins, F., Gers, F., and Schmidhuber, J. (1999). Language Identification from Prosody without Explicit Features. In *EUROSPEECH*, pages 371–374, Budapest, Hungary.

Dan, Q. and Bingxi, W. (2003). Discriminative Training of GMM for Language Identification. In *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, pages 67–70, Tokyo, Japan.

Darley, F. L., Aronson, A. E., and Brown, J. R. (1975). *Motor Speech Disorders.* WB Saunders, first edition.

Davis, S. B. and Mermelstein, P. (1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics Speech and Signal Processing*, 28(4):357–366.

De Bodt, M., Hernández-Díaz Huici, M., and Van De Heyning, P. H. (2002). Intelligibility as a Linear Combination of Dimensions in Dysarthric Speech. *Journal of Communication Disorders*, 35(3):283–292.

Dehak, N. (2009). *Discriminative and Generative Approaches for Long- and Short-Term Speaker Characteristics Modeling. Application to Speaker Verification.* PhD thesis, Université du Québec.

Dehak, N., Dehak, R., Glass, J., Reynolds, D., and Kenny, P. (2010). Cosine Similarity Scoring Without Score Normalization Techniques. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, Brno, Czech Republic.

Dehak, N., Dumouchel, P., and Kenny, P. (2007a). Modeling Prosodic Features With Joint Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):2095–2103.

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (2011a). Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Dehak, N., Kenny, P., and Dumouchel, P. (2007b). Continuous Prosodic Features and Formant Modeling with Joint Factor Analysis for Speaker Verification. In *INTERSPEECH*, number 2, pages 853–856, Antwerp, Belgium.

Dehak, N., Torres-Carrasquillo, P. A., Reynolds, D., and Dehak, R. (2011b). Language Recognition Via I-Vectors and Dimensionality Reduction. In *INTERSPEECH*, number August, pages 857–860, Florence, Italy.

Deller Jr., J. R., Liu, M. S., Ferrier, L. J., and Robichaud, P. (1993). The Whitaker Database of Dysarthric (Cerebral Palsy) Speech. *The Journal of the Acoustical Society of America*, 93(6):3516–3518.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

D'Haro, L. F., Cordoba, R., Salamea, C., and Echeverry, J. D. (2014). Extended Phone Log-Likelihood Ratio Features and Acoustic-Based I-Vectors for Language Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5342–5346, Florence, Italy.

D'Haro, L. F., Glembek, O., Plchot, O., Matejka, P., Soufifar, M., Cordoba, R., and Cernocký, J. (2012). Phonotactic Language Recognition Using i-Vectors and Phoneme Posteriogram Counts. In *INTERSPEECH*, pages 42–45, Portland, OR, USA.

Diez, M., Varona, A., Penagarikano, M., Rodriguez-Fuentes, L. J., and Bordel, G. (2012). On the use of Phone Log-Likelihood Ratios as Features in Spoken Language Recognition. In *Spoken Language Technology Workshop (SLT)*, pages 274–279, Miami, FL, USA.

Diez, M., Varona, A., Penagarikano, M., Rodriguez-fuentes, L. J., and Bordel, G. (2013). Dimensionality Reduction of Phone Log-Likelihood Ratio Features for Spoken Language Recognition. In *INTERSPEECH*, number 1, pages 64–68, Lyon, France.

Doyle, P. C., Leeper, H. A., Kotler, A.-L., Thomas-Stonell, N., O'Neill, C., Dylke, M.-C., and Rolls, K. (1997). Dysarthric Speech: a Comparison of Computerized Speech Recognition and Listener Intelligibility. *Journal of Rehabilitation Research and Development*, 34(3):309–316.

Drummond, S. S. (1993). *Dysarthria Examination Battery*. Communication Skill Builders, first edition.

Elordieta, G. (2008). An Overview of Theories of the Syntax-Phonology Interface. *International Journal of Basque Linguistics and Philology*, 42(1):209–286.

Enderby, P. (1983). *Frenchay Dysarthria Assessment*. College Hill Press, first edition.

Enderby, P. (2013). Disorders of Communication: Dysarthria. In *Handbook of Clinical Neurology*, chapter 22, pages 273–281. Elsevier B.V., 110 edition.

Enderby, P. and Emerson, J. (1995). *Does Speech and Language Therapy Work?* Wiley, first edition.

Falk, T. H., Chan, W.-Y., and Shein, F. (2012). Characterization of Atypical Vocal Source Excitation, Temporal Dynamics and Prosody for Objective Measurement of Dysarthric Word Intelligibility. *Speech Communication*, 54(5):622–631.

Falk, T. H., Hummel, R., and Chan, W.-Y. (2011). Quantifying Perturbations in Temporal Dynamics for Automated Assessment of Spastic Dysarthric Speech Intelligibility. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4480–4483, Prague, Czech Republic.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2007). LIBLINEAR: A Library for Large Linear Classification. Available from: `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

Farinas, J., Pellegrino, F., Rouas, J.-L., and Andre-Obrecht, R. (2002). Merging Segmental and Rhythmic for Automatic Language Identification. In *IEEE International*

Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages I/753–I/756, Orlando, FL, USA.

Ferguson, T. S. (1973). A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230.

Ferrer, L., Scheffer, N., and Shriberg, E. (2010). A Comparison of Approaches for Modeling Prosodic Features in Speaker Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4414–4417, Dallas, TX, USA.

Fisher, R. A. (1936). The Use of Multiple Measurements in Taxoonomic Problems. *Annals of Eugenics*, 7(2):179–188.

Foil, J. T. (1986). Language Identification Using Noisy Speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 861–864, Tokyo, Japan.

Ganapathy, S., Omar, M., and Pelecanos, J. (2012). Noisy Channel Adaptation in Language Identification. In *Spoken Language Technology Workshop (SLT)*, number Lid, pages 307–312, Miami, FL, USA.

Garcia-Romero, D. and Espy-Wilson, C. Y. (2011). Analysis of I-Vector Length Normalization in Speaker Recognition Systems. In *INTERSPEECH*, pages 249–252, Florence, Italy.

Gauvain, J.-L. and Lee, C.-H. (1994). Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Transactions os Speech and Audio Processing*, 2(2):291–298.

Gauvain, J. L., Messaoudi, A., and Schwenk, H. (2004). Language Recognition Using Phone Lattices. In *INTERSPEECH*, number 2, pages 25–28, Jeju Island, Korea.

Gay, T. (1978). Effect of Speaking Rate on Vowel Formant Movements. *The Journal of the Acoustical Society of America*, 63(1):223–230.

Ghahramani, Z. and Hinton, G. E. (1996). The EM Algorithm for Mixture of Factor Analysers. Technical report, University of Toronto. Available from: `http://mlg.eng.cam.ac.uk/zoubin/papers/tr-96-1.pdf`.

Gleason, T. and Zissman, M. (2001). Composite Background Models and Score Standardization for Language Identification Systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 529–532 vol.1, Salt Lake City, UT, USA.

Glembek, O., Burget, L., Dehak, N., Brümmer, N., and Kenny, P. (2009). Comparison of Scoring Methods Used in Speaker Recognition with Joint Factor Analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4057 – 4060, Taipei, Taiwan.

Glembek, O., Burget, L., Matějka, P., Karafiat, M., and Kenny, P. (2011). Simplification and Optimization of iVector Extraction. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, number c, pages 4516–4519, Prague, Czech Republic.

Glembek, O., Matejka, P., Burget, L., and Mikolov, T. (2008). Advances in Phonotactic Language Recognition. In *INTERSPEECH*, number 2, pages 743–746, Brisbane, Australia.

González V., R. A. and Bevilacqua R., J. A. (2012). Las Disartrias. *Revista Hospital Clínico Universidad de Chile*, 23:299–309.

Green, P., Carmichael, J., and Hatzis, A. (2003). Automatic Speech Recognition with Sparse Training Data for Dysarthric Speakers. In *INTERSPEECH*, pages 1189–1192, Geneva, Switzerland.

Gutiérrez, J., Rouas, J.-L., and André-Obrecht, R. (2004). Fusing Language Identification Systems Using Performance Confidence Indexes. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/385–I/388 vol.1, Montreal, QC, Canada.

Hamidi, F., Baljko, M., Livingston, N., and Spalteholz, L. (2010). CanSpeak: A Customizable Speech Interface for People with Dysarthric Speech. In *International Conference on Computers Helping People with Special Needs*, pages 605–612, Vienna, Austria.

Harbeck, S. and Ohler, U. (1999). Multigrams for language identification. In *EUROSPEECH*, pages 375–378, Budapest, Hungary.

Hatch, A. O., Kajarekar, S., and Stolcke, A. (2006). Within-Class Covariance Normalization for SVM-Based Speaker Recognition. In *INTERSPEECH*, Pittsburgh, PA, USA.

Hatch, A. O. and Stolcke, A. (2006). Generalized Linear Kernels for One-Versus-All Classification: Application to Speaker Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages V/585–V/588, Toulouse, France.

Hayes, B. (1989). The Prosodic Hierarchy in Meter. In *Phonetics and Phonology, Volume 1: Rhythm and Meter*, pages 201–260. Academic Press.

Hazen, T. J. and Zue, V. W. (1993). Automatic Language Identification Using a Segment-Based Approach. In *EUROSPEECH*, pages 1303–1306, Berlin, Germany.

Hazen, T. J. and Zue, V. W. (1994). Recent Improvements in an Approach to Segment-Based Automatic Language Identification. In *International Conference on Spoken Language Processing (ICSLP)*, number March, pages 1883–1886, Yokohama, Japan.

Hermansky, H. (1990). Perceptual Linear Predictive (PLP) Analysis of Speech. *The Journal of the Acoustical Society of America*, 87(4):1738–1752.

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

Hermansky, H., Variani, E., and Peddinti, V. (2013). Mean Temporal Distance: Predicting ASR Error from Temporal Properties of Speech Signal. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7423–7426, Vancouver, BC, Canada.

Hosom, J.-P., Kain, A. B., Mishra, T., van Santen, J. P., Fried-Oken, M., and Staehely, J. (2003). Intelligibility of Modifications to Dysarthric Speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I–924 – I–927, Hong Kong.

Hotelling, H. (1933). Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24(6):417–441.

House, A. S. and Neuburg, E. P. (1977). Toward Automatic Identification of the Language of an Utterance. I. Preliminary Methodological Considerations. *The Journal of the Acoustical Society of America*, 62(3):708–713.

Hubeika, V., Burget, L., Matejka, P., and Schwarz, P. (2008). Discriminative Training and Channel Compensation for Acoustic Language Recognition. In *INTERSPEECH*, pages 301–304, Brisbane, Australia.

Itahashi, S. and Du, L. (1995). Language Identification Based on Speech Fundamental Frequency. In *EUROSPEECH*, pages 1359–1362, Madrid, Spain.

Itahashi, S., Zhou, J. X., and Tanaka, K. (1994). Spoken Language Discrimination Using Speech Fundamental Frequency. In *International Conference on Spoken Language Processing (ICSLP)*, pages 1899–1902, Yokohama, Japan.

Jancik, Z., Plchot, O., Brümmer, N., Burget, L., Glembek, O., Hubeika, V., Karafiát, M., Matejka, P., Mikolov, T., Strasheim, A., and Cernocký, J. H. (2010). Data Selection and Calibration Issues in Automatic Language Recognition - Investigation with BUT-AGNITIO NIST LRE 2009 System. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number July, pages 215–221, Brno, Czech Republic.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer Series in Statistics, second edition.

Kadambe, S. and Hieronymus, J. L. (1995). Language identification with Phonological and Lexical Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, number Lid, pages 3507–3510 vol.5, Detroit, MI, USA.

Kain, A. B., Hosom, J.-P., Niu, X., van Santen, J. P., Fried-Oken, M., and Staehely, J. (2007). Improving the Intelligibility of Dysarthric Speech. *Speech Communication*, 49(9):743–759.

Kenny, P. (2006). Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms. Technical report. Available from: `www.crim.ca/perso/patrick.kenny/FAtheory.pdf`.

Kenny, P., Boulianne, G., and Dumouchel, P. (2000). Bayesian Adaptation Revisited. In *ISCA ITRW ASR*, pages 112–119, Paris, France.

Kenny, P., Boulianne, G., and Dumouchel, P. (2002). Maximum Likelihood Estimation of Eigenvoices and Residual Variances for Large Vocabulary Speech Recognition Tasks. In *INTERSPEECH*, Denver, CO, USA.

Kenny, P., Boulianne, G., and Dumouchel, P. (2005). Eigenvoice Modeling with Sparse Training Data. *IEEE Transactions on Speech and Audio Processing*, 13(3):345–354.

Kenny, P., Boulianne, G., Ouellet, P., and Dumouchel, P. (2004). Speaker Adaptation Using an Eigenphone Basis. *IEEE Transactions on Speech and Audio Processing*, 12(6):579–589.

Kenny, P., Boulianne, G., Ouellet, P., and Dumouchel, P. (2007). Joint Factor Analysis Versus Eigenchannels in Speaker Recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 15(4):1435–1447.

Kenny, P. and Dumouchel, P. (2004). Experiments in Speaker Verification Using Factor Analysis Likelihood Ratios. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, Toledo, Spain.

Kenny, P., Mihoubi, M., and Dumouchel, P. (2003). New MAP Estimators for Speaker Recognition. In *INTERSPEECH*, pages 2691–2694, Geneva, Switzerland.

Kent, R. D., Weismer, G., Kent, J. F., and Rosenbek, J. C. (1989). Toward Phonetic Intelligibility Testing in Dysarthria. *Journal of Speech and Hearing Disorders*, 54:482–499.

Kim, H., Hasegawa-Johnson, M., Perlman, A., Gunderson, J., Huang, T., Watkin, K., and Frame, S. (2008). Dysarthric Speech Database for Universal Access Research. In *INTERSPEECH*, pages 1741–1744, Brisbane, Australia.

Kim, J., Kumar, N., Tsiartas, A., Li, M., and Narayanan, S. S. (2015). Automatic Intelligibility Classification of Sentence-Level Pathological Speech. *Computer speech & language*, 29(1):132–144.

Kim, M. J. and Kim, H. (2012). Automatic Assessment of Dysarthric Speech Intelligibility Based on Selected Phonetic Quality Features. In *Computers Helping People with Special Needs*, volume 7383 of *Lecture Notes in Computer Science*, pages 447–450. Springer Berlin Heidelberg.

Kirchhoff, K. and Parandekar, S. (2001). Multi-Stream Statistical N-Gram Modeling with Application to Automatic Language Identification. In *INTERSPEECH*, number 2, pages 803–806, Aalborg, Denmark.

Kockmann, M., Burget, L., and Cernocký, J. (2010a). Investigations into prosodic syllable contour features for speaker recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4418–4421, Dallas, TX, USA.

Kockmann, M., Burget, L., Glembek, O., Ferrer, L., and Cernocký, J. (2010b). Prosodic Speaker Verification using Subspace Multinomial Models with Intersession Compensation. In *INTERSPEECH*, number 102, pages 1061–1064, Makuhari, Japan.

Kockmann, M., Ferrer, L., Burget, L., and Cernocký, J. (2011). iVector Fusion of Prosodic and Cepstral Features for Speaker Verification. In *INTERSPEECH*, number August, pages 265–268, Florence, Italy.

Kohler, M. and Kennedy, M. (2002). Language Identification Using Shifted Delta Cepstra. In *Midwest Symposium on Circuits and Systems*, volume 3, pages III/69–III/72 vol.3, Tulsa, OK, USA.

Kuhn, R., Nguyen, P., and Junqua, J. (1998). Eigenvoices for Speaker Adaptation. In *International Conference on Spoken Language Processing (ICSLP)*, pages 1774–1777, Sydney, Australia.

Kwan, H. and Hirose, K. (1997). Use of Recurrent Network for Unknown Language Rejection in Language Identification System. In *EUROSPEECH*, pages 63–66, Rhodes, Greece.

Kwasny, S. C., Kalman, B. L., Maynard, A. E., and Wu, W. (1993). Real-Time Identification of Language from Raw Speech Waveforms. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*, pages 161–167, Princeton, NJ, USA.

Kwasny, S. C., Kalman, B. L., Wu, W., and Engebreston, A. M. (1992). Identifying Language from Speech: An Example of High-Level, Statistically-Based Feature Extraction. In *Annual Conference of the Cognitive Science Society*, pages 909–914, Bloomington, IN, USA.

Lamel, L. F. and Gauvain, J.-L. (1993). Identifying Non-Linguistic Speech Features. In *EUROSPEECH*, pages 23–31, Berlin, Germany.

Lawson, A., McLaren, M., Lei, Y., Mitra, V., Scheffer, N., Ferrer, L., and Graciarena, M. (2013). Improving Language Identification Robustness to Highly Channel-Degraded Speech through Multiple System Fusion. In *INTERSPEECH*, number August, pages 1507–1510, Lyon, France.

Lei, Y., Ferrer, L., Lawson, A., McLaren, M., and Scheffer, N. (2014). Application of Convolutional Neural Networks to Language Identification in Noisy Conditions. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number Ml, pages 287–292, Joensuu, Finland.

Leonard, R. G. (1980). Language Recognition Test and Evaluation. Technical report, Air Force Rome Air Development Center.

Leonard, R. G. and Doddington, G. R. (1974). Automatic Language Identification. Technical report, Air Force Rome Air Development Center.

Leonard, R. G. and Doddington, G. R. (1975). Automatic Language Identification. Technical report, Air Force Rome Air Development Center.

Li, H. and Ma, B. (2005). A Phonotactic Language Model for Spoken Language Identification. In *Annual Meeting of the Association for Computational Linguistics*, number June, pages 515–522, Ann Arbor, MI, USA.

Li, H., Ma, B., and Lee, C.-H. (2007). A Vector Space Modeling Approach to Spoken Language Identification. *IEEE Transactions on Audio, Speech and Language Processing*, 15(1):271–284.

Li, K.-P. (1994). Automatic Language Identification Using Syllabic Spectral Features. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 297–300, Adelaide, Australia.

Li, K. P. and Edwards, T. J. (1980). Statistical Models for Automatic Language Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 884–887, Denver, CO, USA.

Lin, C.-Y. and Wang, H.-C. (2005). Language Identification Using Pitch Contour Information. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 601–604, Philadelphia, PA, USA.

Lin, C.-Y. and Wang, H.-C. (2006). Language Identification Using Pitch Contour Information in the Ergodic Markov Model. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I/193–I/196, Toulouse, France.

Lindblom, B. (1963). Spectroraphic Study of Vowel Reduction. *The Journal of the Acoustical Society of America*, 35(11):1173–1781.

Litman, D., Hirschberg, J., and Swerts, M. (2000). Predicting Automatic Speech Recognition Performance Using Prosodic Cues. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–225, San Francisco, CA, USA.

Liu, H.-M., Tsao, F.-M., and Kuhl, P. K. (2005). The Effect of Reduced Vowel Working Space on Speech Intelligibility in Mandarin-Speaking Young Adults with Cerebral Palsy. *The Journal of the Acoustical Society of America*, 117(6):3879. Available from: `http://scitation.aip.org/content/asa/journal/jasa/117/6/10.1121/1.1898623`.

Lloyd-Thomas, H., Parris, E. S., and Wright, J. H. (1998). Recurrent Substrings and Data Fusion for Language Recognition. In *International Conference on Spoken Language Processing (ICSLP)*, volume 1, Sydney, Australia.

Lopez-Moreno, I., Gonzalez-Dominguez, J., Plchot, O., Martínez, D., Gonzalez-Rodriguez, J., and Moreno, P. (2014). Automatic Language Identification Using Deep Neural Networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5337–5341, Florence, Italy.

Lopez-Moreno, I., Ramos, D., Gonzalez-Dominguez, J., and Gonzalez-Rodriguez, J. (2011). Von Misses-Fisher Models in the Total Variability Subspace for Language Recognition. *IEEE Signal Processing Letters*, 18(12):705–708.

Ma, B., Li, H., and Tong, R. (2007). Spoken Language Recognition Using Ensemble Classifiers. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):2053–2062.

Makhoul, J. (1975). Linear Prediction: A Tutorial Review. *Proceedings of the IEEE*, 63(4):561–580.

Martin, A. F., Greenberg, C. S., Howard, J. M., Doddington, G. R., and Godfrey, J. J. (2014). NIST Language Recognition Evaluation - Past and Future. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number June, pages 145–151, Joensuu, Finland.

Martínez, D., Burget, L., Ferrer, L., and Scheffer, N. (2012a). iVector-Based Prosodic System for Language Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4861–4864, Kyoto, Japan.

Martínez, D., Burget, L., Stafylakis, T., Lei, Y., Kenny, P., and Lleida, E. (2014a). Unscented Transform for iVector-Based Noisy Speaker Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, number 1, pages 4042–4046, Florence, Italy.

Martínez, D., Green, P., and Christensen, H. (2013a). Dysarthria Intelligibility Assessment in a Factor Analysis Total Variability Space. In *INTERSPEECH*, pages 2133–2137, Lyon, France.

Martínez, D., Lleida, E., and Ortega, A. (2012b). Albayzin 2012 LRE @ ViVoLab UZ. In *Albayzin Language Recognition Evaluation*, Madrid, Spain.

Martínez, D., Lleida, E., Ortega, A., and Miguel, A. (2012c). Score Level versus Audio Level Fusion for Voice Pathology Detection on the Saarbrücken Voice Database. In *Advances in Speech and Language Technologies for Iberian Languages*, volume 328 of *Communications in Computer and Information Science*, pages 110–120. Springer Berlin Heidelberg.

Martínez, D., Lleida, E., Ortega, A., and Miguel, A. (2013b). Prosodic Features and Formant Modeling for an iVector-Based Language Recognition System. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6847–6851, Vancouver, BC, Canada.

Martínez, D., Lleida, E., Ortega, A., Miguel, A., and Villalba, J. (2012d). Voice Pathology Detection on the Saarbrücken Voice Database with Calibration and Fusion of Scores Using MultiFocal Toolkit. In *Advances in Speech and Language Technologies for Iberian Languages*, pages 99–109. Springer Berlin Heidelberg.

Martínez, D., Miguel, A., Ortega, A., and Lleida, E. (2011a). I3A Language Recognition System for Albayzin 2010 LRE. In *INTERSPEECH*, pages 849–852, Florence, Italy.

Martínez, D., Plchot, O., Burget, L., Ondrej, G., and Matejka, P. (2011b). Language Recognition in iVectors Space. In *INTERSPEECH*, pages 861–864, Florence, Italy.

Martínez, D., Ribas, D., Lleida, E., Ortega, A., and Miguel, A. (2013c). Suprasegmental Information Modelling for Autism Disorder Spectrum and Specific Language Impairment Classification. In *INTERSPEECH*, pages 195–199, Lyon, France.

Martínez, D., Villalba, J., Lleida, E., and Ortega, A. (2014b). Unsupervised Accent Modeling for Language Identification. In *Advances in Speech and Language Technologies for Iberian Languages*, volume 8854 of *Lecture Notes in Computer Science*, pages 49–58. Springer International Publishing.

Martinez, D., Villalba, J., Miguel, A., Ortega, A., and Lleida, E. (2010). ViVoLab UZ Language Recognition System for Albayzin 2010 LRE. In *Albayzin Language Recognition Evaluation*, Vigo, Spain.

Martínez, D., Villalba, J., Ortega, A., and Lleida, E. (2011c). I3A Language Recognition System Description for NIST LRE 2011. In *NIST Language Recognition Evaluation Workshop*, Atlanta, GE, USA.

Mary, L. and Yegnanarayana, B. (2008). Prosodic Features for Language Identification. In *International Conference on Signal Processing, Communications and Networking*, pages 57–62, Chennai, India.

Matejka, P., Burget, L., Glembek, O., Schwarz, P., Hubeika, V., Fapso, M., Mikolov, T., Plchot, O., and Cernocký, J. (2008). BUT language recognition system for NIST 2007 evaluations. In *INTERSPEECH*, pages 739–742, Brisbane, Australia.

Matejka, P., Plchot, O., Soufifar, M., Glembek, O., D'Haro, L. F., Veselý, K., Grézl, F., Ma, J., Matsoukas, S., and Dehak, N. (2012). Patrol Team Language Identification System for DARPA RATS P1 Evaluation. In *INTERSPEECH*, pages 50–53, Portland, OR, USA.

Matejka, P., Schwarz, P., Burget, L., and Cernocký, J. (2006). Use of Anti-Models to Further Improve State-Of-The-Art PRLM Language Recognition System. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/197–I/200, Toulouse, France.

Matejka, P., Schwarz, P., Cernocký, J., and Chytil, P. (2005). Phonotactic Language Identification using High Quality Phoneme Recognition. In *INTERSPEECH*, pages 2237–2240, Lisbon, Portugal.

Matejka, P., Zhang, L., Ng, T., Mallidi, S. H., Glembek, O., Ma, J., and Zhang, B. (2014). Neural Network Bottleneck Features for Language Identification. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number June, pages 299–304, Joensuu, Finland.

McCree, A. (2014). Multiclass iscriminative Training of i-Vector Language Recognition. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number June, pages 166–172, Joensuu, Finland.

McLaren, M., Lawson, A., Lei, Y., and Scheffer, N. (2013). Adaptive Gaussian Backend for Robust Language Identification. In *INTERSPEECH*, pages 84–88, Lyon, France.

Mendoza, S., Gillick, L., Ito, Y., Lowe, S., and Newman, M. (1996). Automatic language identification using large vocabulary continuous speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 785–788 vol.2, Atlanta, GA, USA.

Menendez-Pidal, X. (1996). The Nemours Database of Dysarthric Speech. In *International Conference on Spoken Language Processing (ICSLP)*, pages 1962–1965 vol.3, Philadelphia, PA, USA.

Menéndez-Pidal, X., Polikoff, J. B., and Bunnell, H. T. (1997). An HMM-Based Phoneme Recognizer Applied to Assessment of Dysarthric Speech. In *EUROSPEECH*, Rhodes, Greece.

Mengistu, K. T. and Rudzicz, F. (2011). Comparing Humans and Automatic Speech Recognition Systems in Recognizing Dysarthric Speech. In *Canadian Conference on Artificial Intelligence*, St. John's, NL, Canada.

Mengistu, K. T., Rudzicz, F., and Falk, T. H. (2011). Using Acoustic Measures to Predict Automatic Speech Recognition Performance for Dysarthric Speakers. In *Models and Analysis of Vocal Emissions for Biomedical Applications (MAVEBA)*, Florence, Italy.

Middag, C., Bocklet, T., Martens, J.-P., and Nöth, E. (2011). Combining Phonological and Acoustic ASR-Free Features for Pathological Speech Intelligibility Assessment. In *INTERSPEECH*, pages 3005–3008, Florence, Italy.

Middag, C., Martens, J.-P., Van Nuffelen, G., and De Bodt, M. (2009). Automated Intelligibility Assessment of Pathological Speech Using Phonological Features. *EURASIP Journal on Advances in Signal Processing*, 2009.

Miguel, A., Villalba, J., Ortega, A., Lleida, E., and Vaquero, C. (2014). Factor Analysis with Sampling Methods for Text Dependent Speaker Recognition. In *INTERSPEECH*, number September, pages 1342–1346, Singapore.

Mikolov, T., Plchot, O., Glembek, O., Matejka, P., Burget, L., and Cernocký, J. (2010). PCA-based Feature Extraction for Phonotactic Language Recognition. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number July, pages 251–255, Brno, Czech Republic.

Montavon, G. (2009). Deep Learning for Spoken Language Identification. In *NIPS Deep Learning for Speech Recognition and Related Applications*, Whistler, BC, Canada.

Muthusamy, Y., Berkling, K., Arai, T., Cole, R., and Bernard, E. (1993). A Comparison of Approaches to Automatic Language Identification Using Telephone Speech. In *EUROSPEECH*, pages 1307–1310, Berlin, Germany.

Muthusamy, Y. K. (1992). A Review of Research in Automatic Language Identification. Technical report, Oregon Graduate Institute.

Muthusamy, Y. K., Barnard, E., and Cole, R. A. (1994). Reviewing Automatic Language Identification. *IEEE Signal Processing Magazine*, 11(4):33–41.

Muthusamy, Y. K. and Cole, R. A. (1991). A Segment-Based Automatic Language Identification System. In *Neural Information Processing Systems (NIPS)*, pages 241–248, Denver, CO, USA.

Muthusamy, Y. K. and Cole, R. A. (1992). Automatic Segmentation and Identification of Ten Languages Using Telephone Speech. In *International Conference on Spoken Language Processing (ICSLP)*, pages 1007–1010, Banff, AB, Canada.

Muthusamy, Y. K., Cole, R. A., and Gopalakrishnan, M. (1991). A Segment-Based Approach to Automatic Language Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 353–356 vol.1, Toronto, ON, Canada.

Muthusamy, Y. K., Cole, R. A., and Oshika, B. T. (1992). The OGI Multi-Language Telephone Speech Corpus. In *International Conference on Spoken Language Processing (ICSLP)*, pages 895–898, Banff, AB, Canada.

Nakagawa, S., Ueda, Y., and Seino, T. (1992). Speaker Independent, Text-Independent Language Identification by HMM. In *International Conference on Spoken Language Processing (ICSLP)*, pages 1011–1014, Banff, AB, Canada.

Nandi, D., Pati, D., and Rao, K. S. (2013). Language Identification Using Hilbert Envelope and Phase Information of Linear Prediction Residual. In *International Conference Oriental COCOSDA held jointly with Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE)*, Gurgaon, India.

National Institute of Standards and Technology (NIST) (1996). 1996 Language Recognition Evaluation. Available from: `http://www.itl.nist.gov/iad/mig/tests/lre/1996/`.

National Institute of Standards and Technology (NIST) (2003). 2003 Language Recognition Evaluation. Available from: `http://www.itl.nist.gov/iad/mig/tests/lre/2003/`.

National Institute of Standards and Technology (NIST) (2005). 2005 Language Recognition Evaluation. Available from: `http://www.itl.nist.gov/iad/mig/tests/lre/2005/`.

National Institute of Standards and Technology (NIST) (2007). 2007 Language Recognition Evaluation. Available from: `http://www.itl.nist.gov/iad/mig/tests/lre/2007/`.

National Institute of Standards and Technology (NIST) (2009). 2009 Language Recognition Evaluation. Available from: `http://www.itl.nist.gov/iad/mig/tests/lre/2009/`.

National Institute of Standards and Technology (NIST) (2011). 2011 Language Recognition Evaluation. Available from: `http://nist.gov/itl/iad/mig/lre11.cfm`.

Navratil, J. and Zühlke, W. (1997a). Double Bigram-Decoding in Phonotactic Language Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1115–1118 vol.2, Munich, Germany.

Navratil, J. and Zühlke, W. (1997b). Phonetic-Context Mapping in Language Identification. In *EUROSPEECH*, number September, pages 71–74, Rhodes, Greece.

Navrátil, J. and Zühlke, W. (1998). An Efficient Phonotactic-Acoustic System for Language Identificaton. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 781–784 vol.2, Seattle, WA, USA.

Ng, R. W. M., Lee, T., Leung, C.-C., Ma, B., and Li, H. (2009). Analysis and Selection of Prosodic Features for Language Identification. In *International Conference on Asian Language Processing*, number Figure 2, pages 123–128, Singapore.

Ng, R. W. M., Leung, C.-C., Lee, T., Ma, B., and Li, H. (2010). Prosodic Attribute Model for Spoken Language Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5022–5025, Dallas, TX, USA.

Noor, E. and Aronowitz, H. (2006). Efficient Language Identification Using Anchor Models and Support Vector Machines. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, volume 00, San Juan, Puerto Rico.

Onwuegbuzie, A. J., Daniel, L., and Leech, N. L. (2007). Pearson Product-Moment Correlation Coefficient. In *Encyclopedia of Measurement and Statistics*, pages 750–755. SAGE Publications, first edition.

O'Shaughnessy, D. (2008). Springer Handbook of Speech Processing. In *Springer Handbook of Speech Processing*, volume 126, chapter 11. Forman, page 2130. Springer-Verlag Berlin Heidelberg.

Paja, M. S. and Falk, T. H. (2012). Automated Dysarthria Severity Classification for Improved Objective Intelligibility Assessment of Spastic Dysarthric Speech. In *INTERSPEECH*, Portland, OR, USA.

Paul, D. B. and Baker, J. M. (1991). The Design for the Wall Street Journal-based CSR Corpus. In *HLT'91 Workshop on Speech and Natural Language*, Pacific Grove, CA, USA.

Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2(6):559–572.

Penagarikano, M., Varona, A., Diez, M., Rodriguez-Fuentes, L. J., and Bordel, G. (2012). Study of Different Backends in a State-Of-the-Art Language Recognition System. In *INTERSPEECH*, number Ml, pages 2049–2052, Portland, OR, USA.

Plchot, O., Karafiát, M., Brümmer, N., Glembek, O., Matejka, P., de Villiers, E., and Cernocký, J. (2012). Speaker Vectors From Subspace Gaussian Mixture Model as Complementary Features for Language Identification. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number June, pages 330–333, Singapore.

Prince, S. J. D., Elder, J. H., Warrell, J., and Felisberti, F. M. (2008). Tied Factor Analysis for Face Recognition Across Large Pose Differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):970–984.

Raghavendra, P., Rosengren, E., and Hunnicutt, S. (2001). An Investigation of Different Degrees of Dysarthric Speech as Input to Speaker-Adaptive and Speaker-Dependent Recognition Systems. *Augmentative and Alternative Communication*, 17(4):265–275.

Red Temática en Tecnologías del Habla (2010). Albayzin 2010 Language Recognition Evaluation. Available from: `http://fala2010.uvigo.es/`.

Red Temática en Tecnologías del Habla (2012). Albayzin 2012 Language Recognition Evaluation. Available from: `http://iberspeech2012.ii.uam.es/index.php/call-for-evalproposals-2/language-recognition`.

Reynolds, D. (2003). Channel Robust Speaker Verification via Feature Mapping. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages II/53–II/56 vol.2. Ieee.

Reynolds, D. A. and Rose, R. C. (1995). Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83.

Richardson, F. S. and Campbell, W. M. (2008). Language Recognition with Discriminative Keyword Selection. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4145–4148, Las Vegas, NV, USA.

Robertson, S. (1982). *Dysarthria Profile*. Winslow Press, first edition.

Rodríguez-Fuentes, L. J., Penagarikano, M., Varona, A., Díez, M., Bordel, G., Abad, A., Martínez, D., Villalba, J., Ortega, A., and Lleida, E. (2011a). The BLZ Systems for the 2011 NIST Language Recognition Evaluation. In *NIST Language Recognition Evaluation Workshop*, Atlanta, GE, USA.

Rodríguez-Fuentes, L. J., Penagarikano, M., Varona, A., Díez, M., Bordel, G., Abad, A., Martínez, D., Villalba, J., Ortega, A., and Lleida, E. (2012). The BLZ Submission to the NIST 2011 LRE : Data Collection , System Development and Performance. In *INTERSPEECH*, pages 38–41, Portland, OR, USA.

Rodríguez-Fuentes, L. J., Penagarikano, M., Varona, A., Díez, M., Bordel, G., Martínez, D., Villalba, J., Miguel, A., Ortega, A., Lleida, E., Abad, A., Koller, O., Trancoso, I., Lopez-otero, P., Docio-Fernandez, L., Garcia-mateo, C., Saeidi, R., Soufifar, M., Kinnunen, T., Svendsen, T. r., and Fränti, P. (2011b). Multi-site Heterogeneous System Fusions for the Albayzin 2010 Language Recognition Evaluation. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 377–382, Waikoloa, HI, USA.

Rouas, A. J.-l., Farinas, J., Pellegrino, F., and André-Obrecht, R. (2005). Rythmic Unit Extraction and Modelling for Automatic Language Identification. *Speech Communication*, 47(4):436–456.

Rouas, J.-L. (2005). Modeling Long and Short-Term Prosody for Language Identification. In *INTERSPEECH*, pages 2257–2260, Lisbon, Portugal.

Rouas, J.-L. (2007). Automatic Prosodic Variations Modelling for Language and Dialect Discrimination. *IEEE Transactions on Audio, Speech and Language Processing*, 15(6):1904–1911.

Rouas, J.-L., Farinas, J., Pellegrino, F., and André-Obrecht, R. (2003). Modeling Prosody for Language Identification on Read and Spontaneous Speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/753–I/756 vol.1.

Roweis, S. (1997). EM Algorithms for PCA and SPCA. In *Neural Information Processing Systems (NIPS)*, pages 626–632, Denver, CO, USA.

Roweis, S. and Ghahramani, Z. (1999). A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2):305–345.

Rubin, D. B. and Thayer, D. T. (1982). EM algorithms for ML Factor Analysis. *Psychometrika*, 47(1):69–76.

Rudzicz, F. (2007). Comparing speaker-dependent and speaker-adaptive acoustic models for recognizing dysarthric speech. In *Proceedings of the 9th International ACM SIGACCESS conference on Computers and accessibility (ASSETS'07)*, pages 255–256, Tempe, AZ, USA.

Rudzicz, F., Namasivayam, A. K., and Wolff, T. (2011). The TORGO Database of Acoustic and Articulatory Speech from Speakers with Dysarthria. *Language Resources and Evaluation*, 46(4):523–541.

Sai Jayram, A. K. V., Ramasubramanian, V., and Sreenivas, T. V. (2002). Automatic Language Identification Using Acoustic Sub-Word Units. In *INTERSPEECH*, volume 1, pages 81–84, Denver, CO, USA.

Sai Jayram, A. K. V., Ramasubramanian, V., and Sreenivas, T. V. (2003). Language Identification Using Parallel Sub-Word Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/32–I/35 vol.1.

Sangwan, A., Mehrabani, M., and Hansen, J. H. L. (2010). Automatic Language Analysis and Identification Based on Speech Production Knowledge. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5006–5009, Dallas, TX, USA.

Sarkar, A. K., Rath, S. P., and Umesh, S. (2010). Vocal Tract Length Normalization factor based speaker-cluster UBM for speaker verification. In *National Conference On Communications (NCC)*, pages 1–5, Chennai, India.

Savic, M., Acosta, E., and Gupta, S. K. (1991). An Automatic Language Identification System. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 817–820 vol.2, Toronto, ON, Canada.

Schölkopf, B., Bartlett, P., Smola, A., and Williamson, R. (1998a). Shrinking the Tube: A New Support Vector Regression Algorithm. In *Neural Information Processing Systems (NIPS)*, Denver, CO, USA.

Schölkopf, B., Bartlett, P., Smola, A., and Williamson, R. (1998b). Support Vector Regression with Automatic Accuracy Control. In *International Conference on Artificial Neural Networks (ICANN)*, number 1, pages 111–116, Skövde, Sweden.

Schultz, T., Rogina, I., and Waibel, A. (1995). Experiments with LVCSR Based Language Identification. In *Proceedings of the Speech Research Symposium SRS XV*, pages 89–94, Baltimore, MD, USA.

Schwarz, P. (2008). *Phoneme Recognition based on Long Temporal Context*. PhD thesis, Brno University of Technology.

Selkirk, E. (2011). The Syntax-Phonology Interface. In *The Handbook of Phonological Theory*, chapter 14. Wiley-Blackwell, second edition.

Selkirk, E. O. (1978). On Prosodic Structure and Its Relation to Syntactic Structure. *Nordic Prosody II. Trondheim: TAPIR.*

Sharma, H. V. and Hasegawa-Johnson, M. (2010). State-Transition Interpolation and MAP Adaptation for HMM-based Dysarthric Speech Recognition. In *Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, number June, pages 72–79, Los Angeles, CA, USA.

Sharma, H. V. and Hasegawa-Johnson, M. (2013). Acoustic Model Adaptation Using In-Domain Background Models for Dysarthric Speech Recognition. *Computer Speech and Language*, 27(6):1147–1162.

Sharma, H. V., Hasegawa-Johnson, M., Gunderson, J., and Perlman, A. (2009). Universal Access: Preliminary Experiments in Dysarthric Speech Recognition. In *INTERSPEECH*, pages 7–10, Brighton, UK.

Shlens, J. (2014). A Tutorial on Principal Component Analysis. Technical report, Google Research. Available from: `http://arxiv.org/pdf/1404.1100.pdf`.

Sim, K. C. and Li, H. (2008). On Acoustic Diversification Front-End for Spoken Language Identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16(5):1029–1037.

Singer, E., Torres-Carrasquillo, P., Reynolds, D., McCree, A., Richardson, F., Dehak, N., and Sturim, D. (2012). The MITLL NIST LRE 2011 Language Recognition System. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, number June, pages 209–215, Singapore.

Singer, E., Torres-Carrasquillo, P. A., Gleason, T. P., Campbell, W. M., and Reynolds, D. A. (2003). Acoustic, Phoonetic, and Discriminative Approaches to Automatic Language Identification. In *INTERSPEECH*, pages 1345–1348, Geneva, Switzerland.

Sjölander, K. (1997). The Snack Sound Toolkit. Available from: `http://www.speech.kth.se/snack/`.

Smola, A. J. and Schölkopf, B. (2004). A Tutorial on Support Vector Regression. *Statistics and Computing*, 14:199–222.

Sokolova, M. and Lapalme, G. (2009). A Systematic Analysis of Performance Measures for Classification Tasks. *Information Processing and Management*, 45(4):427–437.

Solomonoff, A., Quillen, C., and Campbell, W. M. (2004). Channel Compensation for SVM Speaker Recognition. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, pages 41–44, Toledo, Spain.

Song, Y., Jiang, B., Bao, Y., Wei, S., and Dai, L.-R. (2013). i-Vector Representation Based on Bottleneck Features for Language Identification. *Electronics Letters*, 49(24):1569–1570.

Soufifar, M., Kockmann, M., Burget, L., Plchot, O., Glembek, O., and Svendsen, T. r. (2011). iVector Approach to Phonotactic Language Recognition. In *INTERSPEECH*, number August, pages 2913–2916, Florence, Italy.

Spearman, C. (1904). "General Intelligence," Objectively Determined and Measured. *American Journal of Psychology*, 15:201–293.

Stevens, K. N. (1998). *Acoustic Phonetics*. The MIT Press, first edition.

Stolcke, A., Akbacak, M., Ferrer, L., Kajarekar, S., Richey, C., Scheffer, N., and Shriberg, E. (2010). Improving Language Recognition with Multilingual Phone Recognition and Speaker Adaptation Transforms. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, pages 256–262, Brno, Czech Republic.

Strand, E. A. (2004). Dysarthrias: Management. In *The MIT Encyclopedia of Communication Disorders*, pages 129–132. MIT Press, first edition.

Sugiyama, M. (1991). Automatic Language Recognition Using Acoustic Features. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 813–816 vol.2, Toronto, ON, Canada.

The Center for Speech and Language Processing at Johns Hopkins University (2013). Speaker and Language Recognition Workshop. Available from: `http://www.clsp.jhu.edu/workshops/archive/ws13-summer-workshop/groups/spk-13/`.

Thymé-Gobbel, A. E. and Hutchins, S. E. (1996). On Using Prosodic Cues in Automatic Language Identification. In *International Conference on Spoken Language Processing (ICSLP)*, pages 1768–1771 vol.3, Philadelphia, PA, USA.

Tipping, M. E. and Bishop, C. M. (1999a). Mixtures of Probabilistic Principal Component Analysers. *Neural Computation*, 11(2):443–482.

Tipping, M. E. and Bishop, C. M. (1999b). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622.

Tong, R., Ma, B., Li, H., and Chng, E. S. (2009a). A Target-Oriented Phonotactic Front-End for Spoken Language Recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 17(7):1335–1347.

Tong, R., Ma, B., Li, H., Chng, E. S., and Lee, K.-A. (2009b). Target-Aware Language Models for Spoken Language Recognition. In *INTERSPEECH*, pages 200–203, Brighton, UK.

Torres-Carrasquillo, P. A., Reynolds, D. A., and Deller, Jr., J. R. (2002a). Language Identification using Gaussian Mixture Model Tokenization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/757–I/760, Orlando, FL, USA.

Torres-Carrasquillo, P. A., Singer, E., Kohler, M. A., Greene, R. J., Reynolds, D. A., and Deller, Jr., J. R. (2002b). Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features. In *INTERSPEECH*, volume 2002, pages 89–92, Denver, CO, USA.

van Leeuwen, D. and Brümmer, N. (2006). Channel-Dependent GMM and Multi-Class Logistic Regression Models for Language Recognition. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, volume 00, San Juan, Puerto Rico.

Van Neuffelen, G., Middag, C., De Bodt, M., and Martens, J. P. (2009). Speech Technology-Based Assessment of Phoneme Intelligibility in Dysarthria. *International Journal of Language & Communication Disorders*, 44(5):716–730.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer New York, first edition.

Verdet, F., Mtrouf, D., Bonastre, J.-F., and Hennebert, J. (2009). Factor Analysis and SVM for Language Recognition. In *INTERSPEECH*, pages 164–167, Brighton, UK.

Viikki, O. and Laurila, K. (1998). Cepstral Domain Segmental Feature Vector Normalization for Noise Robust SpeechRecognition. *Speech Communication*, 25(1-3):133–147.

Wegmann, S., McAllaster, D., Orloff, J., and Peskin, B. (1996). Speaker Normalization on Conversational Telephone Speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 339–341 vol. 1, Atlanta, GA, USA.

White, C., Shafran, I., and Gauvain, J.-L. (2006). Discriminative Classifiers for Language Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/213–I/216, Toulouse, France.

Wong, E., Pelecanos, J., Myers, S., and Sridharan, S. (2000). Language Identification Using Efficient Gaussian Mixture Mode Analysis. In *Australian International Conference on Speech Science and Technology*, pages 78–83.

Wong, E. and Sridharan, S. (2002). Utilise Vocal Tract Length Normalization for Robust Automatic Language Identification. In *Australian International Conference on Speech Science & Technology*, pages 409–414, Melbourne, Australia.

Wu, W., Kwasny, S. C., Kalman, B. L., and Engebretson, E. M. (1993). Identifying Language from Raw Speech An Application of Recurrent Neural Networks. In *Midwest Artificial Intelligence and Cognitive Science Conference*, pages 53–57, Chesterton, IN, USA.

Yan, Y. and Barnard, E. (1995a). A Comparison of Neural Net and Linear Classifier as the Pattern Recognizer in Automatic Language Identification. In *International Conference on Neural Networks and Signal Processing*, Nanjing, China.

Yan, Y. and Barnard, E. (1995b). An Approach to Language Identification with Enhanced Language Model. In *EUROSPEECH*, number 1, pages 1351–1354, Madrid, Spain.

Yan, Y., Barnard, E., and Cole, R. A. (1996). Development of an Approach to Automatic Language identification Based on Phone Recognition. *Computer Speech and Language*, 10(1):37–54.

Yin, B., Ambikairajah, E., and Chen, F. (2007). Hierarchical Language Identification Based on Automatic Language Clustering. In *INTERSPEECH*, pages 178–181, Antwerp, Belgium.

Yin, B., Ambikairajah, E., and Chen, F. (2008a). Improvements on Hierarchical Language Identification Based on Automatic Language Clustering. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4241–4244, Las Vegas, NV, USA.

Yin, B., Ambikairajah, E., and Chen, F. (2009). Voiced/Unvoiced Pattern-Based Duration Modeling for Language Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4341–4344, Taipei, Taiwan.

Yin, B., Thiruvaran, T., Ambikairajah, E., and Chen, F. (2008b). Introducing a FM Based Feature to Hierarchical Language Identification. In *INTERSPEECH*, pages 731–734, Brisbane, Australia.

Yorkston, K. M. and Beukelman, D. R. (1980). A Clinician-Judged Technique for Quantifying Dysarthric Speech Based on Single-WordIntelligibility. *Journal of Communication Disorders*, 13(1):15–31.

Zavaliagkos, G., Schwartz, R., and Makhoul, J. (1995). Batch, Incremental and Instantaneous Adaptation Techniques for Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 676–679, Detroit, MI, USA.

Zhang, Q., Liu, G., and Hansen, J. H. L. (2014). Robust Language Recognition Based on Diverse Features. In *ODYSSEY: The Speaker and Language and Language Recognition Workshop*, pages 152–157, Joensuu, Finland.

Zhang, Y., Alder, M., and Togneri, R. (1994). Using Gaussian Mixture Modeling in Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/613–I/616 vol.1, Adelaide, Australia.

Zhu, D., Adda-Decker, M., and Antoine, F. (2005). Different Size Multilingual Phone Inventories and Context-Dependent Acoustic Models for Language Identification. In *INTERSPEECH*, pages 2833–2836, Lisbon, Portugal.

Zissman, M. A. (1993). Automatic Language Identification Using Gaussian Mixture and Hidden Markov Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 399–402 vol.2, Minneapolis, MN, USA.

Zissman, M. A. and Singer, E. (1994). Automatic Language Identification of Telephone Speech Messages Using Phoneme Recognition and N-Gram Modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I/305–I/308 vol.1, Adelaide, Australia.

Zissman, M. A. and Street, W. (1997). Predicting, Diagnosing and Improving Automatic Language Identification Performance. In *EUROSPEECH*, pages 51–54, Rhodes, Greece.