Exploration par l'analyse factorielle des variabilités de la reconnaissance acoustique automatique de la langue

Florian Verdet

Académie d'Aix-Marseille
Université d'Avignon et des Pays de Vaucluse

Universität Fribourg
Mathematisch-Naturwissenschaftliche Fakultät

THESIS

presented under *joint supervision* to the University of Avignon (France)
and to the Faculty of Science of the University of Fribourg (Switzerland)
in consideration for the award of the academic grade of
Doctor scientiarum informaticarum

SPECIALTY: Computer Science

École Doctorale 536 «Sciences et Agrosciences»
Laboratoire Informatique d'Avignon (EA 4128)
Document, Image and Voice Analysis (DIVA) Group
Department of Informatics, University of Fribourg

Exploring Variabilities through Factor Analysis in Automatic Acoustic Language Recognition

by

Florian VERDET

Defended publicly on September 5, 2011 in front of a committee composed of:

M$^{rs.}$	Régine ANDRÉ-OBRECHT	Professor, IRIT, Toulouse	Rapporteur
M.	Pietro LAFACE	Professor, Politecnico di Turino, Turin	Rapporteur
M$^{rs.}$	Lori LAMEL	Directeur de Recherche, LIMSI-CNRS, Paris	Examiner
M.	Rolf INGOLD	Professor, University of Fribourg, Fribourg	Examiner
M.	Jean-François BONASTRE	Professor, LIA, Avignon	Supervisor
M.	Jean HENNEBERT	Lecturer, Université de Fribourg, Fribourg	Supervisor
M.	Driss MATROUF	Maître de Conférence HDR, LIA, Avignon	Supervisor

Laboratoire Informatique d'Avignon

Departement für Informatik
Document, Image and Voice Analysis

Co-tutelle CRUS · Thesis-No. 1723 (Fribourg) · 2011

Accepted by the Faculty of Science of the University of Fribourg (Switzerland) upon the recommendation of:

M$^{rs.}$ Régine ANDRÉ-OBRECHT

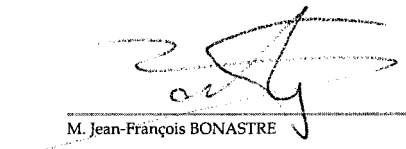
M. Pietro LAFACE

M$^{rs.}$ Lori LAMEL

M. Rolf INGOLD
(thesis co-examiners)

Avignon, 5 September 2011
(date of the oral examination)

Thesis supervisor(s)

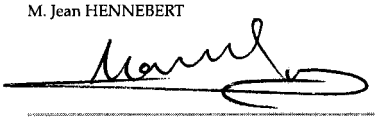


M. Jean-François BONASTRE

Dean

(name of the (Vice-)Dean at the time of the oral examination)

M. Jean HENNEBERT



M. Driss MATROUF

Short Abstract

Language Recognition is the problem of discovering the language of a spoken utterance. This thesis achieves this goal by using short term acoustic information within a GMM-UBM approach. definition

The main problem of many pattern recognition applications is the variability of the observed data. In the context of Language Recognition (LR), this troublesome variability is due to the speaker characteristics, speech evolution, acquisition and transmission channels. problem

In the context of Speaker Recognition, the variability problem is solved by the Joint Factor Analysis (JFA) technique. Here, we introduce this paradigm to Language Recognition. The success of JFA relies on several assumptions: The global JFA assumption is that the observed information can be decomposed into a universal global part, a language-dependent part and the language-independent variability part. The second, more technical assumption consists in the unwanted variability part to be thought to live in a low-dimensional, globally defined subspace. In this work, we analyze how JFA behaves in the context of a GMM-UBM LR framework. We also introduce and analyze its combination with Support Vector Machines (SVMs). solution

The first JFA publications put all unwanted information (hence the variability) into one and the same component, which is thought to follow a Gaussian distribution. This handles diverse kinds of variability in a unique manner. But in practice, we observe that this hypothesis is not always verified. We have for example the case, where the data can be divided into two clearly separate subsets, namely data from telephony and from broadcast sources. In this case, our detailed investigations show that there is some benefit of handling the two kinds of data with two separate systems and then to elect the output score of the system, which corresponds to the source of the testing utterance. improvement

For selecting the score of one or the other system, we need a channel source detector. We propose here different novel designs for such automatic detectors. In this framework, we show that JFA's variability factors (of the subspace) can be used with success for detecting the source. This opens the interesting perspective of partitioning the data into automatically determined channel source categories, avoiding the need of source-labeled training data, which is not always available. related analyses

The JFA approach results in up to 72% relative cost reduction, compared to the GMM-UBM baseline system. Using source specific systems followed by a score selector, we achieve 81% relative improvement. overall results

Résumé Français

définition

La problématique traitée par la Reconnaissance de la Langue (LR) porte sur la découverte de la langue contenue dans un segment de parole. Cette thèse se base sur des paramètres acoustiques de courte durée, utilisés dans une approche d'adaptation de mélanges de Gaussiennes (GMM-UBM).

problème

Le problème majeur de nombreuses applications du vaste domaine de la reconnaissance de formes consiste en la variabilité des données observées. Dans le contexte de la Reconnaissance de la Langue (LR), cette variabilité nuisible est due à des causes diverses, notamment les caractéristiques du locuteur, l'évolution de la parole et de la voix, ainsi que les canaux d'acquisition et de transmission.

solution

Dans le contexte de la reconnaissance du locuteur, l'impact de la variabilité peut sensiblement être réduit par la technique d'Analyse Factorielle (Joint Factor Analysis, JFA). Dans ce travail, nous introduisons ce paradigme à la Reconnaissance de la Langue. Le succès de la JFA repose sur plusieurs hypothèses. La première est que l'information observée est décomposable en une partie universelle, une partie dépendante de la langue et une partie de variabilité, qui elle est indépendante de la langue. La deuxième hypothèse, plus technique, est que la variabilité nuisible se situe dans un sous-espace de faible dimension, qui est défini de manière globale. Dans ce travail, nous analysons le comportement de la JFA dans le contexte d'un dispositif de LR du type GMM-UBM. Nous introduisons et analysons également sa combinaison avec des Machines à Vecteurs Support (SVM).

amélioration

Les premières publications sur la JFA regroupaient toute information qui est nuisible à la tâche (donc ladite variabilité) dans un seul composant. Celui-ci est supposé suivre une distribution Gaussienne. Cette approche permet de traiter les différentes sortes de variabilités d'une manière unique. En pratique, nous observons que cette hypothèse n'est pas toujours vérifiée. Nous avons, par exemple, le cas où les données peuvent être groupées de manière logique en deux sous-parties clairement distinctes, notamment en données de sources téléphoniques et d'émissions radio. Dans ce cas-ci, nos recherches détaillées montrent un certain avantage à traiter les deux types de données par deux systèmes spécifiques et d'élire comme score de sortie celui du système qui correspond à la catégorie source du segment testé.

analyses engendrées

Afin de sélectionner le score de l'un des systèmes, nous avons besoin d'un détecteur de canal source. Nous proposons ici différents nouveaux designs pour de tels détecteurs automatiques. Dans ce cadre, nous montrons que les facteurs de variabilité (du sous-espace) de la JFA peuvent être utilisés avec succès pour la détection de la source. Ceci ouvre la perspective intéressante de subdiviser les

données en catégories de canal source qui sont établies de manière automatique. En plus de pouvoir s'adapter à des nouvelles conditions de source, cette propriété permettrait de pouvoir travailler avec des données d'entraînement qui ne sont pas accompagnées d'étiquettes sur le canal de source.

résultats généraux

L'approche JFA permet une réduction de la mesure de coûts allant jusqu'à 72% relatives, comparé au système GMM-UBM de base. En utilisant des systèmes spécifiques à la source, suivis d'un sélecteur de scores, nous obtenons une amélioration relative de 81%.

Zusammenfassung Deutsch

Definition

Automatische Sprachen-Erkennung (Language Recognition, LR) besteht darin, die Sprache einer gesprochenen Sequenz herauszufinden. Die vorliegende Dissertation erreicht dieses Ziel, indem kurzzeit-akustische Informationen in einem GMM-UBM-basierten Ansatz analysiert werden.

Problem

Das Hauptproblem zahlreicher Anwendungen der Mustererkennung ist die Veränderlichkeit der beobachteten Daten. Im Umfeld der Sprachenerkennung (LR), stammt diese störende Variabilität aus den individuellen Merkmalen des Sprechers, aus Schwankungen und Veränderungen des Gesprächs und der Stimme, sowie aus dem Erfassungs- und dem Übermittlungskanal.

Lösung

Dieses Variabilitätsproblem wird im Kontext der Sprecher-Erkennung mit der Faktoranalysis-Technik (Joint Factor Analysis, JFA) gelöst. In dieser Arbeit wird dieses Paradigma in die Sprachenerkennung eingeführt. Der Erfolg der JFA basiert auf mehreren Annahmen: Die Globalanschauung der JFA besteht darin, dass die beobachteten Informationen aufgetrennt werden können in einen universellen globalen Teil, einen sprachabhängigen Teil und einen sprachunabhängigen Variabilitätsteil. Die zweite, technischere Annahme besagt, dass sich dieser störende Variabilitätsteil auf einen global definierten Unterraum von geringer Dimension beschränkt. In der vorliegenden Arbeit wird analysiert, wie sich die JFA im Rahmen eines GMM-UBM-basierten LR-Systems verhält. Auch dessen Kombination mit Support Vector Machine (SVM) wird hier eingeführt und analysiert.

Verfeinerung

Die ersten Veröffentlichungen zu JFA fassen alle störenden Informationsteile (die Variabilitäten) in einer einzelnen Komponente zusammen. Es wird davon ausgegangen, dass diese einer Gauss-Verteilung folgt. Dies bewältigt verschiedene Arten von Variabilität in einer einheitlichen Weise. In der Praxis wird hier jedoch beobachtet, dass diese Hypothese nicht immer bestätigt werden kann. Die Daten der NIST LRE 2009 Kampagne können z.B. in zwei klar trennbare Untermengen aufgeteilt werden, nämlich Daten aus telefonischen und aus Rundfunk-Quellen. In diesem Fall zeigen die hier durchgeführten ausführlichen Untersuchungen, dass ein gewisser Vorteil besteht, diese zwei Datentypen durch zwei separate, spezialisierte Systeme zu handhaben und als Ergebniswert denjenigen zu wählen, dessen System der Quelle der Testsequenz entspricht.

weiterführende Analysen

Um den Ergebniswert des einen oder anderen Systems zu wählen, wird ein Modul benötigt, welches die Quelle erkennt. Hier werden verschiedene neuartige Designs solcher automatischer Detektoren vorgeschlagen. In diesem Rahmen wird auch gezeigt, dass die Variabilitätsfaktoren (des JFA-Unterraums) erfolgreich dazu verwendet werden können, den Quellenkanal zu erkennen. Dies eröffnet die

interessante Perspektive, die Daten in automatisch bestimmte Quellkategorien zu unterteilen. Dies würde vermeiden, eine quellenbezeichnende Annotation für die Trainingsdaten zu benötigen, welche nicht immer zur Verfügung steht.

globale Ergebnisse

Der JFA-Ansatz erlaubt eine relative Kostenreduktion von bis zu 72% gegenüber dem GMM-UBM-basierten Grundsystem. Beim Einsatz von quellspezifischen Systemen und einem Ergebniswert-Selektor wird eine relative Verbesserung von gar 81% erreicht.

Abstract

Language Recognition (LR) is the problem of discovering the language of a spoken utterance. In this work, we focus on a short term acoustic modeling approach, leaving apart solutions which use phonetics, phonotactics or prosody. Our baseline approach builds on Gaussian Mixture Models (GMMs), whose mean values are adapted from the Universal Background Model (UBM) through a Maximum A Posteriori (MAP) criterion. The systems are evaluated on the NIST LRE 2005 and 2009 core tasks, including utterances of nominally 30 seconds of 7 and 23 languages, respectively.

One of the big problems of researches in the general field of pattern matching is the variability of the observed data. In the narrower domain of Language Recognition (LR), this variability is due to factors of different nature: the differences between individual speakers of a language, the evolution of the speech of each speaker, differences or fluctuations in the signal due to the acquisition and transmission channel, and possibly dialects or accents (in the case we do not want to distinguish between them). A good LR system should come along with considerable robustness against variations of these factors.

This thesis introduces a series of novelties. As guiding axis, we analyze the effects of a technique called Joint Factor Analysis (JFA) in order to handle data variability and to add considerable robustness to the system. The JFA approach has been reported to work well in the context of Speaker Recognition. Here, we adapt and introduce it to the LR problem. When adding the JFA step to the baseline (GMM-UBM) system, we observe a relative reduction of the detection cost of up to 72%.

Another novelty consists in combining the JFA approach with Support Vector Machines (SVMs). So we investigate how JFA behaves in the context of an SVM, for which different topologies are trialed. The addition of SVMs to the GMM-UBM approach is already able to reduce the error rates or the costs considerably. But applied to the baseline GMM-UBM, JFA has an even bigger gain. The gain of applying SVMs on JFA-compensated models is not as high as the JFA gain is for the GMM-UBM approach. The resulting absolute average cost (with JFA) may even be slightly higher.

The first JFA publications put all unwanted information (hence the variability) into one and the same component, which is thought to follow a Gaussian distribution. This handles diverse kinds of variability in a unique manner and produces a considerable enhancement over systems without JFA. But in practice, we observe that this hypothesis is not always well verified. In NIST LRE 2009, we have for

example the case, where the data can be divided into two clearly separate subsets, namely data from telephony, Conversational Telephone Speech (CTS), and from broadcast Voice of America (VOA) sources. In analyses, we show that the data of these two channel sources have rather big differences.

In an in-depth investigation of unseen magnitude, we look at different ways to cope with this problem. We try to handle these two channel categories in a separate, parallel way up to a certain point and then to merge together these two parts to produce one score or decision in output.

The best strategy of this case is to build two completely parallel systems and to choose the one or the other score for output. In the context of this investigation, this is first done by an oracle indicating of which channel category the actual utterance is. This approach yields an additional cost reduction of 18.6% relative, compared to the global JFA approach. In total, this is an enhancement of 81% relative over the baseline system.

category selector

The analyses selecting the score of the correct system lead to researches on finding automatic ways to detect the category of an utterance. We propose different novel ways to achieve this. They operate directly and solely on the variability factors of the JFA approach. While the oracle detection constitutes ground-truth, our automatic detectors have a channel category identification rate of 87%. When replacing the oracle by the automatic module, the system performance degrades only by 6%, relative to the oracle one.

The detectors which are based on the variability factors have a big advantage. Using these factors opens the interesting perspective of partitioning the data into automatically determined channel source categories. A dedicated system can then be built for each of these classes and this automatic detector finally be used for merging them. They may even be employed in an environment where the training data is not accompanied by labels about the channel source.

Contents

Problem and Challenge

language recognition

The works presented in this document concern the large domain of automatic speech processing. It is articulated around Language Recognition, whose task is to recognize the language of some spoken utterance. The general approach is narrowed down to a purely acoustic based technique, allowing to run it in an automatic way without the need of laborious data preparation steps like authoring phonetic transcriptions.

structure of speech

Speech is a living entity. It comes under very different natures, such as read or conversational speech. Speech may also be characterized depending on the person who speaks, with the obvious example being the natural differences between women and men speaking. Another kind of variation may also be due to more technical aspects like the environment or the type of microphone used for the recording.

language models

Automatic Language Recognition relies on automatically trained language models, which have to capture thoroughly the structure of spoken languages. They have also to be able to highlight the particularities of single languages or the differences between languages in order to discern them.

nuisance

All the information contained in the captured speech signal and that does not belong to the language itself can be considered as nuisance from the point of view of the Language Recognition task. This typically corresponds to the troublesome variabilities sketched above and which are due to the manifold sources as the type of speech, the speaker particularities, as well as the recording and transmission channel.

variability compensation

This thesis focuses on handling these variabilities in order to enhance the ability to recognize the language. The proposed approach particularly looks at the variabilities and introduces ways to compensate the systems for this nuisance.

different approaches

The variability problem can be looked at either in a holistic way considering everything that does not depend on the language itself as being nuisance in an abstract manner. On the other hand, the different variabilities may be approached separately and enumerated individually. This implies to identify the diverse natures of variability: speaker particularities, modifications in the recording setup or even distinguishing general transmission sources like the telephone or broadcasts.

main challenge

The heart of this thesis builds around an approach, which directly takes into account this nuisance. It consists in explicitly modeling the variability at the same time as the classical language-dependent information, but in a separate term. This thesis will analyze on one hand the holistic treatment by applying this Joint Factor Analysis (JFA) approach, which proved well for speaker verification. It will as well investigate ways to handle transmission source categories in a separate manner.

The main challenge of this thesis is the nuisance that hits Language Recognition applications in a considerable way. The effects of these variabilities and their compensation are investigated under the protocols defined by National Institute of Standards and Technology's biennial Language Recognition Evaluations.

Part I

Introduction

Für alle, die die Schönheit von Wissenschaft anderen zeigen wollen.

— Dedication of the BEAMER class user guide [0]

Chapter 1

Introduction to the Research Field

Contents

Language Recognition is the problem of discovering the language of a spoken utterance. It is achieved by automatic approaches founded typically on signal processing and stochastic modeling. They are applied on different parts of information carried in the signal (acoustic, phonotactic, semantic, etc). The whole is realized by a sequence of processing steps, which are subjected to various kinds of factors.

— ◇ —

[0] The (LaTeX) BEAMER class, User Guide for Version 3.10. Till Tantau, Joseph Wright, Vedran Miletić; July 12, 2010.
English translation: *To all, who want to show the beauty of science to others.*

1.1 Scientific goals

This section sketches the problems faced by Language Recognition (LR) systems. It further presents some basic requirements of an automatic speech processing system, together with the difficulties to meet them. The broad goals of the work presented in this document are located with respect to these basic requirements. Subsequent sections will then give a more in depth presentation of the domain of Language Recognition as well as a more ample introduction to the problem we try to solve.

— ◇ —

goal of research

A general goal of a research project may consist in improving performances of a system in a given (and controlled) evaluation context. For Language Recognition systems, this would mean to defy the performance of the best system at any price. But before all, it is surely of bigger value to the scientific community if research targets in producing advances and novelties that are persistent over the change of setups.

easy extension

In order to be perennial, an approach should evolve towards *universal applicability*. This means that results of research efforts should still be applicable if several factors of the evaluation or the application context get changed. This permits for instance to be able to add new languages to the recognition system with a minimal (or at a lowered) effort.

The universality of the chosen approaches is one of the basic principles adopted for the research efforts presented in this document. This means that the same process may be applied on any set of target languages, independent of the number or the identity of these languages. It should be possible to extend this set to include additional languages with a very small effort in human and computational labor. One way to achieve this goal is by choosing a purely acoustic approach. In the concrete case, the only thing required for adding a new language should be a decent amount of recorded speech in that language.

robustness

Besides the independence to the set of languages that have to be recognized and the ease to add new languages, we think an LR system should also be robust. This research goal includes *robustness* against modifications in the environment (for instance changing from indoor to outdoor recording). It can also be the robustness against changing signal conditions such as when data is coming from diverse origins (e.g. telephonic vs. broadcasted data). Particularly targeted is also robustness to the natural "fluctuation" of the observations for a given class (for a language, in our case). This fluctuation is the *variability* that occurs between the single observations or *utterances* to be more specific.

variability

This thesis focuses on the general problem of *variability*. By variability, we mean the differences we can observe between samples belonging to the same class. Variability is one of the main factor hindering Language Recognition performances. This is also true for most pattern recognition problems. For instance in image processing, this may include distortions like rotation or illumination, as well as

variabilities inherent to the classes' subjects like wearing glasses or hair style in a face recognition context.

In the automatic speech processing domain, fluctuation problems can be considered to be conceptually similar to those in image processing, but of a different nature since we are working on a temporal signal. One fundamental difference is that the speech signal is said to be non-stationary in the long term, reflecting the fact that an utterance is the result of the production of a sequence of phones. These have rather different short-term characteristics. The aspect of an utterance (thus the sequence of phones) is mainly dependent on the speech *content* (what is being said).

Even if the observed characteristics in speech processing are of *acoustic* nature, problems about variability have to be faced. In the LR context, different kinds of variability may be distinguished. They will be highlighted in the following paragraphs.

speaker

In the particular domain of Language Recognition the languages can be distinguished by their acoustic properties. Apart from this, acoustic differences mainly occur between distinct *speakers*. This has a big impact in modeling the spoken language concisely. Each speaker has a specific (physical) configuration of its vocal tract, as well as behavioral differences in the production of speech. These distinctions are usually called *inter-speaker variability*. They are useful and typically tracked by Speaker Recognition (SR) applications, but are disturbing in the context of a Language Recognition system. As (Abe et al., 1990; Li, 1994) also point out, the variability between speakers (of a same language) is of roughly the same magnitude as the difference between the languages themselves.

accent

On an other aspect, since languages are living objects undergoing slow, but perpetual evolution, a class representing a particular language can not be defined with precision. This fact is largely endorsed by different *accents* or even *dialects* of one and the same language. Hence for practical usage of automatic LR, the choice of pooling together different dialects into the same class or to handle them separately is dictated by the application context. This inherent aspect of dialects, accents or simply the language's natural evolution also accounts for differences between utterances of the same class.

channel

Besides differences between individual speakers, differences in the speech signal recording and transmission steps can also be observed. These steps are referred to as *channel*. Their variability is troublesome in most applications.

intra-speaker

A further variability that may be observed lies in the fact that a same speaker will produce slightly different signal characteristics depending to her/his state (being of health, emotional, environmental, etc nature). The voice of a given speaker is also evolving over time. In Speaker Recognition, this is called the *intra-speaker variability*.

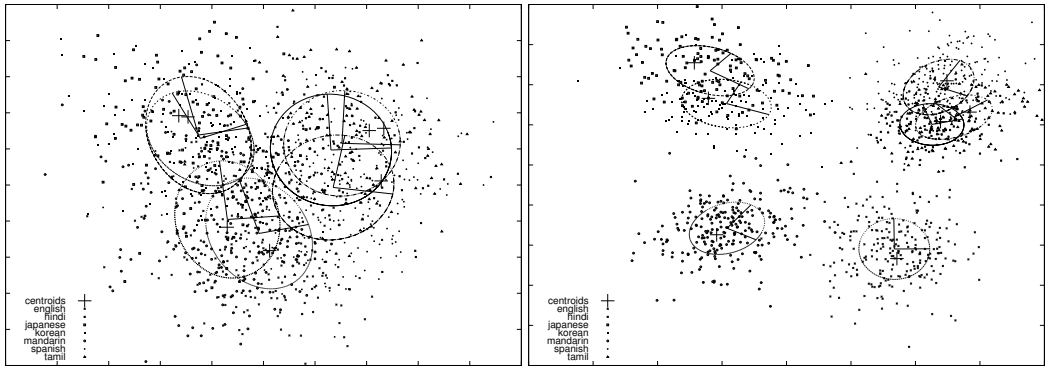
acoustic variability

All the mentioned differences will be referred to as *variabilities*. The works presented and proposed in this document are aimed at solving this problem involving *variabilities* of diverse sources.

As hinted above, Speaker Recognition (SR) has to handle session and channel variability. One of the major questions addressed in the presented works is if the techniques used in SR may also be applied to the Language Recognition problem. A further important question is if these techniques are able to handle the aforementioned differences between individual speakers in the same run as the channel effects. The channel effect is probably of a more fine-grained nature than the inter-speaker variability. Is this even the case if the variabilities to track down are substantially different between a Speaker Recognition and a Language Recognition application? In Language Recognition, they are more diverse and span bigger differences[1].

class separability

To sum it up, the basic problem we have to face is the variability of the data. All this variability can be seen as being the spreading of the points (coordinate-vectors representing individual utterances) of one class with regard to the spacing between the different classes. Fig. 1.1 illustrates this in (a) where the point distributions of different classes largely overlap — the classes are difficult to be separated. The ellipses depict the scattering of each class' points (being the covariance of a Gaussian fitting). What we target to obtain is a *compensation* or a *reduction* of the variability. This means that the points of each class will be more concentrated and the different classes farther apart. This is depicted in (b). As result, the classes will be more separable since the point distributions of different classes will overlap less.



(a) Problem: classes largely overlap (b) Target: reduced variability = separability

Figure 1.1: *Distribution of language-centroids and individual utterances, projected onto first two PCA dimensions, standard deviation ellipses for each language's points*

As a further addition of variability, we may see differences between recording databases (corpora) in their design and realization. This kind of variability is similar to the channel variability, but on a far larger scale than the fine differences of the channel between individual utterances.

To give an idea, in the presently analyzed case, this involves corpora of conver-

[1] Assuming that the speaker variability is more important than the variability about the channel. LR includes the problem of speaker variability whereas on the other side, SR usually does not contain the problem of language variability (a speaker speaking in several languages, at least without accent).

sational telephone speech opposed to recordings of radio broadcast.

database differences

Handling corpora of noticeably different channel categories constitutes the second major axis of the works presented herein. We think that the huge differences, as they may occur between databases, require more ingenious approaches. In a first step to resolve such a possible mismatch of channel category, we assume that these categories (corpora sources) may be handled separately up to a certain level. Then the results will have to be merged.

A next step to this parallel handling may then be some way to detect the category by automatic means. This will allow to partition the data into two or more subsets (i.e. for the case the category is not known a priori). These subsets can then be handled separately following the parallel approach. Since there is a multitude of different ways to design such channel category detectors, some interesting approaches will be investigated in this work.

— ◇ —

summary

All presented works are based on the principle of keeping a Language Recognition system easily extensible to additional languages while improving its robustness.

The main problem we observe in the data is its variability with respect to the different classes, which are the languages. This variability is due to very different sources, including differences between individual speakers, recording setup, signal transmission, or even accent or dialects.

The approaches investigated in this work follow the goal of taking into account the variability. We then compensate this variability in order to enhance class separability. This also yields systems that are more robust to fluctuations in the different kinds of variabilities.

1.2 General introduction

In order to be able to address the burning problems and the goals stated in previous section, we particularly introduce the working domain by a top-down approach. It also locates the application field among similar and linked domains. This will be followed by a description of the challenge Language Recognition implies and some areas of application.

1.2.1 Situation of the domain

pattern recognition

Automatic speech processing is an important domain of artificial intelligence research, or more technically speaking, machine learning or pattern recognition. Its visions comprise computer systems able to dialogue with humans. There is a lot of research going on in this domain and such systems could evolve to next generation's most used human-machine interfaces. They are usually quite well

accepted because of their non-intrusive nature and their simple and natural way of usage.

automatic speech processing

Automatic speech processing covers several fields. One of the best known is automatic speech recognition, whose task is to transform an acoustic speech signal into its textual/symbolic representation which is then interpreted for some action to be taken by the computer. This ranges from keyword recognition (e.g. on cellular phones) over dictation systems (for text processing) to fully featured dialog systems (hotel reservations or similar).

Another area with a lot of research is speaker verification or speaker identification. The verification problem checks if the observed utterance has really been spoken by the speaker the actual person claims to be. Thus this usually fulfills an authentication purpose.

automatic language recognition

A further field of automatic speech processing is the task dealing with automatic Language Recognition (LR), which is the process of recognizing the language used in a sample of speech. LR systems can be evaluated in an identification task, electing the one correct language out of a (closed) set of L languages, which is referred to as Automatic Language Identification (ALI or LID).

But usually the task is designed in a slightly different manner, namely in verification mode, detecting if a candidate language is used in the input waveform or not. So for an utterance of unknown language, the system is confronted with a series of binary (*yes/no*) questions — one for each potential language. LR is often run in a closed set context, meaning that the utterance is forcibly spoken in one of the proposed languages. But language detection easily allows open set experiments, since the system may have responded with "*no*" for all languages. What changes is principally the method of system performance measurement. For the work presented here, we will mainly focus on closed-set language detection.

prior work

While there are only a few publications dating from before 1980 (Leonard and Doddington, 1974; House and Neuburg, 1977; Cimarusti and Ives, 1982), Automatic Language Recognition grew to an important field of research only after 1990 (see (Geoffrois, 2004) for a very concise overview). It really started running in the very early 1990s with several feasibility studies and first dedicated works with real data (Savic et al., 1991; Riek et al., 1991; Muthusamy and Cole, 1992; Hazen and Zue, 1993; Zissman, 1993; Lamel and Gauvain, 1994; Zissman and Singer, 1994). During the same period, the first large Language Recognition centered corpora appeared: OGI[2]-MLTS[3], (Muthusamy et al., 1992) and three years later the OGI 22-Languages (Lander et al., 1995) corpus.

Significant progress has been made over the last decades through advanced modeling that can be applied at the different levels of information. For example on

[2]Oregon Graduate Institute, School of Science and Engineering, a department of Oregon Health & Science University (OHSU), subsequently named Department of Science & Engineering, and now Center for Spoken Language Understanding, of the OHSU (CSLU) (OGI).

[3]Multi-Language Telephone Speech corpus, sometimes just called TS (MLTS).

the acoustic level (Zissman, 1993; Torres-Carrasquillo et al., 2002b; Singer et al., 2003; Campbell et al., 2004; Verdet et al., 2009b) and on the phonotactic level (Zissman and Singer, 1994; Yan et al., 1996; Zissman, 1996; Hazen and Zue, 1997; Torres-Carrasquillo et al., 2002b; Singer et al., 2003; BenZeghiba et al., 2008). A large deal of the progress has been stimulated by a growing availability of large benchmarking data sets (Muthusamy et al., 1992; Lander et al., 1995). A big part of progress has also been induced by systematic comparisons of systems through the organization of evaluation campaigns such as the NIST's Language Recognition Evaluations (LREs) in the years 1996, 2003, 2005, 2007 and 2009 (NIST LRE, 2009).

1.2.2 Challenges

Automatic Language Recognition consists in processing a speech signal to detect which language the speaker is talking in. On a finer level, it could distinguish different dialects or even regional accents or accents due to languages spoken by non native speakers.

The main challenges in automatic LR are the following (the French article (Geoffrois, 2004) gives a good sketch of these points):

- Certain languages are more proximate to other ones by nature, as for instance languages of a same family or language group (e.g. the Latin languages).
- The natural language is a living object which undergoes a perpetual development and evolution. It also has many facets depending on the region, environment, speech topic, partner and communication medium.
- With increasing mobility and globalization, many people are urged to speak in foreign languages (and are thus called non-native speakers). Without a perfect integration and assimilation of this non-native language, there are always some elements of the native language that interfere and which can cause problems to the Language Recognition system. accents
- Even native speakers can have intense accents or dialects. Sometimes, dialects are so special that most Language Recognition systems will not work well enough without special training for that dialect (or even introducing this dialect as a sort of new language)[4]! dialects
- Another problem consists in foreign words imported into a language. If they were not fully adapted to the actual language, their sound (and even phonotactics) does not match the common sound (or construction) of the language. Similarly, the pronunciation of proper names may also make phonemes of other languages appear in the currently spoken language. This phenomenon is especially present in multilingual countries. foreign words

[4]For supplementary information on accents, consider reading at least the introduction of (Vaissière and Boula de Mareüil, 2004).

From what is stated above, boundaries between languages are probably not clear-cut. They are manifold and easily flow one into another. Because of that, a lot of studies on Language Recognition are done with clearly distinctive categories (exemplary languages) — omitting the fuzzy region in between them.

To complicate the LR task even more, we could have to process multilingual speech signals, which requires them to be segmented into monolingual parts and to identify the language of each of these segments. Such tasks are called *language segmentation* and may for instance have to work on online streams (e.g. a radio broadcast). They are out of the direct scope of this work.

1.2.3 Application areas

multilingualism

Automatic Language Recognition is applied where speech has to be handled in a multilingual environment. Nowadays, multilingual capacities are required in many domains due to worldwide collaboration and services (enhanced by the internet) or in environments covering several languages. This includes also tasks where an automatic LR system assists humans to deliver a service in the requested language, but also systems where speech is processed in a fully automatic way.

— ◇ —

security applications

One of the first real-world applications which showed interest to automatic LR originated from nationwide surveillance applications. Such services try to spot telephone calls of chosen languages. This implies a non-interactive LR system which may also work offline on previously recorded data. This kind of application still continues to be prompting nowadays and it is one of the driving entities, notably through regular evaluation campaigns endorsed by this environment (i.e. NIST LRE).

telephone

Another rather popular example is the one of a telephone server taking incoming calls and detecting the language a client talks in. The call is then redirected to some suitably skilled operator. Such servers could be found in multinational companies or in government departments (like in the European Union since most probably no operator speaks every of the 23 languages[5]). Its application could also reach other telephone or computer based services like travel or tourism services, translation or information services or even support in emergency situations (Lamel and Gauvain, 1994).

It can also be used as the first step of an automatic multilingual speech recognition system. After the language has been detected, the speech recognition unit for the corresponding language can be selected. This first step of a telephone voice-activated service is nowadays performed asking the user to press a given key on the telephone or to pronounce the preferred language, which is cumbersome. Examples for such systems would be fully computer based telephone servers for

[5] See http://en.wikipedia.org/wiki/Languages_of_the_European_Union

commercial services like voice-commerce, banking or stock trading (Hieronymous and Kadambe, 1996).

Automatic Language Identification systems will probably also find a large deal of applications in a passive (non-interactive) manner for processing (indexing, tagging,...) large data sets like uploaded videos, broadcast or TV recordings, meeting minutes or for national security applications. tagging

1.3 Approaches to language recognition

The previous sections laid out the domain of Language Recognition (LR) in a general manner and from an applicative point of view. On the more technical side, we now will introduce some basic requirements and characteristics of speech processing systems. They are also valid in the context of LR systems. Subsequently, the different types of automatic approaches are listed and compared to some insights of how recognition is likely to be carried out by humans.

1.3.1 System characteristics

A very early publication about Language Recognition, (Cimarusti and Ives, 1982), gives a good sketch of what an LR system should bring along. The enumerated points are reproduced in this section with updated and extended descriptions:

Content independence — A LR system should not depend on the topic covered by the speech utterance. It should be able to work on speech covering any domain. So it should not be restricted to applications for instance for one specific profession. Most of the times, this condition is verified, even if sometimes it may be linked to the form of the speech (read, prepared or conversational). In comparison to that, we find even nowadays a vast majority of (text based) translation systems not holding this point since they keep to be specific to one application domain (as the medical or the juristic one).

Context independence — "*Context-independence indicates that the speech signal surrounding the extracted portion may be from among an almost infinite number of possible contexts.*" (Cimarusti and Ives, 1982). For standard applications, where a recorded segment containing only one language is presented to the system, this condition is met by design. In other applications, where for instance consecutive segments of different languages have to be classified, we possibly dispose of the information that any two adjacent segments are not of the same language. This could make the detection slightly easier since the priors change. For automatic speech recognition, this condition is met on a short-time level by explicitly modeling context-dependent phones/phonemes.

Acquisition and transmission independence also named *Form independence* by (Cimarusti and Ives, 1982) — An LR system should be able to work with

speech of different form or coming from different sources. This may include recorded speech, speech transmitted over a telephone line, speech acquired by different means (e.g. handhelds) and in different environments or even live acquisitions or streams. This is one of the hardest condition to meet. Generally, as long as the testing data is of similar nature as the data previously seen, there is not much trouble. This is one point which will be addressed in this work (Chapter 6).

Language independence — The system design should not rely on the availability of data or knowledge that exists only for certain languages. It should be able to work with any language. It would be discriminatory if the system was able to work only on a dozen of worlds many thousand languages.

Speaker independence — The LR system should not be dependent on the speaker producing the utterance. Nor should it depend on speaker characteristics like sex, age, emotional state or health. This is also a condition, which is not trivial to verify. It constitutes one of the driving factors for the investigations addressed in this work (see also Sect. 1.4).

Style independence — An LR system should be able to work on speech that comes in different styles, such as read speech, prepared speech, formal speech and conversational or casual speaking. An even bigger impact to the speech style has for instance the high stress of people in an urgency situation (such as 911-calls in the U.S.). Early systems worked on read speech, while the NIST LRE campaigns work with conversational (telephone) speech.

Degraded speech signal — An LR system should also come along with a certain robustness against degraded conditions such as noise. Usually, this point heavily depends on the severity of the degradation. This may range from only a slight background noise to very adverse conditions, where the speech is barely understandable even for humans.

Total automation — The ultimate goal of a Language Recognition system is of course its ability to run without (or with the least possible) human interaction.

While some of the above conditions are easier to be met, other are more delicate and require dedicated research. Our goal is to develop a system that is above all widely reusable and as portable as possible. It should be task and language independent in order to be able to add new languages solely by presenting a set of corresponding utterances to the system.

Specifically for the presented works, the conditions that are easiest met are the content and the context independence. Also met by design (using acoustic modeling) are the independence of the language and the total automation. The driving factors for the analyses following in this document are clearly the speaker, as well as the acquisition and transmission independence. Even if our works do not specifically cope with style independence, it should in many cases not have

a big impact. We are principally working with conversational speech[6], which is, compared to prepared speech, a moderately hard task. One of the biggest problems on the style side is speech of urgency situations (911-calls for instance). Such data is very challenging, even if the signal itself may be of good quality.

1.3.2 Basic human language recognition

The human's capacity in recognizing the language spoken depends a lot of the set of languages he knows (Muthusamy et al., 1994). We may distinguish three performance levels for humans. These are sketched in (Cimarusti and Ives, 1982):

- If the person is fluent in a certain language, the recognition is nearly instantaneous. Probably this works understanding single words and possibly recognizing a particular dialect reflected in the way of pronunciation. Things degrade in adverse conditions like huge background noise or mixed up speaking.
- On the next level are languages that are familiar to the listener and in which he has still certain competency. Usually languages of this class take somewhat longer to be recognized. Either we are looking for some known word (our dictionary being smaller) or we are gathering enough occurrences of characteristic sound structures, that are known by the listener.
- The weakest level of human language recognition could be called handling *out of set* languages. Here the human listener does not have much competency. He may still guess the kind of language, based on the overall stored linguistic knowledge and based on parallels or similarities to known languages. He may for instance be able to attribute the language to a family of languages.

The way the human works is based a lot on its knowledge and previous confrontations with languages. A big part is also due to the knowledge of similarities between languages, as their membership to language families.

— ◇ —

We may give the following interpretation of language recognition by humans: If the languages are well known by the listener, the recognition happens in a rapid way and is likely to use features of high linguistic levels, such as recognizing words. But the recognition seems to be based more on the acoustic (and maybe prosodic) level if the language has to be inferred from other, better known, languages of a same family for instance.

To make the link to automatic LR, we can for instance see that language "families" may be distinguished simply by working on statistics of the ratio between vowels and consonants and similar measures (Rouas and Farinas, 2004; Rouas et al., 2005) and also (Adda-Decker et al., 2003; Barry et al., 2003).

[6]The VOA dataset is likely to contain prepared speech as well.

Opinion: A good part of the observed links between languages can be explained etymologically (languages belonging to the same group or family). Despite this, some links or proximities that may appear can not be explained as simply as that. Such findings even awake the curiosity of linguists[7]. By own observation, some evidence may for instance be explained in terms of more technical linguistic cues. As an example, some grouping of languages seem to be linked to the occurrence possibility of consonant clusters (pronouncing two or more consecutive consonants). This is the fact that the phonology of certain languages is more permissive for consonant clusters whereas other languages are quite restrictive or do not even permit them[8]. As an other example, certain ways of observation seem to link Latin languages more to the Indian ones than to English. Finding an explanation to this is slightly more delicate[9].

1.3.3 Automatic approaches

The diverse approaches and techniques of automatic Language Recognition (LR) explore linguistic units at different levels. The rules describing these units are collected in appropriate models. They may operate on a low level such as purely acoustic models or explore high linguistic units like syllables or words. A brief overview is given in the following list, while acoustic modeling is detailed in Sect. 2.4 and other major approaches in Sect. 3.4.3:

Acoustic modeling: Statistical approach to model directly the probability of a language, given a set of features measured on the acoustic signal in regular intervals. This is done by calculating the likelihood that the observed data have been generated by a previously trained language model (using Bayes' Theorem). For modeling acoustics, most often, GMMs (or GMM based techniques like SVMs) are used. All works presented in this work are built on approaches working on this acoustic level.

Phonetic modeling: A statistical model based on the likelihood that an acoustic feature vector is part of a phoneme that itself belongs to a hypothesized language. Phonetic LR is typically based on the output likelihoods of one Hidden Markov Model (HMM) for each language, where each state reflects a phoneme or a broad phonetic class. In fact to be accurate, the units represented by the models are rather speech sounds than real phonemes as they are defined

[7] In reference to thoughts and discussions for instance during some Interspeech 2009 presentations of the linguistic track, as well as other observed discussions.

[8] On the permissive side may be found Indo-European (e.g. English, Spanish) and Eurasian languages (e.g. Hindi, Tamil) and on the restrictive side may be found North-Asian languages (like Japanese, Korean and Mandarin), whilst the majority of Malayo-Polynesian languages (e.g. Tahitian, Maori) are inhibitive. cf. *The World Atlas of Language Structures Online*, http://www.wals.info

[9] For instance Spanish and Hindi, which, at a quick glance, exhibit some very similar word roots, by comparing their Swadesh lists at http://en.wiktionary.org/wiki/Appendix:Swadesh_lists.

by linguists or phoneticians.

Phonotactic modeling: Statistical approach to the likelihood of a hypothesized language, given a detected (pseudo-) phoneme sequence. One global or several language dependent phonetic recognizers may be used. The succession of phonemes is basically exploited using language dependent *n*-grams. In the most basic approach, only the sequence of recognized phonemes is used for language modeling. But the Likelihoods generated as by-product by multiple recognizers may also be included into the system. The main approaches for the phonetic and the phonotactic levels are presented later on in Sect. 3.4.3.

Syllable modeling: Describes how syllables are composed of consonantal and vocalic phonemes. Syllables are, similar to context-dependent phonemes, derived from proper phonemes. This comprises statistics on the occurrence of different syllables, as well as modeling the structure of the syllables. As an example, languages that are called "open" have all syllables ending with a vowel[10].

Prosodic modeling: Describes the rhythm, the melody, the stress and the intonation of a whole phrase (prosodic unit) or a possible tone of a word (e.g. in Chinese or some African languages). Such prosodic features are more difficult to measure and come in a wide variety (e.g. Itahashi and Du, 1995; Yan et al., 1996; Hazen and Zue, 1997; Barry et al., 2003; Rouas et al., 2003).

Morpho-lexical language modeling: Describes the set of words that a language uses. This recognizes the words pronounced in the utterance using phone recognizers and language-dependent word lexica (Kadambe and Hieronymus, 1995; Hieronymous and Kadambe, 1996).

Syntactic language modeling: Describes the sequences of words by modeling the grammar of a language. The design and implementation of systems using linguistic units of such very high level is often considered too heavy and expensive compared to its benefit over phonotactic modeling.

duration modeling

Some of these approaches include or may be extended by a duration component that is taken into account by the system. So the duration of the units at different levels may be modeled as well and can include additional language-dependent information. In general, doing so tends to show its usefulness. Such observations may be found for instance in (Pellegrino et al., 1999a), which states "*segmental duration provides very useful information*".

system fusion

An automatic LR system exploits one or more of these modelization levels. When using more than one system, the individual results have to be fused together in order to produce a final result. There are some difficulties arising while trying to merge different systems because one of these may generally outperform others or may even have the tendency to produce false results under certain circumstances.

[10] This is typically the case for most Polynesian languages, as for example Tahitian, Marquesan or Hawai'ian.

Some appropriate tuning of the fusion weightings (e.g. Hazen and Zue, 1997) has to be found or some more evolved back-end has to be used (see also Sect. 3.5.2 for this).

best system?

When comparing single acoustic and phonotactic-based systems, neither is consistently more superior than the other. In fact, over history of LR, these two approaches take turn to outperform each other (Pellegrino et al., 1999a,b). Fusing systems of the two approaches is generally rather beneficial since they work on different levels and track different kinds of information.

our choice

Some of these systems require, according to the techniques used, certain phonetic and linguistic knowledge. The majority of the systems working on higher levels, use training data with annotations. For instance phonotactic systems traditionally require some amount of phonetically transcribed data to build one or more phone recognizers.

The phonetic transcription may also be obtained by automatic means where a textual transcription (which has to be provided) is transformed to a phonetic representation according to a phonetic lexicon or phonetization rules.

But some approaches have also been attempting to model higher level features that are obtained in an completely unsupervised manner without the need of transcriptions, as will be stated in Sect. 3.4.3.2. This allows to still use such higher level systems (for instance a phonotactic system) (Pellegrino et al., 1999b; Verdet, 2005).

— ◇ —

The need of transcriptions or more advanced techniques makes up one of the major bottlenecks in the development of higher level LR systems. The transcription process is very time-consuming, costly and error-prone. This is one of the reasons why we focus on acoustics-only systems in our works.

1.4 Variability compensation

After a review on the research domain and the requirements towards a Language Recognition system in Sect. 1.2 and Sect. 1.3, this section states more in detail the problem of the different variabilities which were shortly evoked in Sect. 1.1.

1.4.1 Speaker and channel variability

A recurrent difficulty is the fact that the speech signal includes all sort of information that is not relevant to the task of Language Recognition – such as speaker and channel dependent information. This makes systems with straightforward language modeling not meeting the requirements of being independent of the speaker and of the form (as described in Sect. 1.3.1). This constitutes the main motivation for the

present works. We will show a solution to this, that still meets the total automation condition and which uses purely data driven approaches.

— ◇ —

inter-speaker variability

Speaker variability includes various physical and behavioral features of the speaking person. For instance vocal tract configuration, gender and age or growth, but also culture, the native language or accents, as well as the nativeness or fluency in the currently spoken language.

intra-speaker variability

Intra-speaker variability, on the other side, comprises for instance the current emotion and health status, tiredness or the time of the day. It also includes the topic spoken about. This kind of variability is principally (but not exclusively) observed over several recordings of the same speaker. These are called *sessions*. Intra-speaker or session variability is an important factor for Speaker Recognition applications, but for Language Recognition, it may likewise be attributed to the speaker or the channel variability since these variabilities can not be measured separately, having only one session per speaker.

channel

Further, channel variability is contributed by different means of audio acquisition and transmission procedures. This includes room acoustics, the environment, background noise, microphone setup (the type of microphone or telephone used) and the distance of the speaker to it. Some effects may also be attributed to the speech signal encoding and the transmission channel used.

nuisance

In this work and in many other LR environments, we are mainly dealing with telephony signals (since this field is driven by NIST evaluations). The speaker and the recording condition potentially change from utterance to utterance. Because of that, we propose here to handle all this non-useful or perturbing information together. Further, we qualify it under the general term of *nuisance*. The observed data is thus composed of useful information, which is the information that depends on the language, and useless or even perturbing information that depends on the above mentioned factors.

The feature extraction and modeling strategy (e.g. with GMMs), as described in Chapter 2, should attempt to focus on the useful information and, at the same time, minimize the effect of the language independent, perturbing information.

on feature level

For the speech signal, it is well known that the usual feature extraction (Sect. 2.3) approaches can only partially discard perturbing information related to the recording setup and the transmission channel. Furthermore, a lot of the speaker dependent characteristics are kept in the features. Feature extraction is thus not optimal, since we do not know which parameters carry most useful information. In fact, we know few things about perturbing information. Therefore, we try to countervail this fact in the modeling stage (Sect. 2.4). This constitutes the topic of the works presented in this document. We should therefore seek modeling strategies that can naturally focus on the language dependent characteristics and that discard the language independent ones. The strategy we propose in this work is to explicitly keep track

of the language-independent variability, i.e. the nuisance.

model level

Regarding the modeling methodologies, most of them are statistical. To emphasize language dependent information and minimize the speaker, channel and session dependent information, the solution is usually to use a large set of training data. This data typically includes many speakers and has session and channel information which is similar to the one used in the testing conditions. In consequence the models become robust enough to the speaker variability, but less so to the session and channel variability.

A: augment data volume

The first solution, namely to increase the amount of data used for system training, is not ideal, since it tries to hide the problem about variability with the mass of data. Despite the use of larger and larger quantities of data, we usually still experiment a rather big sensitivity of the models to mismatched conditions between training and testing. So other techniques have emerged to compensate such mismatches.

B: using normalization

A second (and presumably better) way to a solution is applying normalizations at different levels. There has been considerable progress on normalization techniques to achieve robustness in the feature extraction step (e.g. Matějka et al., 2006). But such early-stage approaches like Vocal Tract Length Normalization (VTLN) or RelAtive SpecTrAl transform (RaSta) (outlined in Chapter 3) are not enough to remove all nuisance. Whereas on the side of modeling acoustic features, session compensation has been tried (Castaldo et al., 2007b; Verdet et al., 2009b). Despite these considerable advances, we may still see some disturbing sensitivity of the models to a mismatch of channels. This occurs for instance when working on data of different databases (Verdet et al., 2010b).

— ◇ —

The inter-speaker variability and the intra-speaker variability (as the channel effects) are likely to be of different natures since the origins of these two kinds of variabilities are different: on one side for instance the vocal tract or someone's way of speaking emanating from the speaker as human being and on the other side channel distortions or noise, which are more of technical nature. Also, in Speaker Recognition, they can be separated considerably well (by a technique introduced later on in this chapter). But even when likely being of different natures, we will try to handle them together in a holistic way, and simply englobe them under the nuisance term.

1.4.2 Variability between databases

Even if the approach proposed herein is of great use, the resulting models remain to a certain extent dependent on the global kind of data. This may be reflected in data coming from different corpora. We refer to such a global kind as *(channel) category*. We think this kind of nuisance of being even of another nature than the variabilities presented in last section.

As an example for big differences between databases, we analyze the fact that in 2009, it was the first time that NIST's Language Recognition Evaluation (LRE) (NIST LRE, 2009) included data of two quite different channel categories, namely the classical *Conversational Telephone Speech (CTS)* and data coming from telephone bandwidth parts of *Voice of America (VOA)* radio broadcasts (which constitutes the major part). This LRE dataset disposes of a total of 23 target languages. In Chapter 6, this work addresses also the challenge induced by the differences between these two LRE 2009 channel conditions and analyzes ways to cope with this problem.

— ◇ —

problem: variability

In summary, the term *nuisance* encompasses a number of phenomena including transmission channel effects, environment noise (other people, cars, TV, etc.), variable room acoustics (hall, park, etc.), the position of the microphone relative to the mouth and the variability introduced by the speaker himself. The solutions proposed in the literature involve works at various levels of the signal processing and modeling chain (feature space, model space and score space). In spite of using sophisticated feature extraction modules, the troublesome variability introduces a bias in estimated model parameters. This bias can dramatically influence the classification performance. This is mainly caused by the fact that the training databases cannot offer an exhaustive coverage of all the potential sources of variability. The retained solution consists of explicitly modeling this variability information.

1.4.3 Underlying model

Having stated the problem of variability, this section indicates how the solution of explicitly modeling the nuisance is motivated and how this might be realized. It gives a first presentation of the basic principle behind Joint Factor Analysis (JFA), which is used to cope with this troublesome variability.

— ◇ —

In spite of the research efforts in the fields of audio feature extraction and modeling, several domains of automatic speech processing have to face the problem of *nuisance* or *session/channel variability* stated above. This channel variability problem, related to changing acoustic/transmission conditions from one recording[11] to another, has been identified as one of the most important source of performance degradation in automatic Language Recognition (Kenny et al., 2005b; Vogt et al., 2005; Campbell et al., 2006b; Matrouf et al., 2007; Castaldo et al., 2007b; Kenny et al., 2008; Vogt et al., 2008; Matrouf et al., 2011).

solution: JFA

The commonly used approaches in statistical classifier training (such systems will be described in detail in Chapter 2) aim at estimating the parameters that characterize the target pattern (in our case the language) itself. But the variability is neither explicitly modeled in this training process, nor implicitly captured from

[11] This includes live acquisitions and working on streams as well.

(the incomplete) training corpora. Recently, in the context of the speaker verification task based on Gaussian Mixture Model (GMM)-Universal Background Model (UBM), a Joint Factor Analysis (JFA) paradigm was introduced. This models the speaker characteristics and the session variability at the same time, but as distinct components (Bimbot et al., 2004; Kenny et al., 2005b; Vogt et al., 2005; Matrouf et al., 2007; Kenny et al., 2008; Matrouf et al., 2011). The works presented herein build on this paradigm, but here it is applied to the field of Language Recognition.

basic decomposition

In terms of information contained in the observed speech utterances, we can decompose the model as follows: One component represents the information that is linked to the language of the utterance. This is the useful information about the target pattern. And another component represents the troublesome variability and is explicitly enumerated. So, using the term *model* (here with symbol M) as working entity that tracks information, we can write[12]:

$$M_{observed} = M_{language} + M_{\overline{language}} \tag{1.1}$$

where $M_{observed}$ is the statistical model of the observed data. We assume that it can be decomposed into the two components $M_{language}$ which is the statistical model for the language information, and $M_{\overline{language}}$ being the statistical model for the useless information. So the first underlying idea is to separate the useful and the useless information in order to model them separately and explicitly (but in the same run in the same context). This allows to estimate and to characterize the nuisance instead that it is just some noise in the target model.

The question which arises as next is how to separate the useful ($M_{language}$) and the useless ($M_{\overline{language}}$) components. One of the basic assumptions is that the language-dependent information can be estimated by finding the part that is common to all (training) utterances of a given language. We can easily think of it being the average over these utterances. On the other hand, the information which is not part of all utterances corresponds to the unwanted variability (speaker, microphone, noise, etc...). This would be the remaining part of each utterance, once the (language-dependent) average removed. This is the underlying strategy of JFA, which is basically a simple idea. The exact way it works and a ready-to-go implementation algorithm will be detailed in Chapter 4.

GMM-UBM approach

In this study, the model we propose is based on the use of the GMM-UBM approach, which will be presented in Chapter 2. The UBM is thought to be representative of the whole acoustic space of speech. In LR, this comes down to a kind of average, pooling all languages together. We focus on UBM-based GMMs and Joint Factor Analysis for their intrinsic simplicity.

other approaches

Some other statistical approaches, which are known for instance for speaker verification, may propose a slightly higher potential but can hardly be applied to LR out of the box. For example modeling the speaker using eigen-speaker

[12]This is just representative of what the decomposition aims to do and does not follow any mathematical formalism.

factors (Kenny et al., 2008), which has shown better results than standard MAP adaptation. In LR, Maximum Mutual Information (MMI) training of GMMs is said to be more powerful than standard MAP adaptation (Burget et al., 2006), but its training is very demanding in computation power and it has parameters that are difficult to tune. First experiences of the JFA method applied to LR have been proposed in (Verdet et al., 2009b; Brümmer et al., 2009) and integrating somehow differing strategies in (Castaldo et al., 2007b; Hubeika et al., 2008).

A similar approach to deal with troublesome variability was proposed by (Campbell et al., 2006b). This approach, named Nuisance Attribute Projection (NAP), works in Super-Vector[13] space and has hence to be used with an adequate classifier, for example SVMs. The JFA presented here works at GMM-UBM and at frame level. In this case the theoretical framework is more interesting, because it allows joint modeling of different kinds of information, which is not the case of NAP-SVM. The full JFA for Speaker Recognition is an example of such extended modeling, where, in addition to nuisance modeling, a part of the speaker component is also constrained to a low-dimensional subspace[14]. And the remaining part of the speaker component spans the whole SV space (Kenny et al., 2008; Burget et al., 2009). Very recently, JFA has also been applied to emotion detection (Dumouchel et al., 2009; Kockmann et al., 2009) and to speech recognition (Povey and et al., 2010).

1.5 Summary

The Language Recognition system to be developed in this document should be designed in such a way to be extensible to other languages without much effort and primarily not requiring annotated databases. It thus should be a language independent, data driven approach to automatic LR, which does not require any transcription or annotation. design

The burning problem of such a system is the variability introduced by the differences between speakers of the same language, recording setups, channel effects and similar troublesome factors. This leads the observed data of each language to be widely dispersed, compared to the spreading of the different languages between each other (Fig. 1.1). In order to reduce or to *compensate* this variability, we gave an initial introduction to the JFA paradigm. Having proved its usefulness in Speaker Recognition, this work introduces the same paradigm to the task of Language Recognition, which has its own particularities. problem

Another major problem is the discrepancy in the amount of data available for each language. Whilst there may be hundreds of hours of speech for some common languages, there may be just a handful of utterances for other rarer languages. The kind or the source of this data may also have a big impact, as will be shown (in

[13] A Super-Vector (SV) is usually obtained by stacking the parameters of a model into one big vector. More on this in Chapter 4 and Chapter 5.

[14] The concept of low-dimensional subspace will be introduced in Chapter 4.

Chapter 6) e.g. when our system is trained on broadcast transmitted data and we evaluate it using telephony data. Since the Joint Factor Analysis approach can not fully solve the issue, additional ways to cope with this will also be attempted.

1.6 Structure of this document and our related publications

This document is divided into two main parts: Part I comprises a general and detailed analysis and synthesis of the current state of the art. Chapter 1 provides an introduction to the field of research. Chapter 2 contains a step-by-step presentation of a baseline Language Recognition system. It is followed by some of the major advances on this domain in Chapter 3.

Part II comprises the different novelties spawned by our works, namely a presentation of Joint Factor Analysis (JFA) applied to the field of Language Recognition in Chapter 4 and an introduction of Support Vector Machines (SVMs) and their combination with JFA in Chapter 5.

- Results of these two chapters on NIST LRE 2005 data have been published in our INTERSPEECH 2009 paper entitled "*Factor Analysis and SVM for Language Recognition*" in Brighton: (Verdet, Matrouf, Bonastre, and Hennebert, 2009b).
- The same JFA system has been applied to NIST's 2009 data through our participation to the NIST LANGUAGE RECOGNITION EVALUATION 2009 and the associated workshop in Baltimore: (Verdet, Matrouf, and Bonastre, 2009a).
- Updated results of both systems and on both data sets have been published in an article entitled "*Modeling Nuisance Variabilities with Factor Analysis for GMM-based Audio Pattern Classification*", together with similar approaches applied to Speaker Recognition and Video Genre Recognition, in the COMPUTER SPEECH AND LANGUAGE journal edited by Elsevier: (Matrouf, Verdet, Rouvier, Bonastre, and Linarès, 2011).

JFA solving a main part of the variabilities, we still observe that this approach is not omnipotent in the context of very different databases being used (telephone vs. broadcast). In Chapter 6, we thus analyze how this may be solved.

- This analysis, result of Chapter 6, has been presented at ODYSSEY 2010, THE SPEAKER AND LANGUAGE RECOGNITION WORKSHOP in Brno under the title of "*Coping with Two Different Transmission Channels in Language Recognition*": (Verdet, Matrouf, Bonastre, and Hennebert, 2010b).

Finally, in Chapter 7, we focus on score-level merging of specialized systems by tackling different automatic channel category detectors, which opens the interesting perspective of an automated way to subdivide the data.

- These results have been published in INTERSPEECH 2010 in Makuhari under the title "*Channel Detectors for System Fusion in the Context of NIST LRE 2009*": (Verdet, Matrouf, Bonastre, and Hennebert, 2010a).

Why shouldn't we succeed? Admiral Anderson did.
— Captain Bligh, Mutiny on the Bounty, movie, 1962

Chapter 2

Fundamentals of a Language Recognition System

Contents

In this chapter, we will introduce all elements needed for a baseline system. The different steps required to build a Language Recognition system will be presented in order to inscribe subsequent analyses, improvements and novelties that work towards resolving the variability problem. The baseline system resulting from the composition of this succession of steps allows also to obtain first basic results. It provides also a benchmark which subsequent development is compared against. These results are included in later chapters together with our novelties in order to facilitate their comparison (Sect. 4.4, Sect. 5.4.1, Sect. 6.2 and partly Sect. 3.3.3).

— ◇ —

2.1 General classification system

This section gives a general overview over the different processing steps involved in a pattern recognition system, of which Language Recognition is a particular application. These steps will then be detailed in subsequent sections. They are presented in the context of Language Recognition (LR).

— ◇ —

pattern recognition

The aim of a pattern recognition system is to learn (by automatic means) a certain structure or pattern contained in the labeled training data and to recognize it again while provided with some unknown data. For each category of input data, a separate model is trained holding the pattern for this data category. Pattern recognition addresses classification problems since at the later testing stage, it tries to identify which category (or class) the unknown data belongs to. The models can be seen as templates and the data of unknown class attempted to be matched against them. In the scope of Language Recognition, each category or class generally corresponds to one language (for general references to LR, see Chapter 3).

Bayes' rule

The aim of a Language Recognition (LR) system is to find the most likely language l^*, given a speech utterance $\mathcal{X}$ (the observation) or in a slightly different form, detecting if a specific language is used or not. This may be expressed as:

$$l^*(\mathcal{X}) = \arg\max_{l \in L} [P(l|\mathcal{X})] \tag{2.1}$$

where $P(\blacksquare|\blacksquare)$ is the conditional probability. $P(l|\mathcal{X})$ is a sketch of what we are seeking and is thus called the *posterior* probability. By the fact, $P(l|\mathcal{X})$ is not directly

computable, but Bayes' Theorem (Bayes and Price, 1763; Laplace, 1986; Bernardo and Smith, 1994) and using flat language-priors allows to write[1]

$$l^*(\mathcal{X}) = \arg\max_{l \in L}[P(l|\mathcal{X})] = \arg\max_{l \in L}[P(\mathcal{X}|l)] \tag{2.2}$$

which uses the (computable) conditional Likelihood of the utterance, given a language model l (see also Sect. 2.5.1 on scores).

The above being the aim of a recognition system, let us describe how this is achieved. An overview of the typical steps composing a pattern recognition or classification system is given in Fig. 2.1 and described subsequently in the context of Language Recognition (LR). LR phases

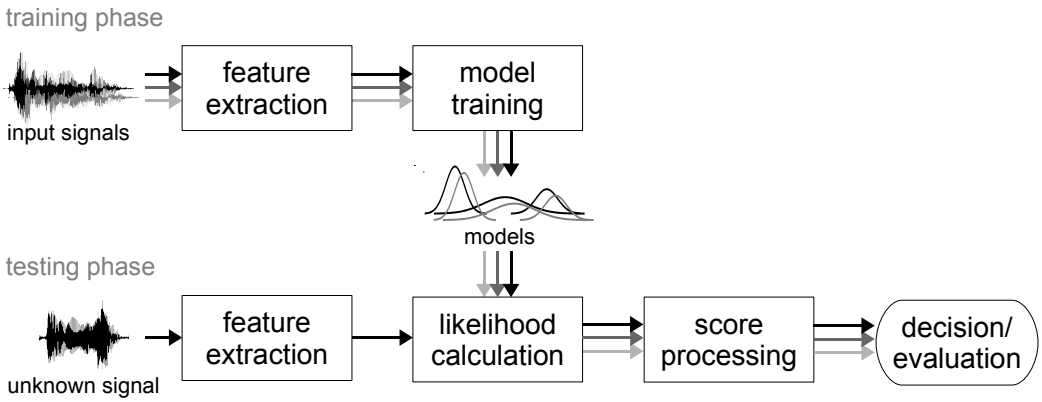


Figure 2.1: *The generic steps involved in any pattern recognition system*

In a pattern recognition system, two sequential phases can be distinguished: the *training* phase where the models are estimated using some training data and the *testing* phase where they are applied on unseen data.

The very first stage in both phases is the preprocessing, called Feature Extraction (FE) and its objective is to reduce the amount of input data (in our case the audio signal) to some useful and more manageable values (parametric representation), called features. The audio signal is cut into chunks of several milli-seconds each and converted to some parametric representation, the features. Every chunk produces one set of features, called a feature vector. They all have the same number of parameters and are therefore called being of d dimensions, since such a feature vector may be seen as a point in a d-dimensional space. A whole utterance (speech data) is thus transformed to a series of feature vectors. This step is presented more in detail in Sect. 2.3. feature extraction

The second stage in the training phase is one of the most most challenging parts. training

[1]Using Bayes' Theorem: $l^*(\mathcal{X}) = \arg\max_{l \in L}[P(l|\mathcal{X})] = \arg\max_{l \in L}[\frac{P(\mathcal{X}|l)P(l)}{P(\mathcal{X})}]$ where the languages l are represented by their respective models. Since $P(\mathcal{X})$ does not depend on l, this term can be omitted without changing the result. Moreover, if we have equal priors, as is the case for the NIST

It consists of training the models - usually one for each language. In the optics of the feature vectors being points in multidimensional space, the classes (here: languages) are often simplified as being concentrations (or clusters) of points in this space. So, training is the process of spotting and delimiting those clusters and possibly tracing decision boundaries between clusters. This is achieved by applying statistical analysis algorithms to the training data. The basic approach consists in estimating a multivariate mixture (weighted sum) of Gaussian distributions over the feature vectors belonging to one class in order to shape them most conveniently. These are called Gaussian Mixture Models (GMMs) and are described in Sect. 2.4.

testing

The key step of the testing phase is the application of the system to some new data, the class of which is unknown. It is called testing and involves calculating the likelihood of the data, given each of the language models in turn (Eq. 2.2). When an utterance of unknown class is presented to the system, it first undergoes the very same Feature Extraction step as the training data. Then, the system calculates the similarity of this data against each of the stored templates (the models). This usually yields a correlation or similarity likelihood, which is called the score.

score normalization

The score then undergoes some score processing in order to be normalized in different ways. This step is partly dependent on the final task, the kind of decision and evaluation that will be carried out. Likelihood scoring and score processing are presented in Sect. 2.5.

decision/ evaluation

The normalized scores are then used to take some decision or used for system evaluation. In a real-world recognition system, this decision would typically be a hard decision (*Yes/No* answers) and may result from comparing the scores to a previously fixed threshold. In a research context, the system performance is usually calculated using different thresholds in order to evaluate the system over a wide range of applications. In such a context, the best threshold may also be determined a posteriori (Sect. 2.6).

confidence

The normalized scores are at the same time an indication of confidence into the (hard) decision obtained by thresholding. To be able to give a confidence interpretation to the scores, they have to be normalized (inter-utterance normalization, Sect. 2.5.2). The confidence in scores that lay far over the threshold is high, whereas the confidence into scores near the threshold is more limited.

— ◇ —

We saw that a pattern recognition system is composed of two phases, the *training* phase with preprocessing and model training, which uses data of a training corpus, and the *testing* phase where unseen and unlabeled data undergo preprocessing and classification, followed by some means of evaluation of the results.

LRE protocols (NIST LRE, 2005, 2009; Greenberg and Martin, 2009), the priors' term $P(l)$ will be the same for all l and can thus also be omitted. We finally end up with Eq. 2.2.

2.2 The speech signal

Because of historical reasons, most Language Recognition (LR) systems operate on and are tuned to telephonic speech data. So the data usually comes with a constant sampling rate of 8 kHz and a 8 bit μ-law resolution (nowadays, it may typically have been downsampled from 16 kHz recordings).

The speech signal is assumed to be stationary on a short term (this is fairly true in the middle portion of phonemes). But obviously it is highly non-stationary on the longer term. Recordings have lengths of a few seconds up to full half-hour conversations. Since the sampling rate is 8 kHz, the Nyquist limit of extractable frequencies is 4 kHz, which covers a good portion of frequencies that are present in speech, but which is far from being optimal[2]. This counts among the issues about speech data quality (Sect. 1.3.1).

2.3 Feature Extraction

As described in Sect. 2.1, a first step called Feature Extraction (FE) is needed to cut down the audio data to a sequence of more manageable vectors, possibly normalized and represented in a domain that is more convenient to model. This step is carried out as well in the training, as in the testing phase, in an independent manner. We present here the different operations which make up FE (see also Fig. 2.2).

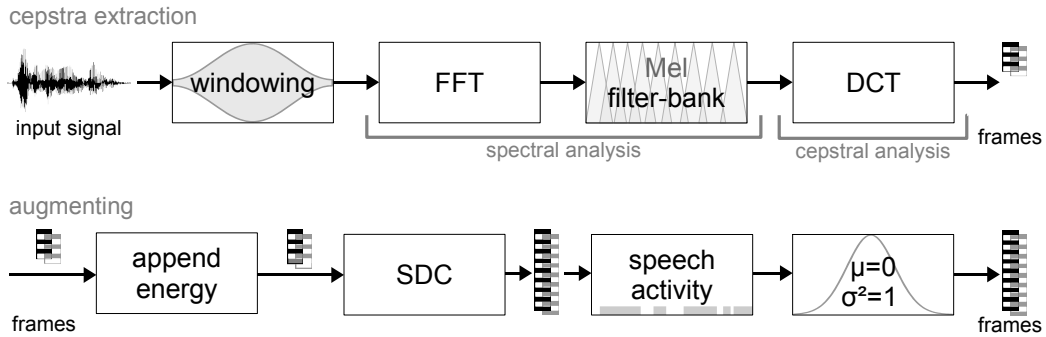


Figure 2.2: *Processing steps of Feature Extraction for obtaining cepstral coefficients and appending SDCs*

2.3.1 Cepstral coefficients

The very first step of FE cuts down the sampled and digitized speech signal to parametric feature vectors, used for further processing. The parameterization

[2]For instance, it becomes hard to distinguish between some fricatives if we are limited at 4 kHz.

chosen for the baseline system is based on a cepstral representation.

— ◇ —

sliding window

We used a sliding window of Hamming type with a length of 20 ms and a shift of 10 ms. This allows on the one hand to take enough samples to be able to reliably estimate frequencies at a given time t and on the other hand it assures that the observation is not too far off the assumption of the signal being stationary in that portion. This window configuration has a long tradition in speech recognition and corresponds more or less to the time-frequency resolution capability of the human ear. So one *frame* will be extracted every 10 ms by the subsequently presented steps.

spectral analysis

Spectral analysis is carried out applying a Fast Fourier Transform (FFT) on the samples contained in the window. Then, a bank of filters, non-linearly warped by the Mel function, is applied. The Mel scale tries to meet towards human spectral resolution capacities and is less sensitive to harmonic frequencies.

cepstral analysis

An additional Discrete Cosine Transform (DCT) is chained on top of the log-spectrum in order to perform cepstral analysis. Whilst the (frequency) spectrum is the result of analyzing periodicity in time (the frequencies contained in the signal), cepstral values depict periodicity in the frequency space. It thus detects salient frequencies together with their harmonic frequencies (which are roughly repetitions at integer multiples of the base frequency) and has therefore the effect of greatly reducing the impact of the harmonics to the other coefficients.

Cepstral analysis can thus be seen as computing the intensity of main basic frequencies together with their harmonics. If SDCs will be appended (next paragraph), we retain only the first 6 coefficients[3], which can be motivated by the strong energy compaction property of the DCT[4]. These 6 cepstral coefficients, together with the frame energy, form the *basic feature vector*, which is often referenced as MFCC vector. Cepstral coefficients are supposed to be an adequate representation of the speech signal for various automatic speech processing tasks.

SDC

Instead of appending deltas and double-deltas to the extracted coefficients, as it is common for speech or Speaker Recognition (SR), the more elaborate Shifted Delta Cepstras (SDCs) are appended. As most acoustic LR systems, SDC vectors in the 7-1-3-7 configuration and based on MFCCs are used (as (Burget et al., 2006; Torres-Carrasquillo et al., 2002b; Campbell et al., 2004; Castaldo et al., 2007b; Matějka et al., 2006) and a majority of the NIST LRE 2009 participants). The values are the parameters *N-d-P-K*, where N is the number of basic features (6 MFCCs plus energy in our case). The remaining parameters signify to append K delta-blocks (7 in this case), each shifted by P frames (here 3) and each being the differences (*delta*) between frames $t + d$ and $t - d$ (d being 1). We thus obtain a final feature vector size of 56.

[3] Speech and Speaker Recognition classically work on 12 or 19 coefficients.

[4] See http://en.wikipedia.org/wiki/Discrete_cosine_transform, and K. R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, Advantages, Applications", Academic Press, Boston, 1990.

They take into account signal evolution over many frames. The present configuration uses static cepstra of a window of 21 frames to compose one final SDC vector. It thus tries in a certain manner to include the evolution of the signal into the feature vectors.

2.3.2 Speech activity detection

In most acoustic speech processing applications, signal parts without speech are removed more or less aggressively. Especially silence parts do not carry any linguistic information (but it possibly is of some importance for SR as it might be an indication of fluency or separating individual words in speech recognizers). Because of its relative simplicity, often just basic silence removal is used, as opposed to more sophisticated methods to remove also regions of music or noise.

energy based

The easiest way to segment a stream of vectors into silence and non-silence regions is to model the energy component. The first approach is to set a threshold below which a frame is considered containing silence and above which it is assigned to speech. This threshold may be fixed beforehand, but in this case the energy component has to be normalized to achieve usable results. A better variant is to automatically determine this threshold for each utterance separately (a posteriori). This can be achieved by training small models on the energy parameter of one utterance only.

smoothing

Since such frame based labeling may be jerky in the middle-energy region, a supplementary step may be required to smooth the segmentation in order not to have too short segments. Thus to introduce some stability to the decisions. This is especially necessary if we use SDC type features since they use a certain number of consecutive frames for their calculation.

motivation

So only the segments labeled as speech are used in any further processing. Speech activity detection (SAD), voice activity detection (VAD) or silence removal is carried out to retain only regions containing speech. Amongst others, this avoids that a part of the (GMM) model is trained on frames containing silence. This would not be ideal since in general, silence matches rather well silence[5] and thus the model would yield good matching likelihoods for silence frames. But since silence does not contain much acoustic or linguistically relevant information, this artificially high likelihood overwhelms the true likelihood computed from frames actually containing speech.

importance

The step of speech activity detection is nevertheless an important step. For SR, (Scheffer, 2006, Sect. 5.3.2, pp. 71f) shows considerable sensibility of the system to the frames selected as speech. By varying just one threshold controlling parameter, the speaker verification system can make up to roughly 70% relative more errors. Some of our analyses regarding this particular topic are presented in Sect. 3.3.4.

[5] The variance being quite small.

Opinion: Speech activity detection is a quite hard domain since it is, up to some degree, dependent on the further processing objective (i.e. in the case of SDCs). For some applications like speaker verification, it seems beneficial to be strict in discriminating speech/non-speech frames on a nearly per-frame basis. However, for instance in the case of SDC parameters, too tiny segments may be damaging as the introduced discontinuity does not work well with SDC's window (or lookahead). The results may vary quite a lot, depending on the speech activity detection approach chosen.

2.3.3 Feature normalization

As last step of the FE process, the feature vectors undergo a normalization process in order to remove one part of the variabilities pointed out at the beginning (Sect. 1.1).

— ◇ —

$\mu = 0$
$\sigma^2 = 1$

Here, all vectors are normalized on a per-utterance basis by subtracting the mean from each parameter (also known under the therm of Cepstral Mean Subtraction (CMS)[6]) and by dividing it by the variance. In this way, the new mean of an utterance's features is 0 and their variance is 1. This allows to transform the parameters of different files to a common (standard) range, which has a normalizing effect. It also equalizes the ranges of the different parameters among each other.

2.4 Modelization

In the training phase, after FE, individual language models have to be estimated. The goal of the training step is to use a set of (training) data samples of a given class in order to estimate a model, which will be representative of that class. There are different ways such models may be designed. We may principally distinguish between two types of models:

Generative models try to mold at best how the data of a given class is structured, how its points are distributed. A well known kind of generative models is the Gaussian Mixture Model (GMM), which will be presented hereafter and which will be used in the baseline system. The GMM structure serves also as basis for other types of models like SVMs (Chapter 5), where the GMM is transformed to one Super-Vector. Further, in Hidden Markov Models, several GMMs are chained together by transition probabilities.

Discriminative models do not focus on the distribution of the data of one class, but rather on the separation between the target class and its competitors. The most well known representative of this kind of models is the Support Vector Machine (SVM), which will be used and discussed in Chapter 5.

[6] Even if the energy and the delta blocks, which undergo the same processing, are not really cepstra.

Once a specific type of models has been chosen, an appropriate strategy to estimate its parameters has to be found. In this section will be presented one possible way to address GMM modeling, namely the GMM-UBM approach. We use this approach, as it is in a large majority of other works in the automatic speech processing domain. And above all, it serves as constituent basis for other approaches, like JFA and SVMs, used in subsequent developments.

As first step, a Universal Background Model (UBM) is estimated using the Expectation Maximization (EM) algorithm with a Maximum-Likelihood (ML) criterion. This UBM will then be adapted through the MAP strategy to obtain all necessary language models. Language models may also be obtained, with similar results, directly through EM-ML, but for several subsequent processing steps this approach is not appropriate[7]. Approaches that work on other linguistic levels (as defined in Sect. 1.3.3) will not be explained, since we chose a purely acoustic approach due of its simplicity and because it does not require any annotated data (as would probably be the case i.e. for training a phone recognizer).

2.4.1 Gaussian Mixture Model

We first need to introduce the structure of a Gaussian Mixture Model (GMM), before their estimation can be presented. This section thus gives a constructive approach to the GMM, starting with a single simple-valued Gaussian.

Gaussian pdf

The Gaussian distribution is also known as normal distribution. A Gaussian distribution is characterized by two values, the mean μ and the variance σ^2 and can be written as follows:

$$\mathcal{N}(\blacksquare|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(\blacksquare-\mu)^2}{2\sigma^2}} \tag{2.3}$$

multivariate Gaussian

Since we work in a feature space that has many dimensions (here: d), the distributions are *multivariate*, which means that the Gaussian extends into multiple dimensions. Thus, the mean is a vector and the variance becomes a covariance matrix Σ (which is symmetric positive definite). In fact, most implementations use a diagonal covariance matrix, which comes down of having only each dimension's variance. This allows for a consequent speedup, since its inversion (i.e. in Eq. 2.4 hereafter) becomes trivial. The matrix form notation of Eq. 2.3 is thus:

$$\mathcal{N}(\blacksquare|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\, e^{-\frac{1}{2}(\blacksquare-\mu)^{\mathrm{T}}\Sigma^{-1}(\blacksquare-\mu)} \tag{2.4}$$

As Fig. 2.3 illustrates, in the one-dimensional case, a (Monte-Carlo) sampled Gaussian density has a distribution of points on the only dimension axis, where they

[7] Using MAP adaptation, the relative configuration of the single Gaussian components does not change a lot. In consequence, we can make the assumption that the single components still correspond to the original ones. We thus can take corresponding components of multiple individually adapted models as being linked. This constitutes a precondition for several subsequently described techniques.

lay nearer together near the mean value. If we draw the distribution function (or a histogram in the discretized case), it resembles to the well known bell line. In two dimensions, the points are dispersed on a plane and cluster together near the mean. The distribution (as third axis) forms a bell or simple hat. In three dimensions, the points crowd together in a cloud (the distribution can not be imagined in a convenient way since we exhausted our easily understandable three dimensions). In the present case, we run with 56 dimensions.

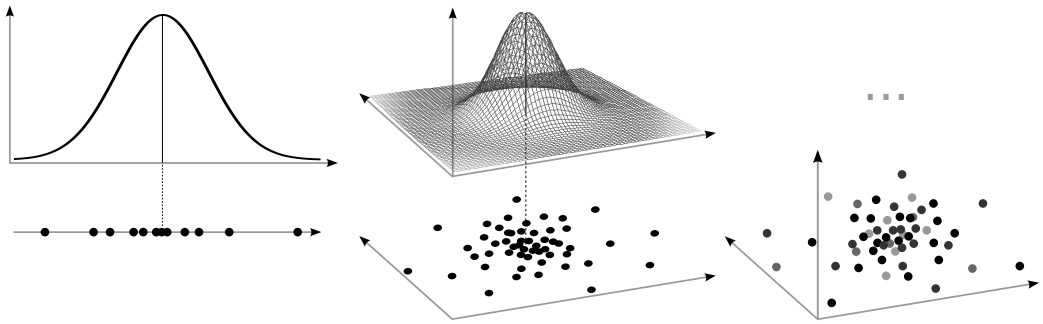


Figure 2.3: *In the lower row, first column shows points randomly drawn from a one-dimensional Gaussian distribution, second column from a 2-dimensional and third from a 3D one. The upper row shows the corresponding density functions.*

The curse of dimensionality

Excurse: With increasing number of dimensions, the space gets less and less populated, what is known as *The curse of dimensionality*, a term introduced by Bellman in 1957 (Chen, 2010; Bellman, 1957b,a; Gutierrez-Osuna). It states also that, if we exceed a certain limit in the number of dimensions for a given number of samples, it may even be beneficial to discard some dimensions in order to increase the population "density". Friedman (Friedman, 1997) states that the complexity of density estimation functions increases exponentially with the number of dimensions, which in turn requires a higher population density in order to be estimated well enough. Further, the multivariate Gaussian is one of the only density functions for high dimensional spaces.

Gaussian Egg

Excurse: Another interesting fact about multivariate Gaussians is the *Gaussian Egg* (Chen, 2010; Lawrence and Barker, 2009): Data sampled from a high dimensional multivariate Gaussian density lives in a "shell" around 1σ from the mean μ of the Gaussian. The distribution of the mass of a one-dimensional Gaussian is well known: about 65.8% falls within 0.95σ, 4.8% between 0.95σ and 1.05σ and the remaining 29.4% lay further than 1.05σ from the mean. Analyzing this distribution while augmenting the dimensionality shows that the mass of the innermost part decreases (more or less logarithmically) and the mass of the thin "(egg) shell" around 1σ augments rapidly, until reaching roughly 98% at about 1024 dimensions. In analogy, we can imagine the mass (area or volume) ratio of a circle inscribed into a square (2D case)[8] or a sphere inscribed into a cube (3D)[9]. It has a very similar

[8] 2D: (working with unit radius r) $V_{\bigcirc} = \pi r^2 = \pi \quad ; \quad V_{\square} = (2r)^2 = 4 \quad ; \quad ratio = \frac{\pi}{4} \approx \mathbf{0.8}$

[9] 3D: $V_{\bigcirc} = \frac{4\pi}{3} r^3 = \frac{4\pi}{3} \quad ; \quad V_{\square} = (2r)^3 = 8 \quad ; \quad ratio = \frac{4\pi}{3 \cdot 8} = \frac{\pi}{6} \approx \mathbf{0.5}$

behavior. With about 10 dimensions, the volume occupied by the hyper-sphere becomes negligible with respect to the volume of the bounding hyper-cube[10].

mixture of Gaussians

Since the observed data usually does not have a Gaussian-like distribution, but a far more complex structure (its histogram does not resemble to a Gaussian), we try to mold it by multiple Gaussian distributions that get summed together. Each Gaussian has an own mean, variance and weight. The weight is the ordinate scale and corresponds to the "maximal height" of the Gaussian. Gaussian Mixture Models are thus linear combinations of multivariate Gaussian density functions. A GMM is defined by a set of G Gaussians along with their associated weights α_g ($g \in 1, ..., G$):

$$P(\blacksquare|\Lambda) = \sum_{g=1}^{G} \alpha_g \mathcal{N}(\blacksquare|\mu_g, \Sigma_g) = \sum_{g=1}^{G} \gamma_g(\blacksquare) \tag{2.5}$$

where we define γ_g as the posterior probability of one Gaussian g (including its weight; see also Eq. 2.7) and Λ as the set of all model parameters of a GMM (this is all we need to characterize a GMM):

$$\Lambda = \{\alpha_1, \mu_1, \Sigma_1, ..., \alpha_G, \mu_G, \Sigma_G\} \tag{2.6}$$

total variance

The variance of the training data is not only contained in the covariance Σ, which might be diagonal or full. A certain part of the variance is also taken up by the fact that we use multiple Gaussians in our mixture. The set of all frames of the training data gets somehow divided into several subsets by the mixture of Gaussian components (if we consider that each frame is assigned to only one Gaussian). The variance of each component is smaller than the whole data's variance since each subset is built up only by frames that fall into the neighborhood of the components mean. The (total) variance of the whole data could be obtained if we use only one Gaussian component. In summary, the variance of the data is represented by the (co-)variances Σ, as well as by the distribution of the means μ. Both together make up the total variance of the training data[11].

— ◇ —

A GMM can thus be fully characterized by the set of its model parameters $\alpha_g, \mu_g, \Sigma_g$ for $g \in 1, ..., G$. They can be estimated from the observed vectors, which

[10] General formula for the volume of a sphere in n dimensions. To simplify, we use unit radius ($r = 1$). Basic formulas can be found on http://en.wikipedia.org/wiki/Sphere and rely on the surface A, which uses Euler's Gamma function $\Gamma(n) = (n-1)!$.

$V_{\bigcirc} = A_{\bigcirc} \cdot \frac{r}{n} = \frac{A_{\bigcirc}}{n} = \frac{2\pi^{\frac{n}{2}}}{n \cdot \Gamma(\frac{n}{2})} = \frac{\pi^{\frac{n}{2}}}{\frac{n}{2} \cdot \Gamma(\frac{n}{2})} = \frac{\pi^{\frac{n}{2}}}{(\frac{n}{2})!}$ for the case n is even.

For $n = 10$ we have thus: $V_{\bigcirc} = \frac{\pi^5}{5!} \approx 2.55$

and $V_{\square} = (2r)^n = 2^{10} = 1024$, which gives a *ratio* of about **0.0025**

[11] The same fact can be seen in Linear Discriminant Analysis (LDA), where we speak more of *scatter* than of variance. In fact, LDA even relies on the distinction between these two kinds of scattering, within-class (here the covariance) and between-class (distribution of the means).

are samples or emissions of the underlying model. The most common algorithm for ground-up estimation of these parameters is presented in the following section.

Fig. 2.4 depicts the derivation of the GMM from a single one-dimensional normal distribution as it was explained in this section.

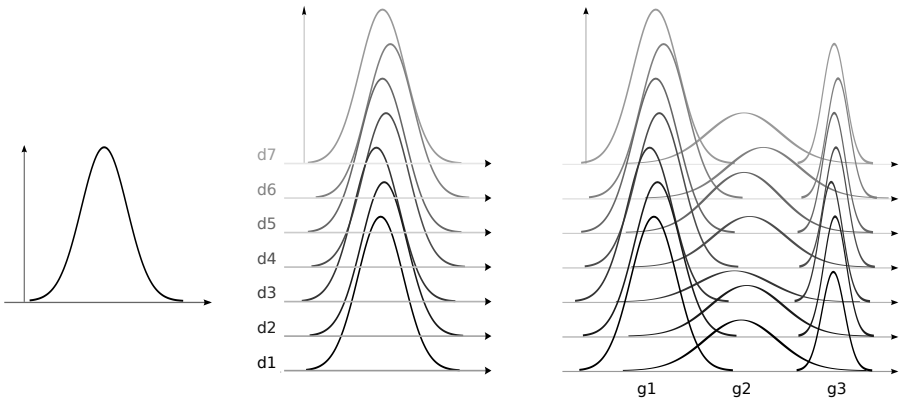


Figure 2.4: *Illustration of a single one-dimensional Gaussian (left), a single multivariate (center) and a mixture of multivariate Gaussian densities (right) with 3 Gaussians and 7 dimensions (represented in a stacked way)*

2.4.2 Expectation Maximization

The Expectation Maximization (EM) algorithm is a basic approach for estimating model parameters. Combined with a Maximum-Likelihood (ML) criterion, this algorithm finds ML Point-Estimates (MLEs) of the parameters of a statistical model (a GMM in the present case), where some model parameters depend on unobserved latent variables. While the standard reference for this procedure is (Dempster et al., 1977), it has first been tackled for gene frequency estimation in (Ceppellini et al., 1955) and in the context of Hidden Markov Models (HMMs) by (Baum et al., 1970). The EM algorithm is presented here, since it is used to obtain the UBM (using the ML criterion, as described in Sect. 2.4.3). It constitutes also the basis for the MAP adaptation (Sect. 2.4.4).

— ◇ —

iterative E +M steps

Starting with some initialization, EM is an iterative process over two steps: An expectation (E) step and a maximization (M) step. In the E step, the attribution of the data (feature vectors) to the different elements of the actual model (the single Gaussians) is gathered. These assignments can not be observed directly and constitute thus the hidden or latent variables of our model. We will see in the details of the M step that it is sufficient to retain certain statistics of these attributions. In the M step, the model parameters are re-computed as to maximize the likelihood with respect to the data that was assigned to them in the E step (ML criterion).

Citing the English wikipedia site[12], "*EM is particularly useful when the likelihood is an exponential family: the E-step becomes the sum of expectations of sufficient statistics, and the M-step involves maximizing a linear function. In such a case, it is usually possible to derive closed form updates for each step, using the Sundberg formula.*" Exactly this will be shown in the following subsections.

The Expectation Maximization algorithm is a general estimation approach. In its M step, this algorithm updates the model parameters as to optimize a certain criterion or objective function, which has to specified. In the case of EM-ML, the Maximum-Likelihood (ML) criterion is used. It will be introduced at the given place. criterion

2.4.2.1 Initialization

The EM algorithm requires the model parameters to be initialized in order to update them by consecutive E- and M steps. The choice for the initialization is rather important, since it may result the algorithm in converging to some more or less expressed local minimum. By changing the way of initialization, convergence may be different. This is specially the case when EM uses an ML criterion, described hereafter.

In the approach chosen in for this work, the model parameters are initialized as follows: The dimensionality of the multivariate Gaussians is fix since it is given by the cardinality of the feature vectors. The number G of Gaussian mixtures in the GMM is set in advance and does not change[13]. The weights of the Gaussian components are initialized as being all equal (thus $1/G$ each). The initial values of the means and the covariance matrices are calculated choosing for each Gaussian an equally sized random subset of the training feature vectors (at most one G-th part). our choice

2.4.2.2 Expectation step

The first step of the iterative EM algorithm is the *Expectation* (E) step. It gathers sufficient statistics, based on the model initialization or the model issued from the previous iteration.

As simplified view, it can be stated that each Gaussian's parameters are calculated using the data points (vectors) for which the actual Gaussian is the nearest one (that yields the highest posterior probability, E step). The new weight of the Gaussian will be the fraction of the data points that got assigned to this Gaussian.

[12] http://en.wikipedia.org/wiki/Expectation-maximization_algorithm (accessed in November 2010, it also gives a series of useful references).

[13] Using different approaches, it could also be set to one at the beginning and be split to obtain more and more Gaussians as the iterations go on. Some of such approaches split only the "heaviest" Gaussian, other do binary splitting (ex. (Verdet, 2005)).

The mean of the Gaussian will be the mean value of the assigned vectors (for each dimension) and the variance is also calculated on these vectors only. This corresponds to the M step.

feature's posterior probability

This simplified view can be generalized such that a vector is not assigned to only one Gaussian, but to all Gaussians (or a fixed number for speed-up) in a weighted manner. Thus the latent variables are not discrete vector-to-Gaussian assignments any more, but rather assignment probabilities[14]. Each vector x has an influence that depends on the proximity to the Gaussian g[15]. In fact, its influence weight is the (a posteriori) probability that it has been emitted by the Gaussian that is defined by the parameter estimates of the current iteration (in Eq. 2.5, this corresponds to the summand for Gaussian g):

$$\gamma_g(x) = P(x|\alpha_g, \mu_g, \Sigma_g) = \alpha_g \mathcal{N}(x|\mu_g, \Sigma_g) \tag{2.7}$$

sufficient statistics

For each Gaussian, we define zeroth (also called occupation), first and second order statistics that will be used in the M step. They are collected over the set $\mathcal{X}$ of all training feature vectors of the model's language as follows (not indicating the index for the language):

$$\begin{aligned} \vartheta_g^0(\mathcal{X}) &= \sum_{x \in \mathcal{X}} \gamma_g(x) \\ \vartheta_g^1(\mathcal{X}) &= \sum_{x \in \mathcal{X}} \gamma_g(x) \cdot x \\ \vartheta_g^2(\mathcal{X}) &= \sum_{x \in \mathcal{X}} \gamma_g(x) \cdot x^2 \end{aligned} \tag{2.8}$$

2.4.2.3 Maximization step

Maximum-Likelihood criterion

In the *Maximization* (M) step, the model parameters get re-estimated by trying to maximize a certain objective function. In the EM-ML case described here, it is the Maximum-Likelihood (ML) criterion. This maximizes, as its name indicates, the average frame (log-)Likelihood. This is calculated on the set of all training feature vectors $\mathcal{X}$. Relaying on Eq. 2.5 and Eq. 2.7 (see also Eq. 2.17), it is given by:

$$\hat{\Lambda} = \arg\max_{\Lambda} P(\mathcal{X}|\Lambda) \tag{2.9}$$

$$\mathcal{F}_{ML}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \log \left(\sum_{g \in G} \gamma_g(x) \right) \tag{2.10}$$

model re-estimation

Solving the maximization problem of this objective function results in following

[14] Readers knowing Hidden Markov Models (HMMs) may see that this generalization corresponds to replacing discrete state assignments used in the Viterbi algorithm by vector-to-state assignment probabilities and using Baum-Welch algorithm.

[15] By taking also into account the Gaussian's variance and weight.

closed-form update formulas for the Gaussian's new weight, mean and variance respectively (the model parameters $\hat{\Lambda}$). They use the sufficient statistics gathered by the E step:

$$\hat{\alpha}_g = \frac{\vartheta_g^0}{\sum_{g \in G} \vartheta_g^0} \quad ; \quad \hat{\mu}_g = \frac{\vartheta_g^1}{\vartheta_g^0} \quad ; \quad \hat{\Sigma}_g = \frac{\vartheta_g^2}{\vartheta_g^0} - \hat{\mu}_g^2 \tag{2.11}$$

The model defined by these new parameters reaches the maximum (log-) Likelihood on the above statistics and is thus a ML Point-Estimate (MLE). But since the underlying model changed, the statistics do not match it any more and the whole process has to be iterated. The latent variables (expressed under the form of the statistics) and the model parameters are interdependent – we thus hit a chicken-egg problem and this new model becomes the input for the next EM iteration and we start over with the E step.

convergence

For the ML criterion, this process converges usually with about 5 to 10 iterations, but the obtained result is not necessarily the best possible, since it may have converged to a local maximum. This local maximum largely depends on the initialization step.

— ◇ —

Compared to the MAP adaptation (presented in Sect. 2.4.4), EM with an ML criterion is computationally more expensive because several iterations are required in order to converge. In addition, MAP has the advantage to work considerably well with a limited amount of data. Using the MAP criterion, it may globally seem to have better convergence properties since the models of all languages have the same initialization (the UBM). This probably builds models that are better balanced between each other. Further, MAP's a priori model can be seen as a mechanism, which at each iteration pushes the model out of a possible local maximum.

2.4.3 Universal Background Model

The GMM-Universal Background Model (UBM) framework is a long established standard in speaker verification (Reynolds, 1997; Reynolds et al., 2000; Bimbot et al., 2004). It was developed out of a strategy, where speaker identification scores were normalized using a set of background speakers. Typically (and in contrast to the UBM), one model was estimated for each background speaker (Reynolds, 1995). Now, the GMM-UBM approach is used in a large variety of audio classification tasks such as speech recognition, Speaker Recognition, Language Recognition or even (video) genre recognition.

— ◇ —

The UBM is a GMM that is representative of all possible observations of the acoustic speech space. It is of general nature and usually independent of the task. So theoretically the same UBM may be used for SR or LR (for the same type of

parameterization). In the present case, it is thought to represent a kind of average over all languages. It is sometimes also called the World Model in reference to all possible languages (or speakers) of the world. The UBM can also be seen as a sort of neutral, universal language.

The UBM is estimated using the EM algorithm with an ML criterion, presented in Sect. 2.4.2 in order to obtain a model from scratch. This model will be a good fit[16] of the predefined number of Gaussian distributions to the data.

For each target pattern (a language in our case), a specific GMM will be obtained by adapting the UBM via the MAP criterion (Gauvain and Lee, 1994; Reynolds et al., 2000). This step will be explained in the next section.

2.4.4 Maximum A Posteriori adaptation

The goal of using Maximum A Posteriori (MAP) (Gauvain and Lee, 1994) is to start with a robust model (in occurrence the UBM) as an a priori model. Some of its parameters are then adapted towards the training data of a specific language. This keeps some generality of the UBM at the same time as fitting well to the target language. The particularity of this method is to work still well when there is not much training data available for a specific language. This is caused by the fact that far more data has been used for training the UBM, which gives it the robustness.

In our case, as is classically done by the LIMSI[17] (Gauvain and Lee, 1994) and the MIT[18] (Reynolds et al., 2000), only GMM means are adapted. The other GMM parameters (variances and weights) are taken from the UBM without any modification. This is the main reason for the robustness of the resulting models.

MAP mean update

In terms of means, we can describe the adaptation (for one Gaussian component, whose index we suppress) as:

$$\hat{\mu}_{client} = (1 - \alpha)\, \mu_{UBM} + \alpha\, \mu_{client} \tag{2.12}$$

where α is a factor which controls the weighting of the new (one-step) EM-ML language model (μ_{client}) to the UBM. It can be seen as a parameter avoiding overfitting.

In fact, usually only one iteration of EM-ML is used. We may show that in this case, only the E step is really necessary and that the MAP adaptation can be written directly in terms of the sufficient statistics.

$$\hat{\mu}_{client} = \frac{\vartheta^1 + \tau\, \mu_{UBM}}{\vartheta^0 + \tau} \tag{2.13}$$

[16] But probably not the best possible, due to the local maximum property of ML.

[17] Laboratoire d'Informatique pour la Mecanique et les Sciences de l'Ingenieur (Computer Sciences Laboratory for Mechanics and Engineering Sciences), CNRS UPR 3251, Orsay

[18] Massachusetts Institute of Technology, Cambridge

where $\tau = (\frac{1}{\alpha} - 1)\vartheta^0$ is the MAP *regulation factor* which regulates the strength of the a priori model (μ_{UBM}) with respect to the statistics of the observed data. For the more commonly written formula Eq. 2.12, α is expressed as function of τ and the occupation counts ϑ^0 as: $\alpha = \frac{1}{1 + \frac{\tau}{\vartheta^0}} = \frac{\vartheta^0}{\vartheta^0 + \tau}$.

Further on, an other a priori model is required for obtaining the required statistics, that reflect the latent variables. This can also be seen as the initialization model for the EM-ML part (μ_{client}) of the MAP adaptation[19]. Usually, MAP adaptation is carried out in one run, but it could also be refined iteratively. Fig. 2.5 shows the two models required as input and indicates the iterative working. Most often, the same model is taken as well for EM-ML initialization, and also as a priori model for the adaptation. In potentially further iterations, the statistics are calculated using the new model estimate, whereas MAP's a priori model still remains the UBM.

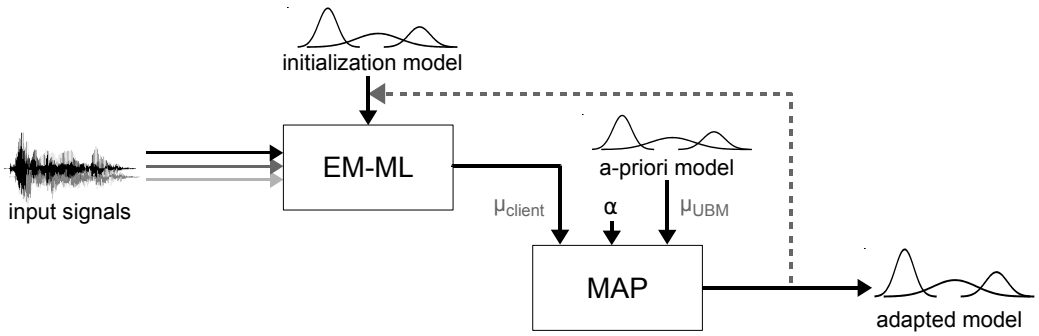


Figure 2.5: *Sketch of Maximum A Posteriori (MAP) adaptation, possibly iterative*

Being a blending of the UBM mean and the observed language statistics, the model resulting from a MAP adaptation lies, in terms of mean vectors, somewhere between the UBM and the language model that would be obtained through EM-ML. This fact is illustrated in Fig. 2.6 for one single Gaussian.

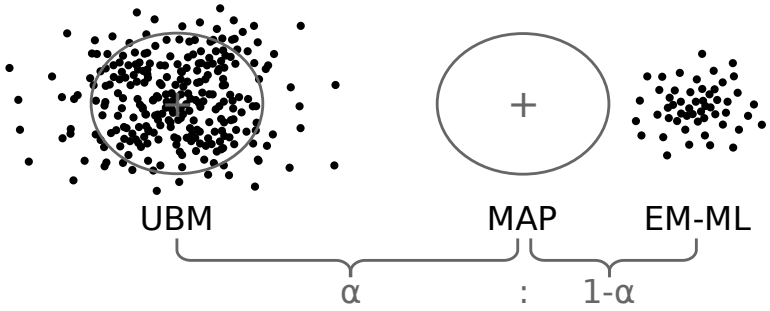


Figure 2.6: *Illustration of the MAP adapted model, laying between the UBM and the client data*

As the name indicates, the Maximum A Posteriori objective function maximizes

Maximum A Posteriori criterion

[19] Instead of using the initialization described in Sect. 2.4.2.1

the a posteriori distribution of the model parameters (Λ), based on the training feature vectors $\mathcal{X}$:

$$\hat{\Lambda} = \arg\max_{\Lambda} P(\Lambda|\mathcal{X}) \tag{2.14}$$

As described at the beginning of this chapter (Sect. 2.1), the a posteriori probability can not be obtained directly and has to be based on the conditional likelihood through Bayes' Theorem (cf. Eq. 2.2). Consequently the MAP implementation is also built on top of the (a priori) conditional likelihood $P(\mathcal{X}|\Lambda)$. Since this likelihood is also used by the ML criterion, the MAP adaptation can be based on ML Point-Estimates (or on the sufficient statistics leading to them). In fact, this situation is also depicted in Fig. 2.5.

— ◇ —

ML vs. MAP

In summary, the ML criterion operates in a non-Bayesian way on the Likelihood. Combined with the EM algorithm, it tries to estimate the best possible model parameters, based on the corresponding training data. It can thus be seen as a priori optimization or to focus on the training phase. On the other hand, MAP is a Bayesian method and is thus based on the a posteriori probability. Consequently, MAP adaptation is thought to meet the overall goal of the posterior probability (Eq. 2.2). It can thus be seen as trying to estimate a good model for the testing phase. Further explanations on these criteria and on optimization can be found in (Kamen and Su, 1999, Sect. 3.5, p. 94).

In addition, we can find parallels in the structure of the update formulas of the EM-ML and the MAP adaptation: They both use sufficient statistics/means and weights[20]. We will see in Sect. 3.4.1 that even Maximum Mutual Information (MMI) builds on the same structure.

2.5 Scoring and score processing

After having presented the universal Feature Extraction and the training phase, we now come to the testing phase. In this second phase, a speech utterance whose language we want to recognize is presented to the system. This returns a score in function of the utterance and the model of the hypothesized language. The score may be a Likelihood, as in the context of GMMs, or a distance measure (for instance for SVMs).

[20] *Excurse:* In EM-ML, we have $\frac{\vartheta^1}{\vartheta^0}$ (Eq. 2.11) for updating the mean. This is a fraction of the first order to the zeroth order statistics. It can equally be written as fraction with in its numerator a mean weighted by zeroth order statistics and in its denominator the zeroth order statistics (weighting) alone ($\mu = \frac{\vartheta^0\mu}{\vartheta^0} = \frac{\vartheta^0 \cdot \vartheta^1/\vartheta^0}{\vartheta^0} = \frac{\vartheta^1}{\vartheta^0}$). Here in MAP (Eq. 2.13), we have also a fraction dividing a sum of weighted means by the sum of their weights. In MMI (Sect. 3.4.1), we will see, that we will have three summands, laid out exactly in the same manner.

The scores of several test speech utterances are then processed in order to evaluate the system. LR systems are most often evaluated as a verification task (as opposed to the identification task, see Sect. 1.2.1), detecting if a candidate language is present in the input utterance or not. Since a good decade, this mode of evaluation is often chosen, because it matches the NIST Language Recognition Evaluations.

2.5.1 Scoring

Scoring is the step of using the model estimated in the training phase. This is typically done on utterances which were not implicated in the model estimation process (called the *testing dataset*). Each utterance is scored against each estimated model.

— ◇ —

For an input utterance and a candidate language, the LR system returns a score. For most system designs, this score can be considered being a log-likelihood, namely of the utterance being of that language. Consequently, the Log-Likelihood type scores have a negative value (since the Likelihood is comprised between 0 and 1) and the bigger (the closer to 0) it is, the more likely the language is present.

We can define the Log-Likelihood score of utterance $\mathcal{X}$ against the model of language l in a general manner:

$$LLk_l(\mathcal{X}) = \log P(\mathcal{X}|l) \tag{2.15}$$

Here, we make the assumption of frame independence (which is not fully true[21], but widely accepted). So this score can further be written in terms of frame probabilities: frame independence

$$LLk_l(\mathcal{X}) = \log \prod_{x \in \mathcal{X}} P(x|l)^{1/|\mathcal{X}|} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \log P(x|l) \tag{2.16}$$

The normalization term $\frac{1}{|\mathcal{X}|}$ is not really required since each test utterance is processed independently (and thus it cancels out in the decision function Eq. 2.2), but it allows to obtain scores that are more comparable between utterances[22].

In the particular case of GMM scoring, this LLk becomes:

$$LLk_l(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \log \left(\sum_{g \in G_l} \gamma_g(x) \right) \tag{2.17}$$

where G_l is the set of Gaussians of language l's GMM and $\gamma_g(x)$ is the Gaussian g's posterior probability of frame x, defined in Eq. 2.7. The whole also corresponds to the EM optimization criterion (Eq. 2.10).

[21] The window shift being smaller than the window size, they overlap and thus contain partly the same information. In addition, the changes in the speech signal are continuous. In most cases, it does not contain abrupt changes on the level of the frame duration.

[22] They are still slightly biased due to the frame independence assumption.

2.5.2 Score normalization

Similarly as on the FE and on the modeling level, normalizations can be and have to be performed also on this score level. Two fundamentally different types of score normalization have to be distinguished: inter-utterance and inter-language normalization.

— ◇ —

utterance normalization

inter-utterance normalization: The objective of the first kind of normalization is to compensate for differences between single utterances, which are reflected in the scores. This is by far the most crucial kind of normalization for any verification task. In the identification case, this is not a problem, since there is no global threshold and the final decision can be drawn on the set of trials of each utterance separately and independently. In verification however, we have (usually a global) threshold. So the scores obtained by different utterances have to be comparable. This kind of normalization is often conducted on a per utterance basis. The two main normalization approaches are: (i) The utterance is used to compute some reference score (the World Model approach, Sect. 3.5.1). (ii) The normalization is carried out using the set of scores of one utterance's trials (hence the scores of the utterance tested on all languages in turn). No normalization factors have to be estimated beforehand and no specially allocated data (development corpus) is required for this.

language normalization

inter-language normalization (or *calibration*): The objective of the second kind of normalization is to balance the scores produced by the different language recognizers among each other. A particular language recognizer (a language model in our case) may have the tendency to produce better (or poorer) scores than others. this recognizer's scores would be over- (or under-) rated and will cause an exceeding number of false positives (or misses, see Sect. 2.6.2 for these notions). This problem applies to both, the identification and the verification task. In order to apply this kind of normalization, some separately estimated normalization parameters are required.

We may explained this in a more visual manner: Let all results be organized in a matrix where each row represents an utterance and the columns the different languages (recognizers/models). The first type of normalization operates (individually) on the different rows. And the second one tries to normalize the columns among themselves. The two kinds of normalizations operate thus along the two axes of this matrix.

— ◇ —

In the baseline system, we use only a rather basic inter-utterance score normalization, which will be presented in the next two sections. More advanced score processing strategies (Sect. 3.5) may also combine both kinds of normalizations into one run.

2.5.2.1 divSum

As (Campbell et al., 2006a) state, "*The performance of language recognition is enhanced considerably by applying back-end processing to the target language scores. A simple back-end process is to apply a log-likelihood normalization*". This Log-Likelihood normalization will be described here.

— ◇ —

The most simple approach to utterance-wise score normalization consists in dividing each score s_l in turn by the sum over the scores the utterance ($\mathcal{X}$) obtained against all language models:

$$\widehat{s}_l^*(\mathcal{X}) = \frac{s_l(\mathcal{X})}{\sum_{k \in L} s_k(\mathcal{X})} \tag{2.18}$$

l being the hypothesized language and k being each language of our set in turn. Let's note that here, we include the current score in the denominator sum (as opposed to the *divOthers* variant described below). This division is performed in Likelihood domain. If the scores are of Log-Likelihood kind (which is the case if it is an average Log-Likelihood, as in Eq. 2.16 or Eq. 2.17), they first have to be transformed to the non-logarithmic domain (and at the end back to log again). So the normalization becomes:

$$\widehat{s}_l(\mathcal{X}) = \log\left(\frac{e^{LLk_l(\mathcal{X})}}{\sum_{k \in L} e^{LLk_k(\mathcal{X})}}\right) = LLk_l(\mathcal{X}) - \log \sum_{k \in L} e^{LLk_k(\mathcal{X})} \tag{2.19}$$

where $LLk_l(\mathcal{X})$ is the (average) Log-Likelihood of utterance $\mathcal{X}$, given the hypothesized language l and where L is the set of all languages. The second term (log-sum) has to be calculated only once for a given utterance, since it does not depend on l.

We will call the normalization presented in this section *divSum*. This type of normalization (and its various forms below) is largely used throughout literature (Campbell et al., 2006a; Castaldo et al., 2007b; BenZeghiba et al., 2009). Often, it is combined with additional score processing steps, as will be shown in Sect. 3.5.2. divSum

Different alternatives of the *divSum* normalization may be found in literature: variants

divOthers
A frequently used variant consists in leaving out the current language l from the denominator. In consequence, the denominator sum range reads $\sum_{k \neq l}^{L}$. This kind has also been trialed in our works and produces substantially similar results. But the advantage of *divSum* is that the denominator is the same for all l in the context of a given $\mathcal{X}$. Thus it has to be calculated only once per utterance, which accounts for some speedup of the scoring step. Also, *divSum* corresponds more to the World Model normalization approach used in SR, where the denominator also only depends on $\mathcal{X}$ (see also Sect. 3.5.1 for this).

divMean
In the denominator, we may also calculate an average over the diverse k (whether including or excluding l), e.g.: $\frac{1}{|L|}\sum_{k\in L}$. This variant is also frequently used.

— ◇ —

Some form of inter-utterance normalization (as the presented ones) is indispensable for an evaluation in detection or verification mode. This is also shown by our baseline system (as well as the subsequent ones), which obtains a useless performance of around 50% errors when it is run without score normalization and a more usable results using this kind of normalization.

In fact, for the baseline system, an other derived version is used and the technique is presented in Sect. 2.5.2.2 just hereafter. Finally, simpler but equally powerful novel steps introduced in our work will be presented in Sect. 7.2.

T-norm

Opinion: Some authors (Castaldo et al., 2007b,c,a) call the type of normalizations presented in this section *T-normalization* (Tnorm). In the context of Speaker Recognition, Tnorm uses a cohort of independent speakers (under the form of models) for the normalization (Auckenthaler et al., 2000; Campbell et al., 2004). In LR, this cohort is built up of the other language models, as discussed above. When the denominator is the average over all languages (thus including l), this comes very close to the Tnorm.
But (Castaldo et al., 2007b) introduce an additional normalization term which does not correspond to any enumerated variant and which thus is difficult to be retraced: $\widehat{s}_l = log\Big(\frac{1}{|L|-1}\cdot\frac{e^{LLk_l}}{\sum_{k\neq l}e^{LLk_k}}\Big)$.

2.5.2.2 divSum with exponent K

Based on the *divSum* normalization described in previous section, system performance can be enhanced by powering each score with a constant K (or multiplying the Log-Likelihood domain score by K) in the following manner:

$$\widehat{s}^*_l(\mathcal{X}) = \frac{s_l(\mathcal{X})^K}{\sum_{k\in L} s_k(\mathcal{X})^K} \tag{2.20}$$

$$\widehat{s}_l(\mathcal{X}) = LLk_l(\mathcal{X})K - \log\sum_{k\in L} e^{LLk_k(\mathcal{X})K} \tag{2.21}$$

l being the hypothesized language and k being each language in turn. In our case, a K of 35 has been chosen. This is based on consequent observations of the impact of different Ks (in steps of 5) prior to our participation to the LRE 2009.

This variant yields considerably better performances compared to the *divSum* approach[23]. On the other hand, the value of the constant K, here set to 35, is not the best one for every experiment. It should typically be determined on a per-case basis using held back development data.

This factor has been introduced by (Matějka et al., 2006) in the context of MMI training. They describe that this procedure attempts to introduce some correction to the assumption of the frames being independent of each other (Sect. 2.5.1). While they use an empirically determined value of 6, its maximum is limited to the number of frames in the current utterance[24].

(Matějka et al., 2006) use this empirical factor K only in the MMI estimation process, but not during scoring. However, we propose here to use it expressively during the scoring (or more accurately during the score processing) step. Applying this technique during score processing in order to enhance score normalization constitutes a novelty.

novelty

— ◇ —

Fig. 2.7 shows typical[25] distributions (histograms) of scores applying different normalizations discussed in the previous sections. In each plot, the distributions of the target and of the non-target trials are shown. Plot (a) uses bare LLk scores, which can be seen at the score range covered (around -72). The two distributions completely overlap — as described, the system is useless without normalization. In (b) *divSum* normalized scores are used. This can also be seen as using a factor K of 1. This normalizes the scores between the different utterances (inter-utterance normalization). Since the target trials tend to have a better score than the non-targets, they get better "aligned" and start to detach from the non-targets. In (c) *divSum* normalization is applied to the scores with a K factor of 35. And in (d) a K factor of 70 is used.

effects of K

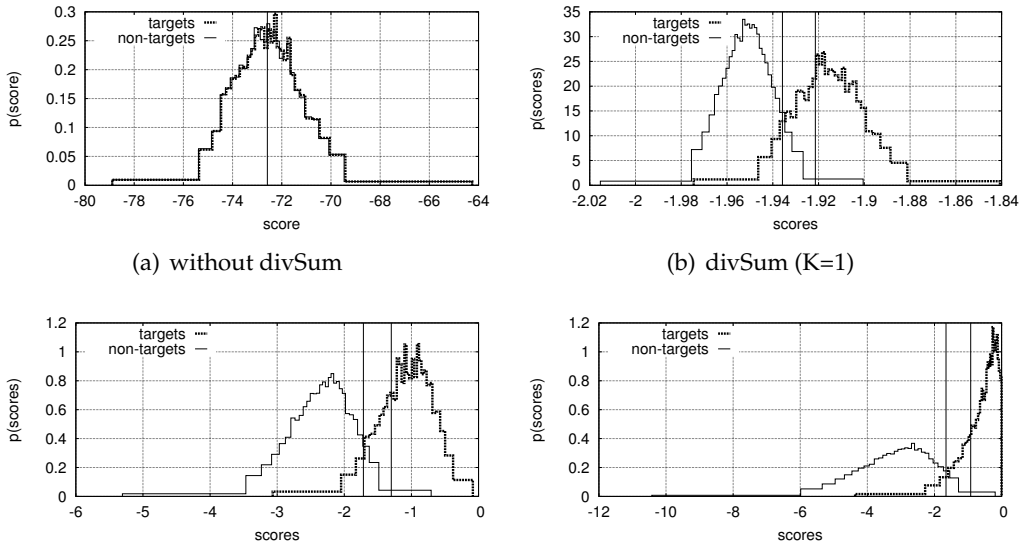
Opinion: The K factor seems to have the effect of compressing high scores towards the 0 limit and to dilate smaller scores exponentially towards negative infinity. This is translated in the distributions as grouping or emphasizing of the target trials (which should have better scores) and to flattening out the non-target distribution towards the negative. So the error rate or cost, which is linked to the overlap of both distributions, has the tendency to decrease up to a certain point.

Such a reasonably straightforward normalization technique already proves to enhance LR performance. More advanced score normalizations like inter-language normalizations are presented later on in Sect. 3.5.2.

[23] Up to -42% relative for the JFA system with 2048 Gaussians.

[24] In fact, our parameter K corresponds to (Matějka et al., 2006)'s parameter C of the full exponent $K_r = \frac{C}{T_r}$ since we included the number of frames (T_r) already in Eq. 2.16.They state $0 < Kr < 1$.

[25] The scores were obtained by the 256 Gaussian system with accurate JFA, LRE 2005 protocol.



(a) without divSum

(b) divSum (K=1)

(c) divSum K=35

(d) divSum K=70

Figure 2.7: *Sample score distributions for raw LLk and divSum normalized scores with factors 1, 35 and 70 for our JFA system*

2.6 Evaluation

The evaluation of a Language Recognition (LR) system involves applying the two phases of training and testing. This is done using previously defined data sets and will produce a set of scores, which will then be analyzed to give some performance measure.

The resulting performance of a LR system not only depends on the strategies chosen at the different levels or processing steps, but also on the data used for its calculation — as well for the training as for the testing stage. This section gives a quick hint about the different available datasets and on how the performance of a system is measured.

2.6.1 Data sources

In order to train models of an automatic speech processing system and later on to evaluate it and to measure its performance in a given environment, we need well defined protocols (Sect. 2.6.3), which also indicate the different data sets. The data sets used for training and for testing have to be disjointed. This section gives an overview of the different available corpora and describes the different parts thereof which have been used in the experiments leading to the results presented in this document.

— ◇ —

Table 2.1 gives an overview of available speech corpora for Language Recognition, distributed by NIST[26], LDC[27] or the CSLU[28], together with their respective identification numbers. These indications are given here since sometimes, it is hard to find out which data is included in an other corpus as sub-part. And also, the same dataset is often referenced under different identifiers. Most of the datasets in this table were used in one or the other way in setups presented in this work. The remaining part of this section describes which datasets were used in which step.

Table 2.1: *Some of the available corpora with their corresponding identification numbers*

Corpus	LDC-ID	NIST-ID	ISBN	is part of	notes
CallFriend	LDC1996S46– LDC1996S60		1-58563-061-6– 1-58563-075-6		
VOA 2				LDC2009E40	
VOA 3				LDC2009E40	
MLTS	LDC2006S35				
lid96d1	LDC2006E103	+r31		LDC2006S31	
lid96e1	LDC2006E102	+r32		LDC2006S31	
lid03e1	LDC2006E107			LDC2006S31	
LDC2006S31			1-58563-364-X	LDC2009E41	should be used as LRE-09 CTS training data
lid05d1	LDC2006E104	R103		LDC2008S05 & LDC2009R31(?)	Indian-English only
lid05e1	LDC2006E105	R104-1.1		LDC2008S05	
LDC2008S05			1-58563-477-8	LDC2009E41	
LDC2009E41					lre09-CTS-train[12] ??
lre05e-full					
lre07d-suppl	LDC2009S05		1-58563-530-9		
lid07e1	LDC2009S04	R117_1_1	1-58563-529-4		extracts of lid07e-full
lid07e-full	LDC2009R31				LRE-09 Supplem' CTS Training
LRE09e1		R124_1_1 & R124_2_1			

On the other hand, Table 2.2[29] gives a list of all languages (and dialects) made available by the corpora listed in Table 2.1. This table here also shows the number of utterances for each language contained in a given data set. No such indication is given for the VOA corpus since the notion of utterance can not really be defined.

[26] National Institute of Standards and Technology

[27] Linguistic Data Consortium, at University of Pennsylvania, http://www.ldc.upenn.edu/

[28] Center for Spoken Language Understanding, of the OHSU

[29] For layout reason, this table was split into two parts: Table 2.2 and Table 2.3.

In most LR corpora, note that generally every utterance is produced by a different speaker. To the exceptions count some evaluation datasets where the 3 and 10 second segments are cut out of the longer 30 *s* ones. Also, there is no notion of speaker for the VOA datasets and the same speaker may appear at several locations in the database[36]. Consequently, the number of utterances given in Table 2.2 reflects at the same time the number of speakers, with the named limits.

2.6.1.1 Data sets used for training

Training material is drawn from various sources. Let us define the different data sets as follows:

cfChans
The *CallFriend* corpus, which is distributed by LDC (see reference (CallFriend, 2010)), is a very well adapted corpus for training LR systems. It has the same amount of data for each of totally 15 languages and dialects. It is organized into three equal-sized parts: *train*, *devtest* and *evltest*. Each of these three parts contains 20 complete two-ended, half-hour conversations per language (the two channels have first to be separated).

Our *cfChans* data set is composed of all data contained in the *train* part of CallFriend for seven languages[37]. This gives 40 files, each containing roughly 14 minutes of speech, for every language or dialect.

cfChansTrDvEv
The *cfChansTrDvEv* data set is similar to *cfChans*, but includes data of all three CallFriend parts for the same languages. This set has thus a total of 1200 files.

CTSsmall
We will detail in due curse (Sect. 6.1) that the NIST LRE 2009 setup includes data from two different channel categories, namely the classical Conversational Telephone Speech (CTS) and the major part coming from telephone bandwidth parts of Voice of America (VOA) radio broadcasts.

The *CTSsmall* data set is composed of the following corpora, providing data for the CTS condition:

[30] (Table 2.2) Bangla and Bengali are two names for the same language
[31] (Table 2.2) English from "Talk to America" emissions, without further precision
[32] (Table 2.2) Persian and Farsi are actually two names for the same language
[33] (Table 2.2) For VOA3, it is African French (West and Central-)
[34] (Table 2.3) Kirundi/Kinyarwanda
[35] (Table 2.3) For VOA3, it is Caribbean, Latin American, and Andean (andina) Spanish
[36] Some NIST LRE 2009 participants could confirm this fact due to recognizing the voice of a well known person of their region.
[37] The seven languages are: English*, Hindi, Japanese, Korean, Mandarin*, Spanish*, Tamil, including two dialects for the languages followed by a star (*).

- In a similar manner as *cfChansTrDvEv*, data from all three CallFriend parts is included for 8 languages[38]
- The 120 Indian English recordings of NIST's LRE 2005 development data (lid05d1).
- The full conversations of the LRE 2007 evaluation data (lid07e-full) for 9 languages[39].

In this dataset, each language has between 40 and 317 segments, representing between 2.7 and 58.6 hours of speech. In total for 11 different languages, we have 312 h in 1867 segments.

CTS
This *CTS* set includes *CTSsmall*, augmented by the 10 and 30 second evaluation segments (ranging from 284 to 1934 segments per language) of LRE 2005 (lid05e1) for 6 languages[40]. For this set, the 3 second Indian English segments of LRE 2005 development (included in *CTSsmall*) have been avoided, since we added other utterances of this language. Each language has segment counts ranging from 40 to 2253, that represent between 2.7 and 64.8 hours of speech. In total for 11 different languages, we have 337 h in 7870 segments.

VOA
The data of the *VOA* set comes from the Voice of America 3 (voa3) corpus and is used together with the *phone/wideband* and *speech/non-speech* segmentation provided by NIST for the LRE 2009 campaign and which were built by the BUT[41] lab (Plchot et al., 2009). Each language is represented by 3.0 to 27.9 hours of speech. In total for 22 languages, we have 333 h across 8632 segments.

VOA10k
This *VOA10k* set contains all data of the *VOA* set and a lot of additional voa3 data for a total of 141 599 segments (1111 to a maximum of 11 029 for each language).

2.6.1.2 Testing data sets

We will evaluate the systems under two protocols (described in Sect. 2.6.3). For this reason, we use two different testing data sets:

LRE 2005
On one side, tests are conducted on NIST LRE 2005 data (NIST LRE, 2005). We name this set *lid05e1*. This evaluation set comprises 10 986 utterances, each containing 3, 10 or 30 seconds of speech.

[38] English, Farsi (Persian), French, Hindi, Korean, Mandarin, Spanish and Vietnamese, again with both available dialects for the three concerned languages.
[39] Cantonese, English, Indian English, Korean, Mandarin, Persian (Farsi), Russian, Spanish and Vietnamese.
[40] English, Hindi, Indian English, Korean, Mandarin and Spanish.
[41] Brno University of Technology

The primary condition of NIST LRE 2005 aggregates just utterances of seven languages (closed-set condition). The seven languages are: English, Hindi, Japanese, Korean, Mandarin, Spanish, Tamil. The closed-set condition has a total of 10 734 utterances. We focus mainly on the 30 second ones, which comes down to 3578 files, giving as many target trials and thus 21 468 non-target trials (one for each non-target language and utterance).

LRE 2009

In the other case, *LRE09e1*, tests are conducted on NIST LRE 2009 data (NIST LRE, 2009). That evaluation set comes with 41 794 utterances also containing nominally 3, 10 and 30 seconds of speech each. Our SAD processing (Sect. 2.3.2) did not detect any speech in 106 of these files[42]. The other sum up to 133.3 hours of speech.

The primary condition aggregates 31 178 utterances of a closed-set of 23 languages: Amharic, Bosnian, Cantonese, Creole (Haitian), Croatian, Dari, English (American), French, Georgian, Hausa, Hindi, Indian English, Korean, Mandarin, Pashto, Farsi (Persian), Portuguese, Russian, Spanish, Turkish, Ukrainian, Urdu and Vietnamese. We focus on the 30 second ones, which are at a number of 10 571. This gives that many target trials and 232 562 non-target trials. There are between 315 and 1015 testing files per language.

As stated, this test set comprises data drawn from CTS and from VOA sources (NIST LRE, 2009; Greenberg and Martin, 2009). There are 8708 testing files in 10 languages[43] originating from CTS sources. Thereof 3081 for the 30 second condition with 32 to 625 for each language. Drawn from VOA are 22 470 testing files with 7490 of 30 seconds. We count between 27 and 399 testing utterances for each of 22 of the 23 languages[44].

2.6.2 Performance metric

Having obtained a series of scores (Sect. 2.5) for a set of speech utterance (Sect. 2.6.1) of a priori unknown languages by running tests against the set of previously estimated language models (Sect. 2.4), we can evaluate the system by calculating a performance.

— ◇ —

In our works, we evaluate the systems in detection mode (Sect. 1.2.1). So the questions we try to answer are of the kind "*Is language l present in this utterance?*". This detection mode is used since it is prescribed by the NIST Language Recognition Evaluations, which became de facto standard over the last decade.

[42] A quick glance at some of these files reveals the presence of huge noise or files containing just music.

[43] Cantonese, English, Hindi, Indian English, Korean, Mandarin, Persian, Russian, Urdu and Vietnamese

[44] The missing language being Indian English

A system's performance is analyzed at a certain operating point, being a specific application of the system with given priors and which returns a hard "*Yes/No*" detection decision for each trial. For this, the trial score is compared to a given threshold. If the score is below the threshold, the answer is "*No*" and, according to the speaker verification terminology, we call it a *rejection*. If it is the same or more positive than the threshold, the answer is "*Yes*" and we call it an *acceptance*.

system decision
reject
accept

The evaluation now checks if the answer is correct or not, by accessing the *key* that holds the true answer which test file is of which language.

We define as *target trial* a test where a file of true language l has been tested on model l also. All the tests where the file is run against a model which does not correspond to its true language are called *non-target trials*.

target
non-target

Depending on the check against the answer key, there are two types of errors:

- If the answer was positive (a "*Yes*"), but the utterance does not contain that language, we speak of *false positive* (or false acceptance, or in an even more historic terminology: false alert). This is thus an utterance in which a certain language has mistakenly been detected.

false positives

- If the answer was negative, but the language is effectively present in the utterance, we call it *false negative* or a miss. This is thus an utterance that is not recognized of being of the true language.

false negatives

In summary, *false positives* are non-target trials with a "*Yes*" answer and *false negatives* are target trials with a "*No*" decision.

— ◇ —

By changing the threshold, the number of errors of one type decreases while the ones of the other type increase[45]. In the following, some performance measures are presented and discussed. They build on the rates (percentages, or expressed as probabilities) of these two types of errors.

As explained, the rate configuration of the two error types depend on the threshold. So these two rates may be plotted against the threshold. For this, each type of errors is put to one coordinate and with each different threshold, a single dot is added to the plot. This plotting of the system performance at all possible thresholds is called the Detection Error Trade-off (DET) curve (Martin et al., 1997).

2.6.2.1 Equal Error Rate

The Equal Error Rate (EER) is a metric concept that is rather simple to understand. It is the performance at the system's operation point of equal error rates. That is the system choosing the (global) threshold in such a way to perform equally on the two types of errors. In a graphical interpretation, it is the point where the DET curve

[45] In some limit cases, the number of one type may remain constant over a certain threshold interval

crosses the first quadrant's diagonal. The diagonal being all the points where the values for the two kinds of errors are the same.

Pooled Equal Error Rate
The easiest way to compute an EER from a set of (target and non-target) trials is to put the trials of all languages together and keep only the information if they are target or non-target trials (and obviously the answer or the error checking). For this, usually, the target trials are put on one side and the non-target ones on the other. Then the threshold is slid along the score space until the percentage of targets scoring below the threshold is the same as the percentage of too high non-targets. This simplistic approach has a big drawback: If the number of tests for the different (target) languages are unbalanced, the yielded results may be quite biased.

Average Equal Error Rate
As first solution, we could calculate an EER for each language separately and then take their average. This approach avoids big bias due to unbalanced test counts, but does not use a global threshold since it is determined on a per language basis.

These drawbacks motivate the use of even an other kind of performance measure, namely minimal average cost (presented hereafter) rather than EER, because the minimal average cost measure is insensitive to unbalanced test sets and features nevertheless a global threshold.

2.6.2.2 Minimal average cost

System performance can also be measured using *minimal average cost* (*min-*C_{avg}), which avoids major drawbacks of EER measures. The detection system under *min-*C_{avg} evaluation chooses the decision threshold (or operation point) in a particular way: It minimizes the average expected cost of misses[46] and false acceptances[47] among all pairs of languages. The cost measure is related to the error rates by including priors and possibly unbalanced penalties (decision costs). An extended description may be found in Section 4.1f of the LRE 2009 plan (NIST LRE, 2009).

— ◇ —

complete closed-set case

In all applications and protocols studied in this work, a false negative (a miss) and a false positive decision (false acceptance) have the same cost and the prior of a target trial is 0.5, independently of the number of languages. The cost function to be minimized is thus:

$$C_{avg} = \frac{1}{|L|} \sum_{l \in L} \Big[\, 0.5 \cdot P_{Miss}(l) \; + \; \frac{0.5}{|L|-1} \sum_{k \neq l \in L} P_{FA}(l,k) \, \Big] \tag{2.22}$$

$P_{Miss}()$ is the probability that a language model misses a match (false negative) and $P_{FA}(l,k)$ is the probability that an utterance of language l is mistakenly recognized

[46] Utterances not recognized as being of the true language
[47] Mistakenly detecting the presence of a language

as being of language k (false acceptance). These probabilities are calculated in function of a global threshold. This average detection cost is thus the mean over all target languages of its probability to be missed and its average probability to be detected by a false language model[48].

In the case where the set of classes (in our case languages) present in the testing utterances does not cover the whole set of language models we have, this cost function turns to the more general one: incomplete closed-set case

$$C_{avg} = \frac{1}{|L_T|} \sum_{l \in L_T} \left[0.5 \cdot P_{Miss}(l) \; + \; \frac{0.5}{|L_M|\text{-}1} \sum_{k \neq l \in L_M} P_{FA}(l,k) \right] \tag{2.23}$$

where L_T is the set of languages in the test data set (also called *target languages*) and L_M is the set of languages for which we have models (*non-target languages*). This more elaborated cost function has to be used if we have to run an evaluation in a context where the there is at least one language (of those we trained models for) that has no test utterances assigned. This will be the case in Chapter 6 and Chapter 7 when we analyze CTS and VOA data separately.. In a general manner, $|L_T| \leq |L_M|$. Otherwise we would speak of an *open-set* evaluation, which is not in the scope of this work.

This C_{avg} function is minimized with respect to the threshold θ which is used for calculating the error probabilities (or rates).

$$min\text{-}C_{avg} = \min_{\theta} \; {C_{avg}}^{\theta} \tag{2.24}$$

The two kinds of error probabilities are defined for a given threshold θ as follows:

$P_{Miss}(l,\theta)$ is the fraction of target tests T of language l with scores $s < \theta$: error probabilities

$$P_{Miss}(l,\theta) = \frac{\sum_{t \in T(l)} \overline{\mathbf{1}_\theta(s_t)}}{|T(l)|} \quad ; \quad \overline{\mathbf{1}_\theta(s)} = \begin{cases} 1 & \text{if } s < \theta, \\ 0 & \text{if } s \geq \theta. \end{cases} \tag{2.25}$$

Similarly, $P_{FA}(l,k,\theta)$ is the fraction of non-target utterances N of true language l tested against model k which have a score $s \geq \theta$:

$$P_{FA}(l,k,\theta) = \frac{\sum_{t \in N(l,k)} \mathbf{1}_\theta(s_t)}{|N(l,k)|} \quad ; \quad \mathbf{1}_\theta(s) = \begin{cases} 0 & \text{if } s < \theta, \\ 1 & \text{if } s \geq \theta. \end{cases} \tag{2.26}$$

— ◇ —

In this work, we stick to the scheme of expressing the $min\text{-}C_{avg}$ measure under the form of percentages (%). In literature it can also be found under the $\times 100$ form (as is done for *min-DCF*). Our choice is motivated by the similar interpretation of the $min\text{-}C_{avg}$ to the EER, where a useless (random) system has a cost of 50%.

[48] C_{avg} is the average of the detection cost (the content of the square brackets of Eq. 2.22, itself usually denoted C_{DET}) over all languages.

An other cost based performance metric introduced by (Brümmer and du Preez, 2006; Brümmer and van Leeuwen, 2006) is the Log-Likelihood Ratio (average) cost (C_{llr}), which will be discussed in Sect. 3.6, since we did not apply it to our works.

2.6.3 Protocols

It is necessary to be able to evaluate a system in a well-defined and consistent environment[49]. One reason is to be able to reproduce the same results using a given system. And also to allow comparisons between systems with slightly different setups. If possible, we should also work on the same basis as other research groups to be able to compare system performances meaningfully. These are the main goals of experimental protocols. Further, they clearly describe the experimental environment such as the application area of the system, the data to be used, the way to measure the performance and so on.

In our works, we use two well-established protocols designed by NIST for their Language Recognition Evaluations (LREs). In a first step, our works were analyzed in the context of 2005's LRE protocol. Amongst others it was the most recent LRE (testing) data available for developing our first systems, since the LRE 2007 data was released only in June 2008. In a later stage, for our participation to the NIST LRE 2009 campaign, we faced the 2009 data with the additional VOA training corpora.

The official NIST LRE protocols do not set constraints on which training data may be used — they simply have to be listed on the system description (see i.e.(Verdet et al., 2009a) and the following subsections). These protocols define different tasks. E.g. there are testing utterances with a content of nominally 3, 10 and 30 seconds of speech. And the evaluation may be carried out in an open-set or a closed-set context. Here, we focus on the core task, which uses only the 30 s utterances, closed-set.

2.6.3.1 NIST LRE 2005

The NIST Language Recognition Evaluation 2005 (NIST LRE, 2005) consists in detecting the following seven languages: English, Hindi, Japanese, Korean, Mandarin, Spanish and Tamil.

For training the UBM, as well as language models (and later on the JFA **U** matrix), the *cfChansTrDvEv* (Sect. 2.6.1) data set is used. Some single analyses were carried out using only the *cfChans* data set. The testing data is the official one that comes from *lid05e1*. In Table 2.2, this data set is highlighted and the number of testing utterances is given (including utterances of all lengths). Performances on NIST LRE 2005 are usually evaluated using the pooled EER metric (Sect. 2.6.2.1).

[49] This has not much to do with the stability of the recording environment. So we are not trying to avoid the robustness problem outlined in Sect. 1.1, but we aim stability of the evaluation environment in the sense of reproducibility.

2.6.3.2 NIST LRE 2009

The NIST Language Recognition Evaluation 2009 (NIST LRE, 2009) consists in detecting the following twenty-two languages: Amharic, Bosnian, Cantonese, Creole (Haitian), Croatian, Dari, (American) English, (West- and Central-African) French, Georgian, Hausa, Hindi, Indian English, Korean, Mandarin Chinese, Pashto, Farsi (which is the same as Persian), (Angolan) Portuguese, Russian, (Caribbean, Latin American and Andean) Spanish, Turkish, Ukrainian, Urdu and Vietnamese.

The UBM was trained with all data of *CTSsmall*[50] and *VOA10k* sets (including thus six times more VOA than CTS data). This comes down to 787 million speech frames (representing 2185 hours in 143 366 utterances). Language models (and JFA's **U** matrix) are estimated using *CTS* and *VOA* data sets. The testing data is the official one that comes from *LRE09e1*. The exact number of testing segments in this set is indicated in Table 2.2. This table also indicates the 16 out-of-set languages (with a star), as well as which languages can be found on the VOA data disks. From 2005 to 2009, NIST LRE changed performance metric from pooled EER to minimal average cost (*min-*C_{avg}, Sect. 2.6.2.2), which will generally be used when measuring system's performance on *LRE09e1* data.

2.7 System implementation

In the following, we give a rapid insight how the baseline system and the subsequent extensions were implemented and which tools were used for this task.

The Feature Extraction step is carried out using the SPRO4 tool (SPro, 2004) where cepstral coefficients needed to be extracted. For PLP based features and those using RaSta filtering, the in-house tool `lia_plp_mt` was used.

The Language Recognition systems of all reported experiments were implemented using the free software library and framework ALIZE (MISTRAL, 2009; Bonastre et al., 2008, 2005; Charton et al., 2010)[51]. They have mainly been developed at the LABORATOIRE INFORMATIQUE D'AVIGNON (LIA) . For working with SVMs, the utils are based on the third party library LIBSVM (Chang and Lin, 2001; Hsu et al., 2003). All these toolkits are freely available to the community.

All this was assembled with a good part of Bash, Ruby[52] and Perl scripting to configure and to glue together the different processing steps. And finally, the evaluation protocols are the ones of NIST LRE 2005 and 2009, as described in previous section.

[50] All data sets are described in Sect. 2.6.1

[51] The high level tools are sometimes also referenced under the MISTRAL term.

[52] Featuring a framework of coherent classes for different file types and extended algebraic calculations, as well as a set of tools for conversions, analyzes and other utilities.

Some more technical aspects which had to be handled comprise ways to cope with the huge amount of data. This ranges from massively parallelized computation and algorithm redesign (i.e. fully parallelized **U** estimation) to loading only one chunk of a big matrix (i.e. statistics) at any given time since the whole would not fit into memory.

— ◇ —

This concludes the chapter which presented the cycle of a pattern recognition or classification application. We have shown a Language Recognition system with all its processing steps. They are organized into the training and the testing phases. The major steps involved in the training phase are Feature Extraction and model estimation, and for the testing phase: Feature Extraction, (Likelihood) scoring, score processing, followed by the final evaluation with performance measurement. Having stated protocols and associated data sets, we now have also at our disposal test beds to evaluate different systems.

In the context of the present works, we have chosen the following system setup: We use Mel scale SDC features in a 7-1-3-7 configuration, energy based Speech activity detection and mean/variance feature normalization. Our modeling approaches are based on a GMM-UBM setup using MAP adaptation. For the NIST LRE 2005 protocol, scores will generally undergo a divSum-K normalization before being evaluated using pooled EER. However for the NIST LRE 2009 protocol, the scores are normalized by the llkMax0 method (which will be introduced in Sect. 7.2.1). The performance measurement is also changed to $min\text{-}C_{avg}$ in order to avoid test set imbalance bias.

Table 2.2: *List of available languages across the diverse corpora. Number of files, star indicates out-of-set. See Sect. 2.6.1 for footnote texts. Part 1 of 2, continuation in Table 2.3.*

language	**LRE09e1**	voa3	CallFriend	OGI-MLTS	OGI 22-L	lid96d1	lid96e1	lid03e1	lid05d1	**lid05e1**	lid07dsuppl	lid07e1	lid07e-full
Albanian		X											
Amharic	1194	X											
Arabic	* 561	*X		X							240		
– Egyptian			40			226	240	240			40		
Azerbaijani	*1098	X											
Bangla/Bengali[30]	* 123	X									40	240	40
Belorussian	*1089	*X											
Bosnian	1065	X											
Bulgarian	*1125	*X											
Burmese		X											
Chinese													
– Cantonese	1071	X			X						40	240	40
– Mandarin	2976	X		X	X	471		240					
– – Mainland/north			40				234			1062		240	40
– – Taiwan			40				234			1800	168	252	42
– Min (Nan/south)	* 139										40	240	40
– Wu (Shanghai)	* 194										40	240	40
Creole (Haitian)	969	X											
Croatian	1128	X											
Czech					X								
Dari	1167	X											
English		X		X	X	477							
– American	2615							720		2445		246	41
– Non-Southern			40				1199						
– Southern			40				237						
– TtAm[31]		!X											
– Indian	1624								X	525		504	44
Farsi/Persian[32]	1159	X	40	X	X	237	240	240				240	40
French				X	X							*240	
– African[33]	1185	X											
– Canadian			40			232	240	240					
Georgian	1197	2X											
German			40	X	X	238	240	240		*252		240	
Greek		X											
Hausa	1167	X											
Hindustani													
– Hindi	1922	X	40	X	X	233	232	240		429		486	41
– Urdu	1133	X									40	246	41

Table 2.3: *List of available languages across the diverse corpora. Part 2 of 2, continuation of Table 2.2*

language	**LRE09e1**	voa3	CallFriend	OGI-MLTS	OGI 22-L	lid96d1	lid96e1	lid03e1	lid05d1	**lid05e1**	lid07dsuppl	lid07e1	lid07e-full
Hungarian					X								
Indonesian		X			X							*240	
Italian	* 80				X							*240	*40
Japanese	* 490		40	X	X	235	239	480		1095		240	40
Khmer		X											
Kirundi[34]		X											
Korean	1365	X	40	X	X	234	236	240		942		240	40
Kurdish		X											
Laotian		X											
Macedonian		X											
Ndebele		X											
Oromo, Afan-		X											
Pashto	1185	X											
Polish					X								
Portuguese					X								
– African (Angolan)	1191	X											
Punjabi	* 22											* 96	*16
Romanian	*1200	*X											
Russian	1486	X			X			*240			40	480	40
Serbian		X											
Shona		X											
Somali		X											
Spanish[35]	1155	X		X	X	456							
– Caribbean			40				234					240	40
– Non-C′/Highland			40				231					480	40
– Latin Am′								240					
– Mexican										1833	130		
Swahili	*1188	X			X								
Swedish					X								
Tagalog	* 224											*240	*40
Tamil			40	X	X	226	222	240		549	92	480	20
Thai	* 511	X									40	240	40
Tibetan	*1104	X											
Tigrigna		X											
Turkish	1182	X											
Ukrainian	1164	X											
Uzbek	*1146	X											
Vietnamese	878	X	40	X	X	228	239	240				480	40

Uguali nella varietà e variegati nell'unità,
unici nella diversità e diversi nella loro atta coadunazione
— Umberto ECO, Il nome della rosa, 1980 [0]

Chapter 3

State of the Art

Contents

This chapter is structured into two parts. The first one gives a chronological or historical overview of the introduction of different techniques. It first discusses the domains of speech and speaker recognition, then Language Recognition (LR) more in particular. This way of presentation has been chosen, since most methods used or tried on the LR problem were first employed in these other automatic speech processing domains.

The second part shows in a little more detail some important advances in Language Recognition during last decades. It is deliberately kept non-exhaustive. These important advances are structured following the system processing chain, as presented in the previous chapter to be able to situate them easily.

— ◇ —

linguistic point of view

The basic structure of state of the art systems do not differ much from those used ten years ago. Thus, Zissman's article (Zissman, 1996) still gives a good overview of the underlying approaches. What research did since then, was trying slightly different algorithms, playing with the tuning of those many parameters, fuse several systems to increase the overall performance and primarily finding ways to cope with special kinds of cases — as the problem of variability, which is addressed in this work. The evolution of computing power allowed also to estimate bigger (that is more precisely or complexly structured) models and to tackle more elaborate approaches (as for instance MMI, Sect. 3.4.1 or GMM-SVM pushback, Sect. 3.4.2).

3.1 Chronology of approaches to speech and speaker recognition

In the following, a very short history of speech recognition and speaker verification is presented. Where explicit references are not given, they can be found in (Furui, 2005). Since speech and Speaker Recognition is not the focus of this document, these references are not repeated. Other overviews of these domains can be found in (Furui, 1994; Juang and Rabiner, 2005). These two domains laid the basis for Language

[0] Primo giorno, Sesta, § 6. Citation used by Gion A. Caminada in the laudatio for my father, Stiftung Bündner Kunsthandwerk 2010.
German: *Gleichförmig in der Vielfalt und vielförmig in der Gleichheit, einig in der Verschiedenheit und verschieden in der Einigkeit*, Der Name der Rose, übersetzt von Burkhart Kroeber, 1982, Carl Hanser.
English: *United in their variety and varied to their unity, unique in their diversity and diverse in their apt assembly*, The Name of the Rose, translated by William Weaver, 1983, Harcourt.

Recognition (LR) since their exploration started well before the investigations in the area of LR.

In the 1950s and 1960, first attempts to exploit fundamental ideas of acoustic phonetics have been undertaken. They were mostly based on spectral resonances of the vowels using an analogue filter-bank and logic circuits.

1950s

In the 1950s, all efforts focused on vowel (RCA Labs) or vowel-based digit (Bell Labs) recognition. In 1959, first consonants have been recognized and their performance enhanced through statistics on allowable phoneme sequences in English.

In the early 1960, several hardware phoneme recognizers appeared.

mid-1960s

In the mid-1960s, elementary time-normalization methods arose, which reduced the variability of the speech recognition scores. In the Soviet Union, time aligning of utterance pairs using Dynamic Time Warping (DTW) was developed. Until late 1970s, several dynamic programming variants, including the Viterbi algorithm, became invaluable in the speech recognition domain.

In the 1960s also appeared first publications on speaker (or "talker") recognition — about one decade later than speech recognition. They use filter banks and order to measure the similarity (correlation) of two spectrograms. Later on, linear discriminators as well as formant analysis appeared in Speaker Recognition (SR).

1970s

In the 1970s, automatic speech recognition became a full-fledged topic of pattern recognition (Velichko and Zagoruyko, 1970) and linear prediction coefficients (LPC) spectral parameters were introduced (Itakura, 1975).

During this period, the IBM Labs focused on Large Vocabulary Speech Recognition (LVSR), the AT&T Bell Labs on speaker-independent speech recognition. Several speech understanding systems were developed under the DARPA (U.S. defense) program. Such as the *Hearsay I* system using semantic information or the *Harpy* system accurately recognizing speech of a 1011 words corpus, which uses graph search on a connected network issued of lexical representations of words, as well as syntactical production and word boundary rules. It was the first one using a finite state network.

In the Speaker Recognition branch, even if research on text-dependent approaches continued because of the inherent simplifications, text-independent methods were attempted quite early. For this, features that are independent of the phonetic context, but which characterize the speaker were extracted: averaged auto-correlation, instantaneous spectra covariance matrix, spectrum and fundamental frequency histograms, linear prediction coefficients, and long-term averaged spectra. Also introduced were instantaneous cepstral coefficients as well as their first and second derivatives (or first and second order polynomial coefficients), commonly named Δ and $\Delta\Delta$ (double-delta) coefficients.

1980s

The speech recognition systems developed in the 1980s focused on robustly recognizing the connected words of fluent speech with a wide variety of approaches and algorithms.

The basis of most practical systems of present times were researched in the 1980s with some major improvements dating from the 1990s. *"Speech recognition research in the 1980s was characterized by a shift in methodology from the more intuitive template-based approach (a straightforward pattern recognition paradigm) towards a more rigorous statistical modeling framework."*, as describes (Furui, 2005).

Also, in the mid-1980s, the language models (grammars) moved towards the nowadays well known *n*-grams, defining the occurrence probability of a sequence of *n* words.

Doubtlessly the most important advance in speech recognition was the HMM approach, which found its way to virtually all speech recognition research groups in the world and was also introduced to SR at about the same time (Ferguson, 1980; Rabiner, 1989). At the very end of the 1980s, first GMMs (single-state HMMs) occurred in speaker identification (Rose and Reynolds, 1990), estimated by an EM-ML algorithm (Sect. 2.4.2).

About one decade after cepstral coefficients and their deltas have been tried in Speaker Recognition (SR), they were also applied to the speech recognition task. This is one of the few examples of Speaker Recognition technologies finding application in speech recognition.

After first trials in the 1950s, neural nets were reintroduced to the speech recognition task only in the 1980s with a better understanding of the technology with its strengths and limitations.

1990s Prior the 1990s, the problem of pattern recognition followed the framework of Bayes requiring data distribution estimation. During the 1990s it was then transformed into an optimization problem where the empirical recognition errors are being minimized. This concept came along with several new techniques like discriminative training or kernel-based methods.

2000s The developments of the 2000s focus mainly on spontaneous and multilingual speech. In conjunction with full dialog systems, confidence measures were more thoroughly investigated. Also combining speech recognition with multimodality as the lip movements has shown to improve recognition in noisy environment. On the modeling level, dynamic Bayesian networks have been analyzed.

In Speaker Recognition, the problem of score normalization in order to reduce intra-speaker variation of Likelihood values was brought up in the 1990s and addressed with Likelihood ratio and a posteriori probability techniques. This normalization problem has then more seriously been researched in the early 2000s with normalizations as the mean-subtraction and standard deviation-division, where the parameters are estimated on a set (cohort) of impostor speakers, which are processed an impostor score distribution — calculated with different approaches such as Znorm, Hnorm, Tnorm, HTnorm, Cnorm and Dnorm (Bimbot et al., 2004).

Generally, normalization is carried out at different stages: At Feature Extraction level with CMS, variance normalization (Sect. 2.3.3) or feature warping, at

model level using the World Model (WM), and several score-level normalizations (Sect. 2.5.2).

The early 2000s also saw higher level features emerge for Speaker Recognition. These comprise amongst others: word idiolect, pronunciation, phone usage and prosody (Rouas and Farinas, 2004).

3.2 Early times of language recognition

Now we will have a look at the early history of automatic Language Recognition. In the first times, we nearly exclusively find the term of *Language Identification* (LID), referring to the identification evaluation mode (as opposed to detection). This has changed a lot during the last decade, due to the different NIST LREs, which are carried out as *detection* tasks. As a side effect, we see a distinction also on the corpora used for one or the other task, even if there is no data-motivated reason: OGI-MLTS and OGI-22-languages corpora are mainly used for Language Identification, whereas LDC's CallFriend and NIST's LRE data are mostly used for detection. Nevertheless, we can still find some publications doing real identification.

3.2.1 The manual or supervised era

Prior to the late 1070s/early 1980s, Language Identification (LID) has largely been guided by human experts based on linguistic units that were thought of being most likely to discriminate languages. These linguistic units have different occurrence frequencies in different languages.

mid-70s

(Leonard and Doddington, 1974) carefully select transitional and steady-state sound segments for each pair of languages. These segments are based on manual analysis of filter-bank feature vectors. Then, they calculate a time-average log-likelihood for template matching. But the processing seems cruelly to lack some BE calibration since one language achieves 0% accuracy while other achieve 90% (over a set of 5 languages). This was one of the earliest LID systems described in literature.

end-70s
early 80s

An early feasibility study (House and Neuburg, 1977) investigates Markov chains (ergodic HMMs) on some artificial data and is thus quite far from a real world application. The data being mold are published phonetic transcriptions. Other studies (Li and Edwards, 1980) apply segmentation and finite-state models to six broad acoustic-phonetic classes.

Another pilot study published in 1982 (Cimarusti and Ives, 1982) presents a totally automatic approach, where the only human interaction was the designation of the speech segments to be used. It uses pattern analysis techniques with the justification that humans are able to perform language discrimination based on "*some language-unique features of which they are apparently minimally aware*". This study

makes the fundamental assumption that introduces acoustics based Language Identification: The discriminating linguistic units used in earlier LID studies "*can be described in terms of their acoustic characteristics in the speech signal. [...] it is reasonable to assume that the acoustic description of the speech signal will retain the discriminating nature of the language-unique properties*". Based on this, features for LR need not be strictly of linguistic nature, but can also be purely acoustic. The study works with a total of three minutes of read speech for each of five (male only) speakers per language and on a set of 8 languages. The feature vectors are a combination of 100 parameters. Since these LPC-derived features are rather uncommon nowadays, they are listed hereafter without further details:

- 15 area functions
- 15 autocorrelation coefficients
- 5 bandwidths
- 15 cepstral coefficients
- 15 filter coefficients
- 5 formant frequencies
- 15 log area ratios
- 15 reflection coefficients.

The trained decision functions were Type-II exponential functions, expressed as a polynomial. This study concludes optimistically with the sentiment that an accurate feature ordering and selection strategy could improve the results even more. The pretty good results (between 77 and 93% correct classification) accrue possibly from the method of dividing the data set into training and testing subsets: The database was randomly divided so that both subsets contained an equal number of feature vectors from each language — possibly missing the fact that data of a given speaker may be attributed to both, training and testing subsets, which would partially lead to speaker recognition instead of LID.

— ◇ —

Before the 1990s, most systems were evaluated on very different and often specially designed databases and thus their results are hardly comparable.

3.2.2 First automatic approaches

Most general directions of LR research were developed out of speech and Speaker Recognition. This section shortly evokes some approaches on the way from the manual era seen above to the automated systems employed nowadays. A more extended overview of LR, as well as further references to research evoked in this section, can be found in (Zissman, 1993).

— ◇ —

late 80s In the late 1980s, formant and prosodic features were investigated and rule-

based thresholding over pitch and formant frequency variance or power density centroids was explored.

early 90s

In the early 1990s, Vector Quantization (VQ) on linear prediction coefficients (LPC) features and GMMs, as well as neural net classifiers (Cole et al., 1989) were introduced to LID. In 1991, (Savic et al., 1991; Riek et al., 1991) successfully tried HMMs and pitch contour analysis. The five states of the HMM represent different articulatory configurations of the vocal tract with the self-transitions giving the occurrence rate of a sound class. *"The observation and transition probabilities show considerable inter-language variations"*, while remaining remarkably similar within a same language. Pitch detection was conducted by offset removal and a method of autocorrelations with user-defined thresholds. Pitch evolution was analyzed on a long-term and a short-term level. The former operating on a sentence level and tracking intonation and the latter analyzing a possible tone, thus on a word level. Whereas the initial database consisted of 15 languages, only a subset of fluent bi-lingal speakers in 4 languages was used. The data being noise-free read speech, the study did not hit any error. But it was observed that only articulatory HMMs or only pitch evolution alone was not sufficient for all cases — a combination of the two approaches is necessary.

(Muthusamy and Cole, 1992) introduced probably the first time in the history of LID the representation of an entire utterance under the form of a fixed size vector. For this, the speech signal was partitioned into a token sequence of seven broad phonetic categories using neural nets and then transformed to one vector of 194 features, which was then used for LID. This system requires some hand-labeled training data and thus some language-specific phonological knowledge.

The first system not relying on transcribed training data is described in (Zissman, 1993). It features an ergodic Hidden Markov Model (HMM) with tied Gaussian mixtures for each language using Mel-scale weighted cepstrum and delta-cepstrum vectors. The presentation of this system concludes that the performance of GMMs (single state HMMs) is comparable to the multistate HMMs, which indicates that the sequential modeling capabilities inherent in the HMMs were not exploited. This could be due to the limited amount of training data or to a lack of balance between static (states' observation likelihoods) and transitional information. But as advantage, this system can easily be extended to other languages since it does not require any phonological knowledge or hand-labeled training data.

— ◇ —

summary

While early Language Recognition involved manual analyses (think of Feature Extraction by hand), first automated systems still involved some expert linguistic knowledge and manual segment selection. Later evolutions were mainly taken over from the linked domains of Speech and Speaker Recognition. In the following, we will have a slightly more detailed look at some major advances of the last decades.

3.3 Feature extraction

This and the next sections present chosen major advances in the domain of language recognition. They are ordered in sections in the same sequence as the presentation of the baseline system (in Chapter 2).

Let us start with Feature Extraction, presenting different approaches used and combinations thereof.

3.3.1 Types of features

There is a considerable variety of different kinds of parameterizations used throughout the large domain of automatic speech processing. Some of them are more frequently used in speech recognition, Speaker Recognition and again others in LR. It follows a list of some more or less widespread types of feature vectors.

We principally distinguish two basic approaches to FE, which will be detailed in the in the following.

3.3.1.1 Cepstral-based

One class of features are based on Cepstral Coefficients, extracted principally the way described in Sect. 2.3.1. Cepstra can be extracted using slightly different setups or adding further processing steps. The following list presents some of these nuances. Most can be seen as individual configuration/processing elements, which may be combined together:

different elements

Energy : The basic feature vector is augmented with one additional parameter, which contains the instantaneous signal energy.

Mel scale : The filters of the filter-bank that is applied to the spectrum are distributed non-linearly along the frequency scale, as mentioned in Sect. 2.3.1. This has the same effect as *warping* the spectrum itself to a non-linear frequency scale.

Linear scale : The more historic approach uses a linear frequency scale (as opposed to Mel warping).

Deltas : Additional parameters are appended to the basic feature vector. They contain the *deltas* (Δ) of the basic parameters. Deltas are the instantaneous evolution of the signal and can be seen as first order derivative. They indicate how the signal (or more precisely the feature vectors) change in the neighborhood of the current frame. Usually about 5 frames are used in a weighted manner for this calculation.

Double-Deltas : Similarly, double-deltas (or delta-deltas, $\Delta\Delta$) can be calculated. They represent the instantaneous evolution of the signal and it can be seen as

"acceleration" of the signal in the neighborhood of the current frame. Usually, about 9 frames are used for its calculation. They get also appended to the basic feature vector.

SDC : Shifted Delta Cepstra are multiple (single-) delta-blocks calculated in the future of the signal (looking ahead of the current frame). Their configuration is given by a set of four parameters in the form *N-d-P-K* (Sect. 2.3.1).

RaSta : RaSta filtering (Hermansky et al., 1991, 1992; Hermansky and Morgan, 1994), uses band-pass filtering in the log-spectral domain and removes slow channel variations in order to smooth over short-term noise variations and remove constant offsets. So it is a window-based filtering technique that adapts itself over time. It is an acronym of "*RelAtive SpecTrAl transform* ".

T-DCT : Temporal Discrete Cosine Transform (Kinnunen et al., 2006) is in effect rather similar to SDC. But it uses all frames of a window of a given length (SDC only use single frames therein for delta-block calculation). This also avoids SDC's problem of the same delta-block repeating some frames later in an other block. This seems an interesting approach, but it is not often employed. It may be considered, together with SDCs, as link between cepstral-based approaches and more complex Time-Frequency (TF) domain approaches.

Combinations

Some common combinations of the above steps can be observed. All the following are based on cepstra-type features:

final parameterizations

MFCC — Mel (scale) Frequency Cepstral Coefficients is perhaps the most used parameterization. It is especially used for speech recognition and for speaker verification. In MFCCs, the number of static cepstral coefficients is often 12. To these are appended energy, the deltas and all or a subset of the double-deltas.

LFCC — Linear scale Frequency Cepstral Coefficients usually come in a similar configuration as the MFCCs, but keeping a plain linear frequency scale.

(Mel-)SDC — SDCs are most often computed on MFCCs and come in a 7-1-3-7 configuration. In this work, we use this kind of features (in the stated setup), like most acoustic LR systems nowadays. This uses static cepstra of a window of 21 frames to compose one final SDC vector.

(Mel-)T-DCT — T-DCT are often applied on top of MFCCs. (Castaldo et al., 2007c) for instance finds that (T-)DCT in a 12-3-7 setup perform better for shorter utterances (10 and 3 seconds), whereas 7-1-3-7 SDCs are better for 30 s.

3.3.1.2 Linear prediction based

An other class of parameterizations are based on PLP features. Perceptual Linear Prediction (PLP) coefficients are based on concepts from psychophysics of hearing (Hynek, 1990). It avoids speaker-dependent details of the auditory spectrum and

would thus be less adapted for speaker verification. PLPs can also be combined with most of the above steps. Some commonly used ones are:

PLP — Plain Perceptual Linear Predictions.

R-PLP — RaSta filtering is usually applied to PLPs (Hermansky et al., 1991, 1992; Hermansky and Morgan, 1994), but intuitively the same can also be applied to LFCCs or MFCCs — or even be combined with SDCs. Using RaSta-PLPs is just as common as using plain PLPs.

PLP-SDC — We also shortly tried to apply the SDC approach to PLPs. The results are shown in the tables later on in this chapter. We did not cross such trials in other publications yet.

— ◇ —

other types

Other possible parameterizations include features which combine, by some trade-off, time and frequency domains (for this, see (Milner and Vaseghi, 1995) and other references indicated in (Castaldo et al., 2007c) for Time-Frequency Principal Components Analysis (PCA)). Even other features may represent prosodic or other linguistic cues (e.g. Itahashi and Du, 1995; Yan et al., 1996; Hazen and Zue, 1997; Barry et al., 2003; Rouas et al., 2003). Results comparing some of the named parameterizations will be presented later on in Table 3.2 and Table 3.3.

feature normalization

Both CMS/variance normalization and RaSta filtering have been considered here as feature normalization step. Obviously more advanced normalization techniques such as Vocal Tract Length Normalization (VTLN) exist, but they are not considered here.

3.3.2 Shifted Delta Cepstra

This section revisits the concept of SDCs, which include a certain evolution of the signal over time directly on the feature level (instead of the model level, as is done i.e. by HMMs). But this introduces some problems other types of features do not have.

— ◇ —

N-d-P-K configuration

Let us define the configuration of an SDC setup by four parameters *N-d-P-K*[1]. Let $c(t)$ be the frame at time t, which has been extracted by the first FE step (Sect. 2.3.1) N is the number of basic parameters in such vectors. This includes, in most cases, the cepstral coefficients as well as energy (usually the last parameter in this vector). This energy is most often also used in generating SDCs and thus counted in the value of N.

stacked delta-blocks

For each time t, the delta is calculated by subtracting the values of frame num-

[1] Readers knowing other publications describing SDC will see that an upper-case K has been chosen here to keep lower-case (k) as index of K.

bered $t - d$ from the frame at $t + d$ (element-wise subtraction of the vectors)[2]. An other delta-block is calculated around time $t + P$, that is frame $t + P - d$ subtracted from $t + P + d$. And further blocks around integer multiples of P in the "future" of time t. P is the number of frames between consecutive delta-block centers — it is the "Shift" in Shifted Delta Cepstra. More generally, the formula for a delta-block is

$$\Delta_k c(t) = c(t + kP + d) - c(t + kP - d) \tag{3.1}$$

where $k \in 0, \ldots, K-1$ is the index of the delta-block and $c(\blacksquare)$ is the basic feature vector. Thus, for each time t, K such delta-blocks $\Delta_k c(t)$ are calculated and stacked on top of the basic vector[3]. This yields SDC vectors of size $N + KN$ for each frame at time t.

In the present case, the parameters N-d-P-K have values 7-1-3-7, as do most other acoustic LR systems (e.g. (Burget et al., 2006; Torres-Carrasquillo et al., 2002b; Campbell et al., 2004; Castaldo et al., 2007b; Matějka et al., 2006) and a majority of the NIST LRE 2009 participants). In Sect. 2.3.1 we have extracted 6 (Mel frequency scale) cepstral coefficients. Together with energy, this gives a N of 7. Further, we have seven delta-blocks, with a shift of 3 frames, stacked on top. This yields vectors of size 56. The range of vectors that are used in the SDC calculation for time t is $t - 1$ (for $\Delta_0 c(t)$) to $t + 19$ (for $c(t + 6 * 3 + 1)$ in $\Delta_6 c(t)$). This interval of 21 frames (which corresponds to $21 * 10\,ms = 0.21\,s$) allows thus to include in a certain manner the dynamics of the signal over a reasonable time span. A duration of 21 frames corresponds roughly to the length of a (simple) syllable (Fletcher and McVeigh, 1992).

7-1-3-7

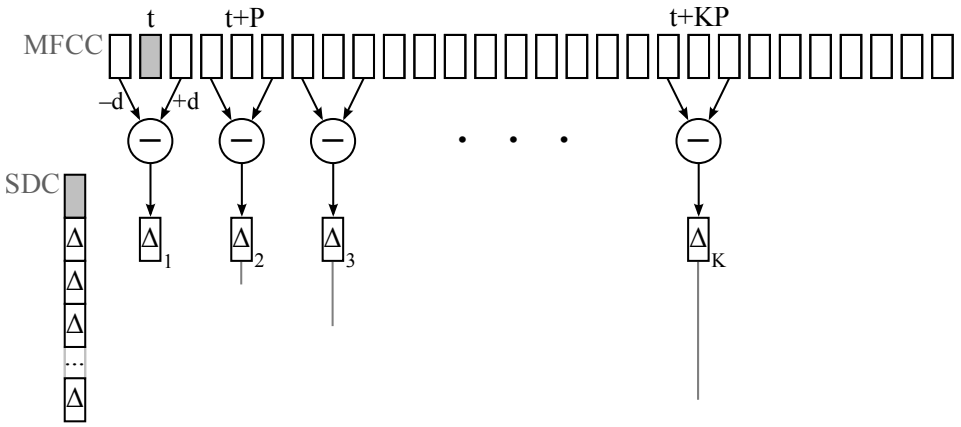


Figure 3.1: *Illustration of how the different delta-blocks (Δ) are obtained from MFCC feature vectors and concatenated together to form one SDC vector. This is repeated for every frame.*

One of the main problems of SDCs is the interaction of this "lookahead" (in our case of about 20 frames) with the Speech activity detection. The question arises

look into silence

[2]We note that for SDCs, the deltas are calculated by simple subtraction of two feature vectors, while standard cepstrum deltas are usually calculated in a weighted manner on 5 consecutive frames.

[3]Early SDC applications used to keep only the delta-blocks, but nowadays the convention is to keep the cepstral coefficients as well as the energy.

whenever it comes to the end of each speech segment, as looking ahead means looking into the silence that follows.

We imagined one simple possibility, which consists in reducing the length of each speech segment. This should avoid or at least lessen the number of SDC frames whose delta-blocks include silence frames. Some results analyzing this idea are presented in Table 3.1 (the system setup will be given together with Table 3.3). A first experiment removes the first and the 19 last frames of each speech segment (*chop 1/19*). This keeps only SDC vectors which are entirely calculated on speech frames. Compared to the base SDC system, performances are degraded a lot. The second part of the table does the same experience on non-SDC parameters. The idea is to find out what is due to the fact of avoiding silence frames (for SDCs, the cost should decrease) and on the other hand, how much the cost increases because we removed (too) many feature vectors. We observe that also for non-SDC parameters, the degradation is of the same order. As result, we see that the "look into silence" problem of SDCs can not be solved this way, since too much data is discarded of the already quite short segments (discussed in Sect. 3.3.4).

Table 3.1: *Impact of the speech segment lengths, trying to remove frames at (beginning and) end of each segment where SDC looks into silence, our NIST LRE 2005 system, in % pooled EER*

parameterization	segment reduction	accurate JFA
PLP-SDC	–	7.97
	chop 1/19	10.87
RaSta-PLP	–	7.38
	chop 0/9	9.03
	chop 1/19	10.23
	train chop 1/19	12.68

Opinion: Common literature does not pay much attention to the problem of SDCs "looking into silence" at the end of segments detected as speech by the Speech activity detection algorithms. We feel that some more in-depth research should be carried out in this field. The context of this problem is rather dependent on the nature of the segments produced by the SAD—the average length of the individual segments, the minimal duration of a segment or the smoothing (Sect. 3.3.4).
The works in (van Leeuwen and Brümmer, 2006) use a special approach to this problem (without more explanation on its usefulness though): They calculate SDCs as usual and add an additional feature parameter, which is a flag taking the value $+0.5$ if all frames used for the different delta-blocks were detected containing speech and otherwise -0.5 if at least one frame used belongs to silence. Discarded are only frames where all implied cepstral feature vectors lay in regions labeled with silence.

repeating blocks

Besides, a questionable fact about SDC is that the very same values of a big

part of the feature vector can be found again P frames later at the position one delta-block before, then P more frames later, two blocks before and so on. This happens since $\Delta_k c(t) = \Delta_{k-z} c(t + zP)$ for all $z \leq k$.

An interesting problem about repeating blocks is pointed out by (Castaldo et al., 2007c): The parameters of an SVM's mean Super-Vector will be highly correlated because of this fact. But since we keep only frames containing speech, the (correlated) segments are all relatively short (but this depends on the used SAD). The effects of the initial and final frames may, up to a certain point, compensate the intrinsic correlation of the features. This correlation problem does not occur in a GMM context.

— ◇ —

This highlighted two problems inherent to SDCs: looking into silence at the end of speech segments and the repeating blocks feature values. Both of them are unsolved or they even can not suitably be resolved.

3.3.3 Comparison of parameterizations

Having listed different parameterizations, this section gives a quick glance at some comparisons of these, applied to the NIST LRE 2005 protocol. All results presented in this section have been obtained either by our baseline system (Chapter 2) or by systems described in Chapter 4, by varying the FE configuration.

— ◇ —

frequency scales, SDC

Table 3.2 shows the effects of different parameterizations. They are analyzed as well in the MAP as the JFA context. It includes also some MMI results for comparison. On one term, the type of frequency scale is changed. In general and particularly for JFA, the Mel scale outperforms the linear one. On the other side, analyses replace the deltas and double-deltas by SDC blocks. SDCs seem to require a certain model size to develop their power. The combination of SDCs with the Mel scale is the way to go.

This Table 3.2 will also be referenced subsequently (in Sect. 3.4.1 and in Chapter 4) because it shows other interesting aspects. This table indicates results on slightly different parameterizations than used later on. These early results have to be taken with certain caution, since the internals of the systems were reworked later on. But several conclusions that will be drawn in Chapter 4 can also be observed in this table. We see for instance that JFA systems have a better evolution with growing model size than models obtained through MAP adaptation only.

Setup: Parameters with first and second order deltas are extracted with 19 basic coefficients plus energy. A feature mask with configuration 0-18, 20-50, meaning that the basic energy and the 9 last double-deltas (8 plus energy) are not used. We note that the final number of 50 parameters (the vector size d) is nearly the

same as the 56 for SDCs, which allows the results to be slightly more comparable. It uses a 3-Gaussian SAD, approximated JFA (Sect. 4.3.3) with CMS (Sect. 4.3.4). The results are obtained on the NIST LRE 2005 protocol.

Table 3.2: *Comparison of linear and Mel frequency scale cepstral coefficients with Delta-Cepstrum or with SDC, for different model sizes (256 to 2048 Gaussians), in % pooled EER, d indicates the dimension of the feature vectors*

basic parameters	delta types	*d*	system	256	512	1024	2048
LFCC	$\Delta, \Delta\Delta$	50	MAP	21.52	–	–	18.64
			JFA 40	13.36	12.63	11.99	11.88
LFCC	SDC 7-1-3-7	56	MAP	19.62	18.53	–	–
			JFA 40	14.87	11.15	11.21	9.25
			MMI	16.9	–	–	–
MFCC	$\Delta, \Delta\Delta$	21	MAP	–	36.1	–	–
			JFA 40	–	24.6	–	–
		39	JFA 40	17.8	–	–	–
MFCC	SDC 7-1-3-7	56	MAP	18.48	16.88	18.47	18.31
			JFA 40	10.46	8.97	8.78	8.27
			MMI	14.81	–	–	–

cepstra vs. PLP

Table 3.3 (as well as the earlier Table 3.1 and Table 3.5 in next section) presents results changing between cepstral based and PLP features with deltas and acceleration or with SDCs. It includes also the addition of RaSta filtering. Best results are obtained by RaSta-PLPs (at least for the accurate JFA strategy).

Setup: These systems use models of 256 Gaussian with model-space JFA (training and testing, Sect. 4.3.1 and Sect. 4.3.3) without additional CMS (Sect. 4.3.4). The column *approximated* shows performances using the UBM based model compensation and the column *accurate* the more expensive model compensation JFA approach (as detailed in Sect. 4.3.3). MFCC and PLP based SDCs are both built on 6 base coefficients plus energy and are both in the configuration 7-1-3-7 and have thus 56 parameters. The non-SDC parameterizations (MFCC, PLP and RaSta-PLP) have 12 base coefficients plus energy, deltas and double-deltas, which sum up to 39 parameters. The Speech activity detection uses a 3-Gaussian setup, as described in next section. The evaluation takes place on the NIST LRE 2005 set, 30 second utterances.

3.3.4 Speech activity detection

In this section, we continue looking into the problem of SDCs taking on silent frames (Sect. 3.3.2). We analyzed two slightly different ways of doing Speech activity detection: a 2- and a 3-Gaussian based SAD. This section gives also a hint

Table 3.3: *Comparison of MFCC and PLP parameterizations combined with deltas or SDCs, and the effect of RaSta filtering, in % pooled EER*

parameterization	dimension	approximated JFA	accurate JFA
MFCC	39	17.78	9.14
MFCC-SDC	56	14.94	10.09
PLP	39	16.55	9.11
PLP-SDC	56	11.21	9.45
RaSta-PLP	39	16.15	8.72

to even other approaches used in literature.

— ◇ —

2-Gaussians

A frequently used SAD approach is to use a model of two Gaussians: one modeling low energy and the other one high energy. This comes down to using two one-Gaussian models (and tracking their priors). The frames are then assigned to speech or silence by comparing the Gaussians' posterior probabilities. The per-utterance threshold is thus implicitly given by the Gaussian distributions. This approach is rather dependent on the initialization and thus on the prior of a frame being speech (on the estimation or the guess of how much speech a recording contains).

3-Gaussians

An other possibility to achieve SAD are strategies using for instance three Gaussians: One for low-, one for the mid-energy range and one for high energy. A speech label is assigned to frames hit by the high-energy Gaussian and by a percentage of the middle one. The results of this approach depend on the parameter indicating where to place the threshold, expressed as a percentage of the middle-energy Gaussian to take as speech.

Table 3.4 gives a small insight into our analyses to the challenge of Speech activity detection. It indicates the effect of the middle-Gaussian parameter (here called *alpha*) of the 3-Gaussian setup on the number and the lengths of the resulting segments labeled as speech. It shows also the effect of the light smoothing algorithm, particularly on the average segment length, which is lengthened by 38 and 49% relative (from 19.2 to 28.6 frames for the CallFriend data). A good set of experiments on the NIST LRE 2005 protocol presented throughout this document were carried out using this SAD approach.

Table 3.5 shows how system performance may vary upon changing SAD algorithm from the three- to the two-Gaussian setup. It shows that over 15% relative can be gained just by changing speech activity detection algorithm with an acceptable initialization. This table also shows that RaSta normalization is beneficial, since it reduces the costs up to 16.7% relative for the better 2-Gaussian SAD.

Setup: The system setups of the experiments in Table 3.5 are the same as in Table 3.3 of previous section. The setup labeled *3-Gaussian energy* uses the previously

Table 3.4: *Statistics on the segments of 3-Gaussian energy-based SAD mainly showing the effect of subsequent segmentation smoothing*

data set	*alpha*	smoothing	total segment count	length in seconds min	max	average
CallFriend-train	0.0	–	1 175 428	0.01	16.89	0.143
	0.5	–	1 272 966	0.01	69.53	0.192
	0.5	smoothing	852 715	0.03	69.53	0.286
NIST LRE 2005	0.0	–	376 407	0.01	6.99	0.135
	0.5	–	421 876	0.01	29.57	0.163
	0.5	smoothing	307 301	0.03	29.57	0.225

described three-Gaussian setup modeling energy, and which takes half of the middle-energy Gaussian as speech (*alpha*= 0.5). It is followed by a slight smoothing step to clean up far too small segments. The other setup uses 2 Gaussians based on a different implementation.

Table 3.5: *Some indications on the impact of the speech activity detection (in % pooled EER)*

parameterization	speech activity detection	approximated JFA	accurate JFA
PLP	3-Gaussian energy	16.55	9.11
	2-Gaussian energy	–	8.86
PLP-SDC	3-Gaussian energy	11.21	9.45
	2-Gaussian energy	(<10.2)	7.97
RaSta-PLP	3-Gaussian energy	16.15	8.72
	2-Gaussian energy	–	7.38

other approaches

Other researchers (BenZeghiba et al., 2008) use two GMMs for the SAD task. There, the speech GMM consists of a mixture of 2048 Gaussians and a 512 Gaussian mixture is modeling silence (presumably applied on the whole feature vectors instead of only the energy component).

More thorough systems may even train dedicated classes to avoid taking music, laughing, coughing or severe noise as speech. SAD may also directly be based on the output of a phone or phoneme recognizer.

3.3.5 Feature Extraction conclusion

As conclusion of the different analyses related to FE, we observe following: Moving from linear frequency to Mel scale (Table 3.2) is doubtlessly the way to go for LR. Taking SDCs instead of deltas and delta-deltas helps most of the time (Table 3.2 and Table 3.3). We see that the fact of applying RaSta improves results up to 16.7%

relative in the analyzed cases (Table 3.5). A well-designed and well-parameterized SAD also plays a very important role, as we witness a 15% relative improvement (Table 3.5). Best results (on accurate JFA) are obtained with PLP-SDC or RaSta-PLP parameterizations.

Since a substantial part of these analyses were made at a quite late stage, all subsequent experiments build on MFCC-SDC features in the 7-1-3-7 configuration. On the SAD side, NIST LRE 2005 experiments use 3-Gaussian setup with an *alpha* of 0.5 and smoothing, while NIST LRE 2009 builds on a 2-Gaussian approach with its considerable improvement.

3.4 Modelization

Let us now move along from FE to the next step which is the modelization. Especially in this environment, advances on various levels could be observed:

The most basic approach is to model the acoustic characteristics of a language as whole. More technically, it is the spectral content that is modeled. For this, we saw GMMs (in Sect. 2.4.1). They feature a rather high number of mixtures (typically from 64 up to some thousands) and are trained directly on the feature vectors.

We saw the EM algorithm with a ML criterion (Sect. 2.4.2), as well as MAP adaptation (Sect. 2.4.4). This section presents two other interesting GMM estimation algorithms, while the Support Vector Machine (SVM) approach is kept for Chapter 5 because it represents one of the important pillars of our works and will further be combined with JFA.

After alternative GMM estimation algorithms, this section will also outline approaches on other linguistic levels, requiring considerable effort.

3.4.1 Maximum Mutual Information

In contrast to the EM-ML or the MAP adaptation algorithms, which are generative, the Maximum Mutual Information (MMI) approach is a discriminative way to train models (Matějka et al., 2006; Burget et al., 2006; Castaldo et al., 2007b). On one side, it is similar to the MAP adaptation because it moves the previous model (starting with the UBM) towards the observed distribution of the target language. But at the same time, the algorithm tries to keep an eye on the other language models, avoiding to move the current model too much towards the others. This is what makes up the discriminative part. It can thus be seen as a discriminative derivation of MAP.

— ◇ —

MMI criterion

The objective function is a combination of a maximization for every language model (the posterior Likelihoods of the corresponding training data) and at the

same time minimizing the effect of every language model on the other languages' training data.

Instead of running the algorithm separately for each language (as is the case in a generative environment), this is expressed globally, involving all language models at once. Assuming equal priors, this can be expressed as

$$\mathcal{F}_{MMI} = \sum_{\mathcal{X}} \log \left(\frac{P(\mathcal{X}|l)^{K_\mathcal{X}}}{\sum_{k \in L} P(\mathcal{X}|k)^{K_\mathcal{X}}} \right) \tag{3.2}$$

with $\mathcal{X}$ being every training utterance of every language in turn and $K_\mathcal{X} = \frac{K}{|\mathcal{X}|}$; $0 < K_\mathcal{X} < 1$ depending on the number of frames $|\mathcal{X}|$ and a tunable constant K. The numerator of the fraction in Eq. 3.2 is the maximization part and the denominator is the minimization and corresponds to the overall posterior likelihood of the utterance $\mathcal{X}$ (given all models): $P(\mathcal{X})$. This has the tendency to move models of different languages away from each other.

mean update

A rather detailed description of MMI, which may be used as basis for an implementation can be found in (Matějka et al., 2006). We will just present and discuss the formula for updating the Gaussians' mean values without presenting all required details. For each Gaussian of every language model (thus neglecting the indices for the model and the Gaussian), we have:

$$\hat{\mu} = \frac{\vartheta^1 - \vartheta^{1*} + \mathbf{D}\mu}{\vartheta^0 - \vartheta^{0*} + \mathbf{D}} \tag{3.3}$$

where ϑ^0, ϑ^1 (and ϑ^2 for potential variance updates) are zeroth, first and second order ML statistics respectively, as defined in Eq. 2.8. Further, ϑ^{0*}, ϑ^{1*} (and ϑ^{2*}) are the statistics corresponding to the denominator of the MMI criterion of Eq. 3.2. They are responsible for the discriminative effect. They are collected over the utterances of the whole training set instead of only the target language (as is the case for ϑ^0 and affiliated). The regulation factor $\mathbf{D}$ is linked to ϑ^0.

This mean update formula has a very similar structure as the mean adaptation of MAP (Eq. 2.13)[4]. Compared to MAP, there is just a negative (first-order) summand added to the numerator, (a zeroth-order) one to the denominator and the weighting factor $\mathbf{D}$ adapted.

iterative

Since in this discriminative modeling approach, the updates of the different language models are interdependent, the whole process is iterated several times (in practice 10 to 20) in order to converge to some stable estimates.

With the necessary amount of caution, the implementation can be simplified up to needing only one single data traversal for each iteration, gathering all required statistics and weightings at once and then updating all mixtures of all language models in one run.

[4] See also the comments in last footnote of Sect. 2.4.4.

Opinion: Some thoughts on (Matějka et al., 2006), which may be helpful in the case of an implementation: The D_j in the numerator of Eq. (2) (of the cited work) should read D_{sm}. The left hand side of third line (occupation counts) of Eq. (3) should read γ_{sm}^{num} (no r in the left hand index). And as guidance for a potential implementation: The fraction in Eq. (5) does not depend on t and can thus be pulled up one level and placed between the two sums (over R and over T_r). The priors compensation factor W_s can similarly be pulled up from the lowest level of Eq. (6) to the top level of Eq. (2). Care has to be taken on the coverage of the different algorithm implementation loops since the cited work lacks some more explicit explanations...

— ◇ —

conclusion

In literature, MMI can also be found under the term of *Maximum Mutual Information estimation (MMIE)* (i.e. Castaldo et al., 2007b). Compared to EM-ML and MAP, MMI requires much more computation because of the interdependence of the models. Our few experiments with MMI (included in Table 3.2 presented previously) certainly showed an enhancement of 14% relative and 20% relative over MAP adapted models of the same size, but it is not of the same magnitude as other researchers have shown (i.e. Burget et al., 2006; Matějka et al., 2006). However our models were estimated on the *cfChans* data set only (to limit the number of frames) and it was stopped after 10 iterations.

MMI has lost of broad interest since it is outperformed by the SVM approach (presented in Chapter 5), which is also discriminative, but far less expensive.

3.4.2 GMM-SVM pushback

GMMs (Sect. 2.4.1) are a well-known generative training approach, but they lacks of the discriminative power of SVMs (that will be introduced in Chapter 5). Thus, (Castaldo et al., 2007a) propose the so called *pushback* procedure. In this approach, the GMMs are trained in a discriminative manner by exploiting the information given by the separating hyperplanes estimated by an SVM. The GMM means are moved along the directions that lay orthogonal to the separating hyperplane of an SVM, which is trained on the Super-Vector (SV) issued of the language's GMM and with a blacklist containing the other languages' SVs.

(Castaldo et al., 2007a) found that MAP adaptation is not necessary in Language Recognition since the language models can be trained robustly enough by ML estimation. We found similar conclusions, but the huge advantage of a MAP adaptation starting with a UBM is clearly that we have a correspondence between the individual Gaussians of the languages' mixtures — the Gaussians are linked and retain thus some similarity. This fact is required for several approaches, like SVMs (Chapter 5) or JFA (Chapter 4).

For GMM-SVM pushback, a GMM is adapted from the UBM for each utterance, as well in the training, as in the testing phase. The adapted means of a GMM are

then stacked to form a SV. But this simple stacking to obtain a vector in a high dimensional space is inaccurate since it does not take into account nor weights nor (co-)variances.

KL space

In order to measure a distance between two GMMs, the approximate Kullback-Leibler Divergence (KLD) can be used. If each component μ_{gp} of the mean is (mean- and variance-) normalized with respect to the UBM, the KLD is an Euclidean distance in a new space having the UBM SV in its origin (note that the weight and the variance were not adapted):

$$\tilde{\mu}^l_{gp} = \sqrt{\alpha_g} \cdot \frac{\mu^l_{gp} - \mu^{UBM}_{gp}}{\sigma_{gp}} \tag{3.4}$$

Since a translation in this space does not change the relative position of points in the same space, nor their distance, the normalization term can be reduced to the scaling factor:

$$\tilde{\mu}^l_{gp} = \sqrt{\alpha_g} \cdot \frac{\mu^l_{gp}}{\sigma_{gp}} = \frac{\sqrt{\alpha_g}}{\sigma_{gp}} \mu^l_{gp} \tag{3.5}$$

SVM

The SVs in this normalized space are used to train a linear SVM (see Chapter 5) for language l with parameters $\mathbf{w}$ and b: Its hyperplane verifies $\mathbf{w} \cdot \blacksquare + b = 0$.

(Castaldo et al., 2007a) have observed that the results were not good enough for short utterance durations when scoring directly on the SVM level. This is due to the small amount of data for the per-utterance model adaptation.

pushback

To circumvent this, the SVM information is "pushed back" to the GMM space: The means of the classical GMM language model are transformed by shifting the corresponding SV along the discriminative direction (given by $\mathbf{w}$) in the KL space:

$$\hat{\mu}^l(\omega^l) = \tilde{\mu}^l + \omega^l \cdot \mathbf{w} \tag{3.6}$$

where ω^l is a tunable shift size. This has then to be put back to the GMM domain. For each GMM model component (Gaussian g and dimension p), this is:

$$\breve{\mu}^l_{gp}(\omega^l) = \frac{\sigma_{gp}}{\sqrt{\alpha_g}} \hat{\mu}^l_{gp}(\omega^l) = \frac{\sigma_{gp}}{\sqrt{\alpha_g}} \left(\tilde{\mu}^l + \omega^l \cdot \mathbf{w} \right) = \frac{\sigma_{gp}}{\sqrt{\alpha_g}} \left(\frac{\sqrt{\alpha_g}}{\sigma_{gp}} \mu^l_{gp} + \omega^l \cdot \mathbf{w} \right) \tag{3.7}$$

This comes down to take the mean component to the KL space, shifting it along $\mathbf{w}$ and transforming it back again. As next step, the transformation could simply be applied to $\mathbf{w}$, representing the discriminative direction (obtained by simplifying Eq. 3.7):

$$\breve{\mu}^l_{gp}(\omega^l) = \mu^l_{gp} + \frac{\sigma_{gp}}{\sqrt{\alpha_g}} \omega^l \cdot \mathbf{w} \tag{3.8}$$

The shift strength ω^l has to be determined carefully. It moves the GMM means in the direction that separates it best from the other languages' models in the above KL domain, but it also moves them away from the position which best fits the

training data (ML) and thus the Likelihood of the training data decreases. The factor ω^l is chosen in such a way that the models do not move too far away in respect to the UBM, thus keeping the Log-Likelihood Ratio (LLR) positive. (Castaldo et al., 2007a) report a ω^l of 12 for all languages, but it can potentially be different for each language.

— ◇ —

conclusion

The discriminative goal behind GMM-SVM pushback is very similar to the goal of MMI (presented in Sect. 3.4.1). This pushback technique requires far less processing to be carried out than MMI since it works on mean SVs instead of traversing all frames in each iteration. (Castaldo et al., 2007a) report a speedup of a factor of 30 despite several computational enhancements on MMI. This pushback technique, called *Discriminative GMM* by (Castaldo et al., 2007a), performs better than SVMs for shorter utterances. For long utterances (30 s), it performs comparably to SVMs and MMI, whilst for shorter utterances, it is not too far away from MMI performances.

Opinion: This approach seems exceedingly interesting and promising. There should be no theoretical issues combining this approach with JFA, because of its additive nature. Some minor troubles may arise if the directions of $\mathbf{w}$ are too similar to JFA's variability directions. But this will be of no issue if the two techniques are estimated and applied sequentially.

3.4.3 Modelizations on other levels

This section will shortly present the main approaches on other linguistic levels, such as phonetics and phonotactics, as hinted in Sect. 1.3.3. Since the present works focus only on the acoustic approach, not much details will be given, but starting points for further reading will be referenced.

3.4.3.1 Phonetic

A typical phonetic approach relies on estimating a phone recognizer for every language and on scoring their output Likelihoods for the task of Language Recognition. Such phone recognizers are usually HMM-based — in one way or the other.

— ◇ —

Early phonetic systems had one single HMM per language. This ergodic (fully interconnected) HMM had its states representing more or less broad acoustic-phonetic classes. The states were implemented by a GMM (being a single-state HMM). The first ones tried by (House and Neuburg, 1977; Li and Edwards, 1980) used five states, the five broad classes.

Later on (Muthusamy et al., 1993; Zissman, 1993; Lamel and Gauvain, 1993b,a; Zissman and Singer, 1994), recognizers started using a network of interconnected small HMMs, each representing a single phoneme. Traditionally, these HMMs have three emitting GMM states arranged in a left-to-right (linear) topology.

— ◇ —

PPR

Such an approach using *language-dependent Parallel Phone Recognition (PPR)* uses a Viterbi algorithm to decode the speech frame sequence. Each language's recognizer produces a decoding Likelihood, which is used as score. The phone sequences themselves, output as a by-product by the recognizers, are not used.

Phonetic LR may also be carried out transforming a sequence of phones or acoustic-phonetic classes to a fixed-size vector (Muthusamy and Cole, 1992), which may then be used for instance in an SVM.

3.4.3.2 Phonotactic

In general, two approaches on the phonotactic level can be distinguished. These are sketched below. Other, intermediary or variant methods were also tried with similar success. The general setup is to have one or more phone/phoneme recognizers, followed by a set of language models that take care of the phonotactic structure of the languages.

— ◇ —

PR-LM

One approach consists in decoding the utterance with a universal phone recognizer (cf. the phonetic approach above). Instead of using the recognizer's Likelihood output, the sequence of phones (the transcription), which is produced at the same time but disregarded in the phonetic approach, is used.

language model

This resulting phone successions is processed by language models (abstract grammars), using one model for each target language. For this, *n-gram* modeling is typically used, which has largely been introduced in the mid-1980s[5]. This describes, which sequences of (n) phones/phonemes may occur and at which frequency they do. Its training is thus count-based with possibly a minimum occurrence thresholding. The language models are then constituted by relative occurrence frequencies and some back-off mechanism handling sequences appearing in testing data which were not observed during training. The captured units may be seen as a kind of automatically determined pseudo-syllables.

This setup is called *language-independent Phoneme Recognition followed by language-dependent Language Modeling (PRLM)* (Zissman and Singer, 1994). This technique can be of use if we have transcribed training data only for one language. Fig. 3.2 illustrates the unique phone recognizer (English in this case) as front-end and the set of n-gram language models as back-end. Each language model produces the

[5] An alternative being context-dependent phonemes...

phonotactic Likelihood, which is used as score. This figure has been copied as-is from (Zissman, 1996, p. 35, Fig. 2) since it has a long tradition and was present in a number of relevant publications covering at least 1993 through 2001 — an overview of phonotactic systems would be incomplete without including these very characteristic figures.

Figure 3.2: *The front-end of a PRLM system includes acoustic preprocessing (FE) augmented by a phone recognition step producing a sequence of phone tokens, which is modeled by n-gram language models (back-end). This figure is copied as-is from (Zissman, 1996).*

data-driven

Approaches working in a data-driven way also exist: In GMM tokenizers (Torres-Carrasquillo et al., 2002a), a GMM is trained without the need of transcriptions. It is then used to tokenize an utterance by the indices of the highest scoring Gaussian for each frame. Other recent studies (Pellegrino et al., 1999b; Verdet, 2005; Verdet et al., 2005) use slightly more complex methodologies coming from low bitrate speech coding that are also completely data-driven and produce a phoneme-like transcription into automatically determined pseudo-phone classes.

variants

Diverse variants of the PRLM approach exist, in particular may be distinguished approaches using data of only one language for training the unique phone recognizer (Hazen and Zue, 1993; Zissman and Singer, 1994; Tucker et al., 1994) or using data of several languages pooled together (Hazen and Zue, 1994).

In earlier times, English was frequently used (due to its availability) to train front-end phone recognizers. Since the 1990s, Hungarian got a lot of attention for this (promoted by the BUT lab), because Hungarian is known to cover natively a rather broad phone/phoneme inventory, which makes it appropriate as universal front-end (Matějka et al., 2005).

parallel PRLM

An other kind of approaches run several (P) phone/phoneme recognizers in parallel as front-end, producing one phone or phoneme sequence each. On top of every sequence, $|L|$ language models are trained. In testing stage, this produces $P * |L|$ scores, which can be combined and used for LR.

This setup is called *Parallel language-independent Phoneme Recognition followed by language-dependent Language Modeling (PPRLM)* or PRLM-P (Zissman, 1993). In some sort, this combines the PRLM with the PPR (phonetic) approach. This strategy has been quite promising and is still used nowadays. Its performances are generally slightly superior to those of PRLM systems, but to a huge additional cost.

An example of PPRLM system is sketched in Fig. 3.3. We can see the multiplication of the number of language models required: At the first level, P phone/phoneme recognizers are run in parallel, each producing a token sequence. At the second level, an n-gram model for each language is run on each of these phone or phoneme

sequences. These $P * |L|$ Likelihood scores are then combined by some back-end to produce the final score. This Fig. 3.3 has also been copied from (Zissman, 1996, p. 37, Fig. 3) or (Zissman and Berkling, 2001, p. 5, Fig. 2). As Fig. 3.2, it is characteristic of Zissman's publications and can easily be recognized wherever it appears.

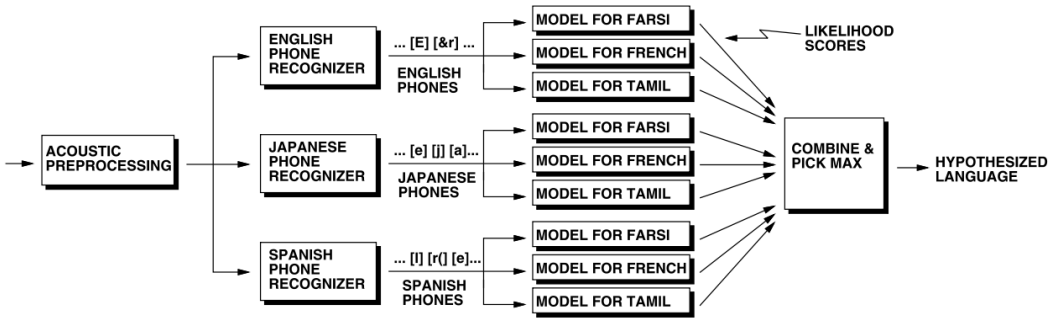


Figure 3.3: *Two-step PPRLM system with $|L|$ language models for each of P phone/phoneme recognizer outputs. This figure is copied from (Zissman and Berkling, 2001).*

variants

Among different variants of this approach, we may find the basic approach training one phone recognizer for each language (thus $P = |L|$). This has the huge drawback that phonetically transcribed training corpora are required for all languages. Thus adding a new language has a big impact and requires a lot of preparation (transcription) and a lot of engineering (one phone recognizer and $2 * |L'| - 1$ language models). (Corredor-Ardoy et al., 1997) use one language-independent phone recognizer, followed by a set of language models. Even other systems may use a phone recognizer of one language to bootstrap the transcriptions (and thus the recognizers) for all other languages (Tucker et al., 1994; Lamel and Gauvain, 1994). As somewhat intermediary systems, a setup with only a few phone recognizers ($P < |L|$) may as well be designed (Zissman and Singer, 1994; Yan and Barnard, 1995).

— ◇ —

conclusion

Phonetic and phonotactic systems require some phonetic and linguistic knowledge (beginning with the production of the transcriptions). Finely annotated training data and quite a lot of engineering is also required for such systems, while systems working on the acoustic level are kept simpler with this respect. As effect of this, it is much easier for the acoustic systems to extend the set of languages to new, possibly under-resourced languages. This is one of the reasons why we focus on acoustics-only systems in our works.

As further pointers to works using the phonotactic level, we may indicate (Yan et al., 1996; Zissman, 1996; Hazen and Zue, 1997), as well as more recent studies like (Torres-Carrasquillo et al., 2002b; Singer et al., 2003; BenZeghiba et al., 2008). Beside the reference overview (Zissman and Singer, 1994), further discussions and comparisons to acoustic approaches may be found in (Pellegrino et al., 1999a,b).

3.4.4 Conclusion on modelization

In this section about state-of-the-art modelization, we presented two interesting acoustic approaches that appeared in LR in rather recent times. Together with EM-ML, MAP and SVMs, introduced in other chapters, they account for the most used modeling approaches to acoustic LR.

We had also a quick glance at the main phonetic and phonotactic approaches, leaving aside other directions like the domain of prosody. First works on lexical modelization can be found in (Kadambe and Hieronymus, 1995).

other modelizations

Other possible approaches to modelization include techniques that are more commonly used in score processing (back-ends) like PCA, LDA and similar ones. They often have severe limitations and are not powerful enough as self-contained modeling approaches. Even other approaches may also build on Artificial Neural Networks (ANNs).

3.5 Score processing

Also, the two steps of score processing (normalization) and performance measurement are full domains of research for their own.

Hereafter are presented some other approaches to score normalization than those explained in Sect. 2.5.2. After one more utterance-wise normalization, we will principally dwell in the field of normalizing or balancing the scores output by the different class recognizers (language models in our case), the inter-language normalizations.

3.5.1 Ratio to the Universal Background Model

Instead of working with the bare Log-Likelihood of an utterance, given a language model, it can be divided by the Log-Likelihood the utterance obtains if it is tested against the UBM. This is an utterance-wise normalization and is exactly what is done in Speaker Verification with the World Model.

LLR

This way, we obtain a Log-Likelihood Ratio (LLR), which indicates if an utterance is more likely to be of the hypothesized language or of the UBM (which represents a global mean, the world). Other more complex normalizations may subsequently be applied on top of this LLR.

comparing to *divSum*

In fact, *divSum* (Sect. 2.5.2.1) emulates this approach by approximating the UBM Likelihood by the sum (or in the more accurate variant, the average) of all scores the utterance obtains. Compared to *divSum* style score normalizations, LLR alone performs between -0.1 and 2.5% relative better (analyzed over different of our systems).

limited use

Opinion: The calculation against the UBM and thus the LLR has been dropped quite early in our system development process in order to reduce calculation load. This because a similar effect is achieved with *divSum* (Sect. 2.5.2.1), which allows for great performance increases when applied with an exponent K (Sect. 2.5.2.2). Using *divSum-K* makes using LLR pointless. This can also be seen purely mathematically: The denominator (UBM LLk) is the same for all languages and thus cancels down when we apply other simple normalization techniques as the *divSum* variants or *llkMax0* (Sect. 7.2.1). Even without further normalization, UBM-LLR could be acceptable for obtaining first results.

3.5.2 Back-end score normalizations

Normalization between the orders of magnitude of the scores output by the different language recognizers is also called *calibration*. It requires a set of parameters previously estimated using some specially allocated data, called development corpus.

problem

If such a corpus is not available, to a certain extent, the data that was used for training may be used. But in our case, this was of no help, mainly due to the structure of the training data. The CallFriend corpus (and a part of VOA) contains only a small amount of files, but each containing a lot of speech. This does not produce enough values to robustly estimate normalization parameters and the utterance durations differ too much. These files may admittedly have been split into smaller chunks for this reason, cf. Sect. 8.1.1. However, similar approaches without separate development set and thus without BE fusion or calibration can also be found in literature, i.e. in (Campbell et al., 2007).

score vector

Such BEs most often operate on feature vectors, which are built by storing all scores an utterance obtained on the different models into a single vector (of size $|L|$). Each utterance thus produces one (score) feature vector for this BE step. First, the BE models get estimated and then applied to the testing data. This comes down to run the models on one single feature vector (considering utterance by utterance).

— ◇ —

back-end pipelines

In a top-down approach, this section presents an overview of several BE processing sequences, which are commonly crossed in literature (references will be given). These BE processing techniques are structured into multiple sequential steps. Potentially, these steps may also be combined in a slightly different way. The details of the individual steps will be elucidated in later sections.

3.5.2.1 Gaussian Back-End sequence

Several authors, such as (Campbell et al., 2004, 2006b) report a three-step back-end (BE). The sequence of the three steps is:

a) A feature transformation step, usually employing Linear Discriminant Analysis (LDA). Details will be presented later on in Sect. 3.5.4.
b) A set of Gaussian classifiers where each output class (in our case each language) is represented by a single (multivariate) Gaussian. They feature a global covariance, which often is a full covariance matrix. Details will follow later in Sect. 3.5.3.
c) As last step, Log-Likelihood Ratio (LLR) normalization divides each score by the average of the other $|L| - 1$ ones, as stated in Sect. 2.5.2.1 (or Sect. 3.5.1).

Often, the combination of LDA and the subsequent Gaussian modeling is presented under the term of *Gaussian Back-End (GBE)*. Sometimes the term is also used only for the Gaussian step alone (we will stick to this one). The same processing chain is also used by (Zissman, 1996; Singer et al., 2003), with the precision that diagonal covariance Gaussians are employed.

Fig. 3.4 illustrates these three steps contained in the GBE sequence described here: LDA maps the input score vectors to a space of reduced dimensionality, which is then modeled by a set of $|L|$ language-dependent Gaussians. Their Likelihoods are finally utterance-wise normalized by LLR.

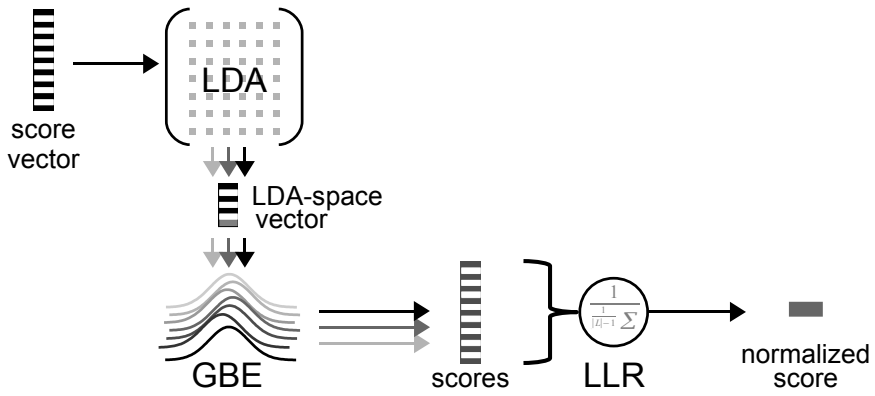


Figure 3.4: *Steps of the frequently used Gaussian Back-End for score normalization: LDA, Gaussian modeling and LLR*

Opinion: This seems appealing and should be tried in this sequence. We gave LDA a try as back-end, but got a bit clueless how to use it alone as BE for a single system (not doing system fusion) since it reduces dimensionality. Due to time and objective constraints, we did not exhaust the possible combinations of LDA with some other technique like a Gaussian modeling.

3.5.2.2 GBE with Multi-class Logistic Regression

Other researchers (BenZeghiba et al., 2009) combine a GBE with Multi-class Logistic Regression (MLR). In this case the described sequence is the following:

a) Mean normalization of the score-space feature vectors
b) Gaussian Back-End (GBE) classifiers (Sect. 3.5.3)
c) Log-Likelihood Ratio (Sect. 2.5.2.1, Sect. 3.5.1)
d) Multi-class Logistic Regression (MLR) with details later on in Sect. 3.5.6.

comparing to *divSumK*

Opinion: Some logistic regression has been tested and evaluated on our systems for a long time in parallel to the *divSumK* approach. As a conclusion, we saw that the results were roughly the same level for both approaches (even if the first is an inter-class normalization and the latter an inter-utterance one). So we kept using *divSumK* with the advantage of not requiring scores to be calculated on a development or on the training data (our case) in order to estimate the regression parameters. A setup combining these two approaches did not improve over a single one either.

3.5.2.3 GBE with Artificial Neural Network

The GBE sequence presented in Sect. 3.5.2.1 may be extended by an Artificial Neural Network. For instance (Campbell et al., 2008) use this pipeline of steps:

a) Transform scores using LDA (Sect. 3.5.4)
b) Model the resulting vectors using one tied-covariance Gaussian per language (GBE, Sect. 3.5.3)
c) The per language scores are then calibrated separately using an Artificial Neural Network (ANN). Thus the LLR normalization is replaced by an ANN.

Opinion: We did not put enough effort to experiments using ANNs and it would have needed deeper analysis and a chapter of its own in order to be of any advantage for our case. The results (not shown here) were in the same rough magnitude as other normalization approaches tested[6].

— ◇ —

BE sequence conclusion

The back-end sequences just presented gave some insight to score processing BEs, which are significantly more complex than simple utterance-wise normalizations presented in Sect. 2.5.2. The main elementary steps of the presented pipelines will be presented on the following pages.

3.5.3 Gaussian Back-End

A Gaussian Back-End (GBE) is mainly used for fusing multiple systems. Each of these usually produces one score per target language. But it can also combine systems together with additional scores, for instance scores of some out-of-set languages. At the same time, the GBE calibrates the output scores to remove biasing

[6]This approach is likely also teared down by the lack of development data.

between target languages. So it may also be applied just as calibration step for a single system.

For a given test utterance, a GBE cross-combines the set of all scores to one score for each target language. The set of input scores are stacked into a vector (as outlined in Sect. 3.5.2), which forms the new feature set for the GBE multivariate distributions.

output score

For each target language, one Gaussian distribution is estimated. The covariance is common to all language dependent Gaussians (they are called *tied*). This common within-class covariance (CWCC) may be as well diagonal or a full matrix. The output score is the a posteriori probability of a normal density $\mathcal{N}(\blacksquare|\mu,\Sigma)$ (Eq. 2.4). Since we have a common (often full) covariance matrix, the following decision function can be found by simplifying the Gaussian probability (l is the language):

$$\delta_l(x) = (x-\mu_l)^{\mathrm{T}}\Sigma^{-1}(x-\mu_l) \tag{3.9}$$

If these outputs will be used for computing an LLR (or by the similar *divSum*), it can be developed and terms that do not depend on the language (l) can be removed since they will cancel down (see BenZeghiba et al., 2009). We finally stick to

$$\delta_l(x) = (\Sigma^{-1}\mu_l)^{\mathrm{T}}x - \frac{1}{2}\mu_l^{\mathrm{T}}\Sigma^{-1}\mu_l \tag{3.10}$$

which still has Likelihood interpretation.

linear GBE

The linear part, $(\Sigma^{-1}\mu_l)^{\mathrm{T}}$, of this function can actually be seen as affine transform, which is the same as an LDA (see Sect. 3.5.4). It maximizes the ratio of the between-class to the within-class variance. The translation part, $-\frac{1}{2}\mu_l^{\mathrm{T}}\Sigma^{-1}\mu_l$, actually performs a *calibration* of the language dependent thresholds. Since this transformation is linear, we also speak of a *Linear GBE*.

log-likelihood ratio

Usually, the outputs $\delta_l(x)$ of the GBE are converted to LLRs by normalizing them with respect to the other likelihoods (see Sect. 2.5.2.1 or similarly Sect. 2.5.2.1f. and also (Campbell et al., 2004)). $\mathcal{X}$ is the input utterance and x is the corresponding score vector:

$$s_l(\mathcal{X}) = LLR(x|l) = log\Big(\frac{\delta_l(x)}{\sum_{k\neq l}\delta_k(x)}\Big) \tag{3.11}$$

posterior probabilities

The Bayes' Theorem can be used to convert LLRs to posterior probabilities by including the priors π_T and π_N of the target language and the non-target languages respectively. If they are equal to 0.5, which is the case in NIST evaluations, they cancel down.

$$s_l(\mathcal{X}) = LLR(x|l) = log\Big(\frac{\pi_T\,\delta_l(x)}{\pi_N\frac{1}{N-1}\sum_{k\neq l}\delta_k(x)}\Big) \tag{3.12}$$

3.5.4 Back-end LDA

Linear Discriminant Analysis (LDA) is commonly used for its dimensionality reduction properties.

The projection matrix which maps a test (score) vector into a subspace of reduced dimensionality is obtained by gathering, as its columns, the leading eigenvectors of the $\mathbf{W}^{-1}\mathbf{B}$ problem, where $\mathbf{W}$ is the within-class variability (the covariance Σ in the case of common/tied covariances) and $\mathbf{B}$ is the between-class variability. The number of eigenvectors required is equal to the desired dimensionality of the transformed space. But there are at most one less eigenvectors than the rank of the matrix containing the input (training data). If we have only one score per language in the input vectors (thus $|L|$ dimensions), the dimensionality of transformed space will be one less: $|L| - 1$.

LDA will fail due to the parametric assumption of Gaussian probability densities in the case where the discrimination does not lie in the means of the data, but in their variability. I.e. if we have two classes with merely the same mean, but very different variances).

3.5.4.1 LDA algorithm for dimensionality reduction

LDA is based on Sir FISHER's work (Fisher, 1936), in which he describes a projection of data points into a (sub-) space in which points of distinct classes are kept far apart (as PCA, Sect. 4.6.1, also does), but at the same time keeping the data points of a same class together.

criterion

FISHER's criterion thus uses the ratio of the *between class covariance* $\mathbf{B}$, which is to maximize, and the *within class covariance* $\mathbf{W}$, which is to minimize. We define the criterion $\mathcal{F}_{LDA}$ on the projection matrix $\mathbf{P}$ for which the ratio between the (projected) covariance matrices has to be maximized[7]:

$$\mathcal{F}_{LDA}(\mathbf{P}) = \frac{|\mathbf{P}^{\mathrm{T}}\,\mathbf{B}\,\mathbf{P}|}{|\mathbf{P}^{\mathrm{T}}\mathbf{W}\mathbf{P}|} \tag{3.13}$$

Fisher's linear discriminant

Finding $\mathbf{P}$ for which $\mathcal{F}_{LDA}$ is maximal comes down to the following eigenvalue problem (see Sect. 4.6.2 for the derivation):

$$\mathbf{W}^{-1}\mathbf{B}\mathbf{p} = \mathcal{F}\mathbf{p} \tag{3.14}$$

where the solutions $\mathbf{p}$ are the eigenvectors (corresponding to the non-zero eigenvalues $\mathcal{F}$). They form the columns of $\mathbf{P}$. This is called the FISHER*'s Linear Discriminant*.

within class covariance matrix

The *within class covariance* matrix $\mathbf{W}$ is the sum of all classes' (full) covariance

[7] Compared to the LDA employed for classification (as described in Sect. 4.6.2), the LDA here uses a projection matrix, which yields a vector in the new space, instead of projecting onto a vector using the dot product (which produces a scalar value).

matrices (μ_l being the mean of all data points of class l)[8]:

$$\mathbf{W} = \sum_{l \in L} \Sigma_l \quad ; \quad \Sigma_l = \hat{\boldsymbol{\mathcal{X}}}_l \hat{\boldsymbol{\mathcal{X}}}_l^{\mathrm{T}} = \sum_{x \in \boldsymbol{\mathcal{X}}_l} (x - \mu_l)(x - \mu_l)^{\mathrm{T}} \tag{3.15}$$

The covariance Σ_l of class l's data can be obtained by two different implementations: on the set of centered data points $\hat{\boldsymbol{\mathcal{X}}}_l = \boldsymbol{\mathcal{X}}_l \vdash \mu_l$ [9] or through the individual data point deviations from the mean.

between class covariance matrix

The *between class covariance* matrix $\mathbf{B}$ is the weighted covariance of the class means μ_l (centroids). Thus, these centroids form our data points and have first to be centered themselves around their weighted mean $\bar{\mu}$ (which corresponds to the global mean of all individual data points). The difference is only that each centroid vector may have a different weight, according to the size of each class ($\boldsymbol{\mathcal{X}}$ pooling together the data of all classes):

$$\begin{aligned} \mathbf{B} &= \sum_{l \in L} |\boldsymbol{\mathcal{X}}_l| \ (\mu_l - \bar{\mu})(\mu_l - \bar{\mu})^{\mathrm{T}} \\ \bar{\mu} &= \frac{1}{\sum_{l \in L} |\boldsymbol{\mathcal{X}}_l|} \sum_{l \in L} |\boldsymbol{\mathcal{X}}_l| \ \mu_l = \frac{1}{|\boldsymbol{\mathcal{X}}|} \sum_{l \in L} |\boldsymbol{\mathcal{X}}_l| \ \mu_l = \frac{1}{|\boldsymbol{\mathcal{X}}|} \sum_{x \in \boldsymbol{\mathcal{X}}} x \end{aligned} \tag{3.16}$$

The covariances of each class centroid (the sum elements on the first line of Eq. 3.16) have a rank of 1[10], since they are calculated on a single vector only. And thus, being the sum of such matrices, $\mathbf{B}$ is at most of rank $|L|$.

projection matrix

For obtaining the projection matrix, we need to solve the eigenvalue problem $\mathbf{W}^{-1}\mathbf{B}$. Since $\mathbf{B}$ is singular, the product is singular too (with a rank less or equal to the one of $\mathbf{B}$). Consequently, we will have at most $|L| - 1$ non-zero eigenvalues and eigenvectors for composing our projection matrix.

global covariance

We can furthermore verify that the *global (total) covariance* of the points, independent of their class, can be decomposed into the presented within class and between class covariances[11]:

$$\mathbf{V} = (\boldsymbol{\mathcal{X}} \vdash \mu)(\boldsymbol{\mathcal{X}} \vdash \mu)^{\mathrm{T}} = \mathbf{W} + \mathbf{B} \tag{3.17}$$

replace between by total covariance

In FISHER's discriminant, the between class covariance matrix $\mathbf{B}$ may likewise be

[8] Note that in the formulas of back-end processing (operating on score vectors), the indices of means μ and covariances Σ are the language and not the GMM's Gaussian component as in Sect. 2.4.1 for instance.

[9] $\boldsymbol{\mathcal{X}}_l$ is the matrix containing the data vectors of class l. μ_l is the (column) vector of the row-wise means of $\boldsymbol{\mathcal{X}}_l$, so it is the mean of all data points of class l: $\mu_l = \frac{1}{|\boldsymbol{\mathcal{X}}_l|} \sum_{x \in \boldsymbol{\mathcal{X}}_l} x$. And $\vdash$ is the operator subtracting the right-hand vector from each column of the left-hand matrix.

[10] Or a rank of 0 for a zero vector as centroid.

[11] *Opinion:* This total covariance $\mathbf{V}$ may also be seen as "mean of the covariances plus the covariance of the means " (neglecting weightings). A quite similar structure in linear algebra can be seen for the variance, which can be expressed as the "mean of squares minus square of means " (speaking of means instead of the more accurate expected value).

replaced by the total covariance matrix $\mathbf{V}$ without changing the resulting projection matrix $\mathbf{P}$ (Sakai et al., 2008)[12].

This corresponds to the original $\mathbf{W}^{-1}\mathbf{B}\mathbf{p} = \mathcal{F}\mathbf{p}$ (Eq. 3.14) with the eigenvalues $\mathcal{F}$ shifted by 1: $\mathcal{F}' = \mathcal{F} + 1$. This has the effect that eigenvalues that were zero in the classical LDA, have now the value of 1. For this reason, using the total covariance matrix assures that a $|L|$-dimensional input vector can be mapped to a $(|L| - 1)$-dimensional output space.

Excurse: Note that instead of the covariances, we may likewise use the *scatter matrices*. They are obtained by normalizing the covariances by the number of points and are commonly noted S_b and S_w. Since the objective function (Eq. 3.13) is a ratio of scatters or covariances, the normalization terms cancel down. Some publications in the domain of computer science have the tendency to mix up two terms of scatter and covariance.

3.5.5 Score Support Vector Machine

Support Vector Machines (SVMs) (described in Chapter 5) as back-end simply operate on the score feature vectors in a straightforward way. Since SVMs discriminate only between two classes, one SVM is trained for each target language using all training/development data score vectors of non-target languages as blacklist.

Upon test, the score vector is run against the SVM of the hypothesized language to obtain the output Likelihood.

3.5.6 Logistic Regression

Logistic Regression is a discriminative learning technique. Let us define a linear combination of the input scores (components of the score vector) together with a general offset. The fundamental hypothesis of Logistic Regression can be stated as:

$$f(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_J x_J \tag{3.25}$$

where β_0 is the offset (also called *intercept*) and the remaining β_j are the individual component weights, the *regression coefficients*. All these values have to be estimated.

testing

At testing time, the output of Logistic Regression is a probability (of the hypothesized class, given the observations). It is conditioned by the combination of the input scores in the *logistic* function:

logistic function

[12] In Eq. 3.13/Eq. 3.14, we replace $\mathbf{B}$ by $\mathbf{V}$ and develop:

$$\mathcal{F}'(\mathbf{P}) = \frac{|\mathbf{P}^T\mathbf{V}\mathbf{P}|}{|\mathbf{P}^T\mathbf{W}\mathbf{P}|} = \frac{|\mathbf{P}^T(\mathbf{W}+\mathbf{B})\mathbf{P}|}{|\mathbf{P}^T\mathbf{W}\mathbf{P}|} \tag{3.18}$$

$$\mathbf{W}^{-1}(\mathbf{W}+\mathbf{B})\mathbf{p} = \mathcal{F}'\mathbf{p} \tag{3.19}$$

$$\mathbf{W}^{-1}\mathbf{W}\mathbf{p} + \mathbf{W}^{-1}\mathbf{B}\mathbf{p} = \mathcal{F}'\mathbf{p} \tag{3.20}$$

$$\mathbf{I}\mathbf{p} + \mathbf{W}^{-1}\mathbf{B}\mathbf{p} = \mathcal{F}'\mathbf{p} \tag{3.21}$$

$$\mathbf{p} + \mathbf{W}^{-1}\mathbf{B}\mathbf{p} = \mathcal{F}'\mathbf{p} \tag{3.22}$$

$$\mathbf{W}^{-1}\mathbf{B}\mathbf{p} = \mathcal{F}'\mathbf{p} - \mathbf{p} \tag{3.23}$$

$$\mathbf{W}^{-1}\mathbf{B}\mathbf{p} = (\mathcal{F}' - 1)\mathbf{p} \tag{3.24}$$

$$P(+|x) = \frac{e^{f(x)}}{1 + e^{f(x)}} = \frac{1}{1 + e^{-f(x)}} \tag{3.26}$$

with $+$ indicating the positive class of the discrimination and $f(x)$ being the linear combination of Eq. 3.25.

For estimating the coefficients ($\beta_{\blacksquare}$), the Eq. 3.26 can be resolved to $f(x)$, which turns to the *logit* function: training

logit function

$$\log\left(\frac{P(+|x)}{1 - P(+|x)}\right) = f(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_J x_J \tag{3.27}$$

which is the (natural) logarithm of the *odds*. This also shows that the *logit* function is the inverse of the *logistic* function.

Having multiple training score vectors, this theoretically represents a linear system of equations, which has to be solved. But because x_j are random variables, the solution is a little more tricky than solving a simple equation system. Usually, ML estimation could be used, but in practice, it is approached for instance by iterative gradient methods to obtain satisfying solutions.

— ◇ —

From a more abstract point of view, instead of the output (p) being a simple linear combination of the input elements ($p = \mathbf{b}x + c$)[13], a certain function of the output is set equal to the linear combination: $g(p) = \mathbf{b}x + c$. In the case of Logistic Regression, $g()$ is the *logit* function.

While the extension of Logistic Regression to multiple classes (thus featuring multi-category dependent variables) is known in computer science as *Multi-class Logistic Regression (MLR)* (van Leeuwen and Brümmer, 2006), mathematicians call it *polytomous regression* or *multinomial logit modeling*.

3.5.7 Score processing conclusion

This section gave an insight into various score processing techniques. The two main kinds of score normalizations are: (i) per-utterance normalization (can be seen as Tnorm in SR) and (ii) inter-language normalization (similar to Znorm in SR).

We saw that inter-class normalization is more tricky to put into action and that usually a whole chain of different processing steps has to be designed. Most back-ends (BEs) use score vectors. They consist of all scores of an utterance presented in a stacked way. Seen from the BE's point of view, the whole recognition system producing these scores is simply an oversized front-end.

[13] c being β_0 and $\mathbf{b}$ being the vector of the remaining β_j.

3.6 Performance metric

We have seen pooled and average EER, as well as *min*-C_{avg} in Sect. 2.6.2. One further measure of system performance will be discussed in this section. This C_{llr} measure has been proposed only a few years ago. A good overview over different performance metrics (including other ones) can be found in (Brümmer and du Preez, 2006).

3.6.1 Log-likelihood ratio cost

Log-Likelihood Ratio (average) cost (C_{llr}) has been published by N. Brümmer in 2004 for the Speaker Recognition task and in (Brümmer and van Leeuwen, 2006) in the Language Recognition (LR) context. It is an empirical measure of the effective quality of information delivered to the user. In a derived form, it can also be used to measure the loss that is due to mis-calibration of the system and thus the goodness of the score normalization (Sect. 2.5.2).

— ◇ —

calibration

Mis-calibration is the problem handled by inter-language normalization (as evoked in Sect. 2.5.2) and arises on LR systems that are run as well in detection as in identification mode. The goal of calibration is to balance the score output distributions of the different language recognizers (these may simply be GMM Likelihoods). Some recognizers may have the tendency to under-evaluate Likelihoods and others may show an opposite trend. As result of this, once the scores are thresholded, the former recognizers will not produce enough positive detections (thus generating misses) and the latter will verify too many detections (false positives).

Mis-calibration is typically a problem that is important in Language Recognition, but far less in Speaker Recognition. This because in LR, we are in an environment of multiple concurrent hypotheses, whereas SR is a binary-hypothesis problem (matching speaker or not) and can be seen as a one-class open-set detection.

The explanations given by (Brümmer and van Leeuwen, 2006) highly link calibration, which is part of the score processing/normalization step (Sect. 2.5.2 and Sect. 3.5), to the performance measure and evaluation (Sect. 2.6.2). The score normalization technique is traditionally often chosen in a way to fit and to be adapted to the performance metric. Thus the performance metric has an impact to the system design — beyond the priors' configuration.

single operating point

Usually, LR systems are evaluated at only one operating point, which is given by some threshold (i.e. by EER, Sect. 2.6.2.1). Such an evaluation thus presents a view of the system performance that is biased by one unique application case. The C_{llr} measure tries to avoid such a single-threshold evaluation. But it still comes in the form of a numeric value instead of a curve like the Detection Error Trade-off (DET) curve. Such curves are hard to read in an accurate way and difficult to compare.

C_{llr} is thus a global performance measure, which means that it does not depend on a threshold. As a second (and somehow linked) advantage, it does neither depend on an application's costs (which may be different for the two types of errors, Sect. 2.6.2, Sect. 2.6.2.2). This type of measure is called to be *application-independent*. It can intuitively be seen as a measure similar to the area below the DET curve[14]. It is consequently a measure of the expected cost of a system. C_{llr}

Further, Log-Likelihood Ratio (average) cost (C_{llr}) is an information-theoretic metric, expressed in bits of Shannon entropy. It is defined as follows, assuming a closed-set evaluation: definition

$$C_{llr} = \frac{1}{|L|} \sum_{l \in L} \quad \frac{1}{|\boldsymbol{\mathcal{X}}_l|} \sum_{\mathcal{X} \in \boldsymbol{\mathcal{X}}_l} - \log_2 P_l(\mathcal{X}) \tag{3.28}$$

where $\boldsymbol{\mathcal{X}}_l$ is the set of all testing utterances $\mathcal{X}$ having l as true language and where $P_l(\mathcal{X})$ is the posterior probability, which is related to the Likelihoods via Bayes' Theorem— here using flat priors:

$$P_l(\mathcal{X}) = \frac{e^{LLk_l(\mathcal{X})}}{\sum_{k \in L} e^{LLk_k(\mathcal{X})}} \tag{3.29}$$

This corresponds to the normalized *divSum* score (without taking the log, Eq. 2.19) and which can be seen as a kind of Likelihood ratio. Consequently:

$$\begin{aligned} C_{llr} &= \frac{1}{|L|} \sum_{l \in L} \quad \frac{1}{|\boldsymbol{\mathcal{X}}_l|} \sum_{\mathcal{X} \in \boldsymbol{\mathcal{X}}_l} - \log_2 \frac{e^{LLk_l(\mathcal{X})}}{\sum_{k \in L} e^{LLk_k(\mathcal{X})}} \\ &= \frac{1}{|L|} \sum_{l \in L} \quad \frac{1}{|\boldsymbol{\mathcal{X}}_l|} \sum_{\mathcal{X} \in \boldsymbol{\mathcal{X}}_l} \frac{-\hat{s}_l(\mathcal{X})}{\log 2} \end{aligned} \tag{3.30}$$

The C_{llr} definition can thus be read as an average over Log-Likelihood Ratio scores, which are transformed to logarithms to the base 2 and put to the positive ($\hat{s}$, defined in Eq. 2.19, is usually negative). This averaging evaluation is done giving every language the same importance. Because of that, the mean over per-language averages is calculated.

It is important to observe that there are no error probabilities $P_{Miss}()$ and $P_{FA}()$, but that the average runs over all single test Likelihood ratio scores. To note is also the base 2, which allows this measure to have an interpretable unit, namely *bits*. unit

A C_{llr} of 0.0 corresponds to the perfect system not making any errors and the threshold of $\log_2 |L|$ corresponds to a system that is just as good as relying only on the priors. boundary values

When applied on the raw system output, the actual C_{llr} is evaluated. This quantity is composed of two parts, namely the *refinement* on one side and *calibration loss* on the other. actual C_{llr}

min-C_{avg} + calibration loss

The measured C_{llr} can be decomposed into these two components: The first component, *min*-C_{llr}, is obtained by finding the optimal calibration and by re-calculating the C_{llr}. Calibration loss is then simply the difference between both values: $C_{llr} - \textit{min}\text{-}C_{llr}$. In the multi-class case, the ideal calibration can not be calculated, but has to be estimated for instance through logistic regression (Sect. 3.5.6).

In summary, the refinement or discrimination part is the quality[15] of the real information *content* of the system. It is not changed by the act of calibration. However the calibration part represents the *form* of the output results[16].

— ◇ —

APE curves

One of the main drawbacks of C_{llr} is that it still depends on the priors, which can not be inferred by the system, but are really application specific. This problem can be solved by plotting the error probability against the (logit of the) prior. The C_{llr} is then the area under the curve and the maximum corresponds to the EER. Such curves are called *Applied-Probability-of-Error curves (APEs)*. The curves of the system C_{llr}, *min*-C_{llr} and the $\log_2 |L|$ threshold may be plotted in the same diagram.

It is to note, that the C_{llr} measure may only be applied if the system's output is (log-)Likelihood like, thus has (log-)Likelihood type interpretation or can be transformed to such.

The article (Brümmer and du Preez, 2006) contains an exhaustive discussion on the subject of this measure (mainly in the context of Speaker Recognition).

Opinion: Another drawback of C_{llr}, which is perhaps the main reason why C_{llr} is not yet more widely used, is its non-trivial interpretation. The values do not correspond to something intuitive like error rates, costs or rate of correctness. A solution to this could be to present C_{llr} in a form that is relative to the $\log_2 |L|$ threshold. For instance in percentage of this threshold:

$$\%relC_{llr} = \frac{C_{llr}}{\log_2 |L|} \tag{3.31}$$

This would represent a kind of total error[17] where the ideal system is at 0% while a useless system run at $\geq 100\%$. This would also allow to compare the performances of systems tested on different numbers of languages.

The steps of diverse back-end processing together with performance measurement in particular for LR can be found composed into the toolkit FoCal Multi-class, published by N. Brümmer on (FoCal, 2007). It is a collection of Matlab functions for BE score processing (and fusion, calibration...) like MLR, as well as performance measures for system evaluation like C_{llr} and APEs.

[14] More accurately the area between the coordinate origin and the curve (first quadrant).

[15] In fact, it is an inverse of quality, since the smaller the value the better.

[16] This conceptual separation of content and its form may be compared to what is done separating (X)HTML and CSS...

[17] As opposed to EER, which is a kind of half-error, since it is the rate of each of the two error types (Sect. 2.6.2), not of both together.

Part II

Novelties

Problems w/ FA:
It is more art than science
This is what makes it great...

— Factor Analysis lecture,
California State University, Northridge [0]

Chapter 4

Joint Factor Analysis

Contents

problem

The main problem inherent to any Language Recognition (LR) system are the different kinds of variabilities contained in the observed speech signal (Sect. 1.4). This is investigated in the works presented here. We address it by means of Joint Factor Analysis (JFA). This choice has been taken, because JFA has proved to work

[0] Online: http://www.csun.edu/~ata20315/psy524/docs/Psy524lecture20FA.ppt

very well in the Speaker Recognition (SR) context (Kenny and Dumouchel, 2004; Kenny et al., 2005b,a, 2008; Matrouf et al., 2007). So we apply the same approach to the LR problem with an in-depth investigation of its effects on this task.

This chapter contains a thorough description of the decomposition used in JFA (the structure of the models we use) and details the full JFA algorithm. We then proceed with presenting how JFA can be applied under different flavors in the training and the testing phases. Presented results compare the JFA approach to the baseline system and analyze variations in the modeling setups (on the GMM and the JFA side).

— ◇ —

strategy/ hypothesis

The classical model training approach keeps track of the class-dependent information only. The overall strategy proposed in this work is to keep also track of the speaker and channel variability, which come along mixed up with the (class-dependent) language information. The variability is estimated separately from the language information. This allows the variability to be removed from the observed data. The hypothesis being that the dispersion of one class' points get reduced. This would improve class separation and facilitate modeling. This makes it more easy to point out the real language dependent information and to model it concisely. At the same time, JFA may also capture speaker particularities, including and exceeding what Vocal Tract Length Normalization (VTLN) would achieve. Where VTLN is part of the FE step (Sect. 2.3 and Sect. 3.3.1) whereas JFA works (or at least is estimated) on the model level.

decomposition

In Sect. 1.4.3, we introduced the basic idea of JFA, which consists in separating the two information components: the language dependent and the language independent parts. So we remember the decomposition we sketched (in Eq. 1.1):

$$M_{observed} = M_{language} + M_{\overline{language}} \tag{4.1}$$

Let $\boldsymbol{\mathcal{X}} = \mathcal{X}_1, \mathcal{X}_2, ...\mathcal{X}_n$ be the information of n utterances of a given language l. The part that is common to all of them is the information about the language l. Everything else, the information not belonging to $\mathcal{X}_1 \cap \mathcal{X}_2 \cap ... \cap \mathcal{X}_n$, corresponds to the nuisance or the session variability. It is caused by a multitude of things like the speaker characteristics, room acoustics, microphone and the transmission channel, as detailed in Sect. 1.1 and Sect. 1.4.

We may depict this as follows: The common part can be described as being the average over all utterances of one language. The variability part (speaker, channel etc.) would then consist in all deviations from this average. In JFA, these variability deviations are explicitly modeled in order to recognize or to estimate them again in a later stage (notably during testing).

strategy

The basic strategy behind the JFA paradigm is that, for one utterance, the speaker/channel component (the unwanted part) is not estimated from its data alone, but also on a large number of utterances coming from different speakers

as well as different languages. Consequently, the language independent part (the variability) will have a globally defined nature.

We assume that each observed utterance $\mathcal{X}_i$ may be represented by a model $M_{\mathcal{X}_i}$ from $\mathbb{R}^D$. It is composed of the set of model parameters to be estimated from $\mathcal{X}_i$. In fact, $\mathcal{X}_i$ is the observation, the set of data frames extracted from a given audio recording by the FE step.

During training, we have the language labels and thus, we can calculate the language-dependent part (the per-language average). Once we have the language dependent part $M_{language}$, we can also calculate the variability term $M_{\overline{language}}$ for each utterance, in consequence of Eq. 4.1. But during testing, we are left alone with only one utterance and we can not compute $M_{language}$ as average. We thus have to find a way to be able to separate the two information parts during the testing stage. For this, we consider developing the model as follows:

$$M_{\mathcal{X}_i} = M_{language} + \mathbf{U}\mathbf{x}_{\mathcal{X}_i,\overline{language}} \tag{4.2}$$

where $M_{language}$ is a vector of parameters that contain the information about the language of interest and $\mathbf{U}\mathbf{x}_{\mathcal{X}_i,\overline{language}}$ is the nuisance or session component of utterance $\mathcal{X}_i$.

nuisance sub-space

Further, this $\mathbf{U}\mathbf{x}_{\mathcal{X}_i,\overline{language}}$ part is composed of two terms: The term $\mathbf{U}$ is a matrix with R columns (the rank of this matrix is R and it is low with respect to the size of M). $\mathbf{U}$ is estimated using a large amount of data corresponding to different speakers or recordings. It is estimated globally, so it is the same for all languages. This gives the variability tracking its global nature. The sub-space generated by the vector columns of $\mathbf{U}$ represents this useless information. The term $\mathbf{x}_{\mathcal{X}_i}$ is a vector characterizing the current recording (with respect to the language of $\mathcal{X}_i$). It thus contains the troublesome information, the nuisance. So the $\mathbf{x}$ vector contains the variability factors in the sub-space opened by $\mathbf{U}$. Or expressed the other way around, the vector $\mathbf{x}$ is projected to the domain of M thanks to the $\mathbf{U}$ matrix.

JFA hypotheses

JFA consists in estimating the different terms of Eq. 4.2, in particular the global $\mathbf{U}$ matrix and the vector $\mathbf{x}_{\mathcal{X}_i}$ for each utterance. The success of the nuisance JFA modeling depends mainly on the correctness of the hypothesis that this variability is located in a sub-space of low dimension. It also relies on the hypothesis that the language and channel effects are of additive nature. The very good results obtained in Speaker Recognition (SR) by (Kenny et al., 2005b; Vogt et al., 2005; Matrouf et al., 2007; Kenny et al., 2008; Matrouf et al., 2011) show that these hypotheses are at least satisfied in the SR task. The same paradigm can be applied to several different audio pattern classification tasks, such as LR (described here) (Brümmer et al., 2009; Verdet et al., 2009b, 2010b,a; Matrouf et al., 2011) or even for video genre classification (Matrouf et al., 2011). In these different application tasks, the nuisance is not defined in the same way and it can be of very different nature. For example, the identity of the speaker, which is the information to be modeled in SR systems, is however part of the troublesome variability in LR or in video genre classification.

difference to SR

While JFA has triggered significant advances in speaker verification, the context

of LR is substantially different from SR. Even if evaluated under a detection task, LR is a typical multi-class problem, whereas SR is a typical verification (target vs. world) problem. Each class has far more training data than is usually the case for speaker verification. Bigger models can thus be estimated or they may be estimated more robustly. Some analyses presented in Sect. 4.4.3, Table 4.4 (and published in (Verdet et al., 2009b)) show this on different model sizes. On the other side, in LR we have less classes — only some languages instead of a lot of speakers. But there are a lot of different sessions for each language. Generally, each recording is produced by a different speaker. This big speaker variability may be caught at the same time as the finer grained variabilities of the channel.

compensation when testing

In summary, we need several utterances for every language (the more the better). In order to detach language dependent information from session variability, we consider the language part being the information that is common to all utterances of that language and the remaining part being the session variability. In testing stage, where we are left alone with only one session, the perturbing variability contained in the data is estimated and removed (compensated). This will be done based on the structure of the variabilities seen in the training data and caught by the matrix **U**. What remains should hopefully emphasize the useful part of the information and thus, the classification should be more precise. The matrix **U** catches inter-speaker and channel variability of all languages' training utterances. Later on, while compensating, the variability estimation is not limited to the variabilities observed within the target language's training data, but any variability structure seen in any of the languages can be recognized and compensated. The variability is thus thought to be independent of the language and to be of a global nature. This allows thus to benefit to a maximum from the information about variability contained in the whole training dataset.

— ◇ —

As short summary, the principle of JFA consists in decomposing each utterance into a language-dependent and a language-independent part. The language part is tied among all training utterances of a same language. The variability part is supposed to be located in a low-dimensional sub-space, thus constrained by a global low-rank matrix.

4.1 The Factor Analysis model

This section defines and describes finely the different model parts of the JFA decomposition. It builds on the model and the principles sketched in Sect. 1.4.3. Until now, this chapter presented the decomposition in an abstract way. This will now be concretized, which will lead to the implementation algorithm.

— ◇ —

super-vector

In the following and similarly to the baseline system (Chapter 2), we will work

with Gaussian Mixture Models (GMMs) and apply JFA on them. Consequently the vector of model parameters M is a GMM mean Super-Vector (SV), which we will write $\mathbf{m}$. A Super-Vector is obtained from a GMM by stacking its G mean values μ_g (of dimension d) into one big (super-) vector. Hence, JFA works in mean Super-Vector space, whose dimensionality is Gd.

The different mean SVs $\mathbf{m}_l$ corresponding to the different languages l are assumed to be statistically independent (with respect to l). They are further assumed to have a normal prior distribution with mean m and variance $\mathbf{D}\mathbf{D}^{\mathrm{T}} = \frac{\Sigma}{\tau}$. m and Σ are the parameters of the GMM-UBM, as defined in Sect. 2.4.3 and τ is the *relevance factor* required in the standard MAP adaptation (Sect. 2.4.4 and (Reynolds et al., 2000)). The justification of this form concerning the inter-language variability can be found (for the case of speaker verification) in (Kenny and Dumouchel, 2004). Based on this, the random variable $\mathbf{m}_l$ can be written as: MAP

$$\mathbf{m}_l = m + \mathbf{D}\mathbf{y}_l \tag{4.3}$$

where $\mathbf{y}_l$ is a latent random vector variable distributed according to the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Given some adaptation data for language l, MAP adaptation consists in estimating the a posteriori distribution of $\mathbf{m}_l$. The a posteriori distribution of $\mathbf{y}_l$ is shown to be normal (Kenny and Dumouchel, 2004). By this fact, the MAP Point-Estimate of $\mathbf{y}_l$ is the mean of this distribution. Actually, this language model (Eq. 4.3) is equivalent to the one obtained by Reynold's MAP (Reynolds et al., 2000) and described in Sect. 2.4.4, where the UBM is combined with the language's sufficient statistics in a weighted way. Here, m corresponds to the (mean SV of the) UBM and the matrix $\mathbf{D}$ plays the weighting role and depends on the adaptation regulation factor τ.

Suppose that for a language l, we have obtained a MAP Point-Estimate m_l of $\mathbf{m}_l$ by using some language adaptation data. Given a collection of utterances for the language l, let $\mathbf{m}_{(h,l)}$ denote the Super-Vector corresponding to the utterance h ($h = 1, 2, ...$). For a fixed l, assume that all GMM mean SVs $\mathbf{m}_{(h,l)}$ are statistically independent (thus with respect to h). If we assume further that the prior distribution of $\mathbf{m}_{(h,l)}$ is normal, then we can get up, similarly to Eq. 4.3 but here for one particular language, with writing $\mathbf{m}_{(h,l)}$ as: variability sub-space

$$\mathbf{m}_{(h,l)} = m_l + \mathbf{U}\mathbf{x}_{(h,l)} \tag{4.4}$$

where $\mathbf{x}_{(h,l)}$ is a latent random vector variable distributed according to the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Given some adaptation data for language l and channel h (typically one utterance), MAP adaptation consist in estimating the a posteriori distribution of $\mathbf{x}_{(h,l)}$. This distribution can be shown to be normal. And the MAP Point-Estimate of $\mathbf{x}_{(h,l)}$ is the mean of this distribution.

We see that we end up with the structure we saw in Eq. 4.2. Under the assumptions stated at the beginning of this chapter, m_l corresponds to the language-dependent part and $\mathbf{U}\mathbf{x}_{(h,l)}$ is the variability. $\mathbf{U}$ is a matrix of low-rank R and $\mathbf{x}_{(h,l)}$ a R-vector.

full model

In order to integrate language modeling (MAP) and the channel variabilities in the same framework we will work with the model obtained by substituting Eq. 4.3 into Eq. 4.4 (m_l is replaced by its corresponding random variable $\mathbf{m}_l$). Thus the final model is given by:

$$\mathbf{m}_{(h,l)} = m + \mathbf{D}\mathbf{y}_l + \mathbf{U}\mathbf{x}_{(h,l)} \tag{4.5}$$

where $\mathbf{m}_{(h,l)}$ is the channel-language dependent mean Super-Vector, $\mathbf{D}$ is a ($Gd \times Gd$)[1] diagonal matrix, $\mathbf{y}_l$ the language vector (a Gd-dimensional vector), $\mathbf{U}$ is the session/channel variability matrix of low rank R (thus a matrix of size $Gd \times R$) and $\mathbf{x}_{(h,l)}$ are the channel factors (an R-dimensional vector). Analyses by (Vair et al., 2006) show that $\mathbf{x}_{(h,l)}$ depends only very weakly on l and that it can be assumed to be independent for the reason of simplification and computation speedup.

The different terms may be explained as follows:

- $\mathbf{m}_{(h,l)}$ are the different training utterances we observe
- m represents the UBM and can be seen as the global or overall mean of the training data; this term does not change
- $\mathbf{D}\mathbf{y}_l$ is the language-dependent term and can be seen as mean of all utterances that belong to language l (actually the difference between that mean and m); this term is composed of:
  - $\mathbf{D}$, which is the weighting term; it depends on the regulation factor τ of the MAP adaptation
  - $\mathbf{y}_l$ are the language factors; they are the same for all utterances h of a given language l
- $\mathbf{U}\mathbf{x}_{(h,l)}$ is the term that catches the troublesome variability; it is composed of
  - $\mathbf{U}$, which is a global matrix describing the sub-space in which speaker/channel variabilities are thought to live
  - $\mathbf{x}_{(h,l)}$ are the nuisance factors in that sub-space; they vary from utterance to utterance

In summary, the observed utterance is decomposed into a global mean (UBM), a per language-mean and the variability leftovers.

frames' distribution

We saw that all frames, independent of the language and the session are assumed to be represented by the GMM parameters μ_g and the frame (co-)variance Σ_g.

latent variables

In the SV space, in which JFA works, the global average term is represented by the UBM SV with mean m, and is thus not a distribution. By Eq. 4.5, the latent variables $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$ define a GMM mean SV. The observation sequence (the frames) is assumed to be generated from the GMM that corresponds to this SV. Latent variables $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$ are both assumed to be sampled from standard Gaussian prior distributions $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Consequently, the language-dependent part $\mathbf{D}\mathbf{y}$ has also a mean of 0 with a covariance of $\mathbf{D}\mathbf{D}^{\mathrm{T}}$ and the language-independent

[1] G being the number of Gaussians in the GMM (Sect. 2.4.1) and d being the dimensionality of a feature vector at the output of the FE step.

part **Ux** is assumed to be a priori distributed on $normaldistribution(0, \mathbf{UU}^{\mathrm{T}}$. $\mathbf{DD}^{\mathrm{T}}$ represents the variability of the language mean SVs. It corresponds to the across- (or inter-) class covariance matrix. $\mathbf{UU}^{\mathrm{T}}$ is the speaker and channel variability and corresponds thus to the within-class covariance matrix. Hence, as we assume the language and the session variabilities to be independent, the total a priori variability, which is the covariance of the different $\mathbf{m}_{(h,l)}$, is given by: $\mathbf{DD}^{\mathrm{T}} + \mathbf{UU}^{\mathrm{T}}$.

posterior distributions

Given the priors on latent variables $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$ and given an observation sequence, one can obtain a posterior distribution (and hence MAP Point-Estimates) of $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$. The MAP Point-Estimate of $\mathbf{y}_l$ for language l is $y(l)$. And the MAP Point-Estimate of $\mathbf{x}_{(h,l)}$ is $x(h,l)$ with a (a posteriori) distribution $\mathcal{N}(\mathbf{L}^{-1}\mathbf{B}, \mathbf{L}^{-1})$. In consequence, the **Ux** part has an a posteriori distribution $\mathcal{N}(\mathbf{UL}^{-1}\mathbf{B}, \mathbf{UL}^{-1}\mathbf{U}^{\mathrm{T}})$. While $\mathbf{m}_{(h,l)}$ has a prior distribution with mean $\mathbf{m}$, its posterior distribution has the mean $m + \mathbf{D}y(l) + \mathbf{U}x(h,l)$, which is depicted by the JFA formula (Eq. 4.5). All derivations and the posterior distributions for $\mathbf{y}$ can be found in (Kenny et al., 2005a) and in the algorithm presented later on in this chapter (containing also the definitions of $\mathbf{L}(h,l)$ and $\mathbf{B}(h,l)$).

This framework (JFA formula Eq. 4.5) is what we need for enrolling a language model and estimating channel factors. Using segments from many utterances (of several languages), the model parameters can be estimated using the EM algorithm described in Sect. 2.4.2, where:

- in the E-step, MAP Point-Estimates of $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$ are calculated (Eq. 4.11 and Eq. 4.12); $\mathbf{y}_l$ is constrained not to change for segments of the same language.
- in the M-step, model parameters are updated (Eq. 4.13 and Eq. 4.14) to increase likelihood of the training data by maximizing the EM auxiliary function (depending on the posterior distribution of the latent variables)

approximations

In our implementation, we make some approximations:

- Alignment of frames to Gaussian components is given by the UBM (rather than the two-level generative model itself). This simplifies the mathematical formulation and allows to work only with sufficient statistics (Eq. 4.8 and Eq. 4.9).
- Only the **U** matrix is estimated - m, Σ and weights are copied from the UBM and not updated during the M-step, **D** is set in an ad-hoc way, so that without **U**, MAP point estimation of $\mathbf{y}_l$ becomes equivalent to relevance MAP adaptation.
- MAP Point-Estimates of $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$ are obtained using a Gauss-Seidel-like approximation method as proposed by (Vogt et al., 2008) rather than estimating the joint posteriors for $\mathbf{y}_l$ and all $\mathbf{x}_{(h,l)}$ corresponding to all sessions of the given language.

4.1.1 Complete Joint Factor Analysis

The model in Eq. 4.5 can be spun further. We saw in Sect. 1.1 and Sect. 1.4 that the variability comes under various forms. It is due to the speaker characteristics, as well as the environment and the channel. This may lead to the idea to separate even the different variabilities, thus to attempt an extended decomposition.

— ◇ —

multiple subspaces

In consequence, we may try to put the inter-speaker variability into one subspace and constrain the channel variability into a different low-dimensional subspace. This assumes that these two variabilities are of additive nature. The complete Joint Factor Analysis model becomes:

$$\mathbf{m}_{(h,s,l)} = m + \mathbf{D}\mathbf{y}_l + \mathbf{V}\mathbf{z}_{(s,l)} + \mathbf{U}\mathbf{x}_{(h,s,l)} \tag{4.6}$$

where we expressively distinguish between the speaker s and the speaker's utterance (h, s). The term $\mathbf{V}\mathbf{z}_{(s,l)}$ only depends on the speaker (and the language l) whereas the term $\mathbf{U}\mathbf{x}_{(h,s,l)}$ depends on the individual utterance. Similarly to the other terms, $\mathbf{V}\mathbf{z}_{(s,l)}$ is distributed with mean 0 and covariance $\mathbf{V}\mathbf{V}^{\mathrm{T}}$ and consequently $\mathbf{z}_{(s,l)}$ is also of standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

hypothesis

This extended decomposition is thus a 4-level decomposition: the global mean (UBM), the language-dependent part, the speaker-dependent part (several speakers per language) and the leftover is the utterance-dependent part. It is motivated by the hypothesis that the speaker and the channel variabilities are of different natures and that they are consequently located in distinct sub-spaces. In addition, we consider also the fact that JFA in SR is effectively capable to separate the speaker characteristics from the session/channel variability. But in LR, the speaker is part of the nuisance and the language has to be separated out.

different means

The first term is the global mean of all training data, the second term is the per-language average, the third term ($\mathbf{V}\mathbf{z}_{(s,l)}$) is also the mean, but over for speaker-language combination individually.

not possible

This complete JFA model would condition to have at one's disposal several utterances for each speaker. Unfortunately, the large majority of the (LR-centered) corpora include only one utterance per speaker. That is, every recording file belongs to another speaker. Consequently, this decomposition can hardly be applied in our LR context. But the model in Eq. 4.6 would surely be a more suitable representation of the reality.

As a side-observation of the drawback that we do not have the required data at our disposal, our experiments also analyze if these different variabilities can be handled at once, in one and the same sub-space.

i-vectors

Excurse: A recently appeared technique goes even further and includes also the information of the wanted class (the language in our case) into the unique sub-space.

The generated sub-space is called *total variability space*[2] and the factors therein are called *i-vector* (Dehak et al., 2009, 2011). It may be sketched as: $\mathbf{m}_{h,s} = m + \mathbf{T}\mathbf{w}_{h,s}$. In this context, JFA is used as a new front-end. Because the target information (the speaker in the SR case) is also included, the class modeling has to be done in a subsequent step applied to these i-vectors and featuring full-fledged modelization techniques (as the LDA-WCCN[3] stack for instance (Dehak et al., 2011)). In the research presented in this document, JFA is not used as a front-end, but to model the language and the variabilities separately, but in one run and in the same framework (Eq. 4.5).

extended speaker factors

Excurse: In the case of Speaker Recognition, (Kenny and Dumouchel, 2004) go even one step further. They separate the original speaker-dependent term $\mathbf{D}\mathbf{y}_l$ into two parts: One continues to be $\mathbf{D}\mathbf{y}_l$ with $\mathbf{D}$ being square diagonal and a second term $\mathbf{V}\mathbf{z}_{(l)}$, which constrains also a part of the speaker-dependent information into a low-dimensional sub-space, in a very similar way to what happens in the $\mathbf{U}\mathbf{x}$ term: $\mathbf{m}_{h,s} = m + \mathbf{D}\mathbf{y}_s + \mathbf{V}\mathbf{z}_s + \mathbf{U}\mathbf{x}_{h,s}$, where s is the speaker. The dimensionality of these additional speaker factors $\mathbf{z}_{(l)}$ is usually just a little bigger than the dimensionality of $\mathbf{x}$. The covariance $\mathbf{V}\mathbf{V}^{\mathrm{T}}$ tries thus to catch (or to learn) a part of the inter-speaker variability from the data (utterances/sessions) itself, instead of relying only on the predefined $\mathbf{D}\mathbf{D}^{\mathrm{T}} = \frac{\Sigma}{\tau}$ covariance.

D based on data

Excurse: The idea about trying to learn the inter-speaker variability (covariance) directly from the data can also be spun further, as shown by a recent study (Burget et al., 2009). Instead of adding an additional speaker-dependent term as in (Kenny and Dumouchel, 2004), the covariance $\mathbf{D}\mathbf{D}^{\mathrm{T}}$ can also be caught directly from the data instead of having it fixed to $\frac{\Sigma}{\tau}$. This goes into a similar direction as the comments in the previous paragraph.

4.2 Algorithm for JFA estimation

The success of the JFA model relies on a good estimation of the variability matrix $\mathbf{U}$. This is possible thanks to a sufficiently high amount of data, where several different utterances per language are available. In the following, all details allowing the implementation of JFA within the GMM-UBM framework are given. For more information concerning equation derivations see for example (Vogt et al., 2008) and partly (Kenny et al., 2005a).

algorithm summary

The goal of the algorithm presented here is first of all to estimate the matrix $\mathbf{U}$ of the JFA formula (repeating Eq. 4.5):

$$\mathbf{m}_{(h,l)} = m + \mathbf{D}\mathbf{y}_l + \mathbf{U}\mathbf{x}_{(h,l)} \tag{4.7}$$

[2] This total variability sub-space has typically a dimensionality of 400 or more.

[3] Within Class Covariance Normalization

At the same time, the vectors $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$ are also obtained. Given is on one side the UBM, whose means are stacked to the mean SV m and whose covariance matrix has influence on $\mathbf{D}$. On the other side, we will gather statistics on the training data, which will be used as left-hand side of the JFA formula. Having obtained this, MAP Point-Estimates of the latent variables $\mathbf{y}_l$ and $\mathbf{x}_{(h,l)}$ are calculated roughly by resolving the JFA formula to the one or the other. Finally the matrix $\mathbf{U}$ gets estimated.

4.2.1 General statistics

General statistics on the data have to be gathered for estimating the latent variables and the $\mathbf{U}$ matrix of Eq. 4.5. These are the zeroth-order and first-order statistics with respect to the UBM[4].

zeroth-order statistics

Let $\mathbf{N}(l)$ and $\mathbf{N}(h,l)$ be vectors containing the zeroth-order language-dependent and utterance-dependent statistics, respectively. Both are of dimension G for a particular language or utterance. Each component corresponds to one Gaussian g and is defined as:

$$\mathbf{N}_g(l) = \sum_{f \in l} \gamma_g(f) \quad ; \quad \mathbf{N}_g(h,l) = \sum_{f \in (h,l)} \gamma_g(f) \tag{4.8}$$

where $\gamma_g(f)$ is the a posteriori probability of Gaussian g for the cepstral vector of observation f (cf. Eq. 2.5 and Eq. 2.7). In Eq. 4.8, $\sum_{f \in l}$ denotes the sum over all the frames that belong to language l and $\sum_{f \in (h,l)}$ denotes the sum over all the frames that belong to utterance h of language l.

first-order statistics

Let $\mathbf{X}(l)$ and $\mathbf{X}(h,l)$ similarly be the vectors that contain the first-order language-dependent and utterance-dependent statistics, respectively. For a language or an utterance respectively, the dimensions of $\mathbf{X}(l)$ and $\mathbf{X}(h,l)$ are equal to Gd. Each block corresponding to a Gaussian g has a dimension of d (which is the dimension of f). They can be defined as:

$$\mathbf{X}_g(l) = \sum_{f \in l} \gamma_g(f) \cdot f \quad ; \quad \mathbf{X}_g(h,l) = \sum_{f \in (h,l)} \gamma_g(f) \cdot f \tag{4.9}$$

4.2.2 Latent variables estimation

The estimation equations for $x(h,l)$ and $y(l)$ in this section represent MAP Point-Estimates of the channel factors $\mathbf{x}_{(h,l)}$ and the language vector $\mathbf{y}_l$, respectively[5].

cleaned statistics

Let $\overline{\mathbf{X}}(l)$ and $\overline{\mathbf{X}}(h,l)$ be the channel- and the language-independent statistics,

[4] All the posterior probabilities are computed on the UBM.

[5] For an easier notation, the indices have been moved into parentheses for the Point-Estimates

respectively, defined as follows:

$$\begin{aligned}\overline{\mathbf{X}}_g(l) &= \mathbf{X}_g(l) - \sum_{h\in l} \mathbf{N}_g(h,l)\cdot\{m + \mathbf{U}x(h,l)\}_g \\ \overline{\mathbf{X}}_g(h,l) &= \mathbf{X}_g(h,l) - \mathbf{N}_g(h,l)\cdot\{m + \mathbf{D}y(l)\}_g\end{aligned} \tag{4.10}$$

where $\overline{\mathbf{X}}(l)$ is used for estimating the language vector (speaker and channel effects are discarded), while $\overline{\mathbf{X}}(h,l)$ is used for estimating the channel factors (language effects are discarded)[6].

Let $\mathbf{L}(h,l)$ be a $R \times R$ dimensional matrix and $\mathbf{B}(h,l)$ a vector of dimension R, defined by:

$$\begin{aligned}\mathbf{L}(h,l) &= \mathbf{I} + \sum_{g\in \text{UBM}} \mathbf{N}_g(h,l)\cdot\mathbf{U}_g^T\cdot\Sigma_g^{-1}\cdot\mathbf{U}_g \\ \mathbf{B}(h,l) &= \sum_{g\in \text{UBM}} \mathbf{U}_g^T\cdot\Sigma_g^{-1}\cdot\overline{\mathbf{X}}_g(h,l),\end{aligned} \tag{4.11}$$

where Σ_g is the covariance matrix of the g^{th} UBM component. $\mathbf{L}(h,l)$ corresponds to the inverse of the covariance of $\mathbf{x}_{(h,l)}$'s posterior distribution, as detailed in (Kenny et al., 2005a), where the proofs can be found.

By using $\mathbf{L}(h,l)$ and $\mathbf{B}(h,l)$, we can obtain $x(h,l)$ and $y(l)$ MAP Point-Estimates from the following equations[7]: MAP Point-Estimates

$$\begin{aligned}x(h,l) &= \mathbf{L}(h,l)^{-1}\cdot\mathbf{B}(h,l) \\ y_g(l) &= \frac{\tau}{\tau + \mathbf{N}_g(l)}\cdot\mathbf{D}_g\cdot\Sigma_g^{-1}\cdot\overline{\mathbf{X}}_g(l),\end{aligned} \tag{4.12}$$

where $\mathbf{D}_g = \frac{1}{\sqrt{\tau}}\Sigma_g^{1/2}$ (resulting from $\mathbf{D}\mathbf{D}^{\mathrm{T}} = \frac{\Sigma}{\tau}$; τ is set to 14.0 in our experiments).

4.2.3 Inter-speaker/channel matrix estimation

The $\mathbf{U}$ matrix can be estimated row by row, with $\mathbf{U}_g^i$ being the i^{th} row of $\mathbf{U}_g$; thus:

$$\mathbf{U}_g^i = \mathcal{L}(g)^{-1}\cdot\mathcal{R}^i(g) \tag{4.13}$$

where $\mathcal{L}(g)$ and $\mathcal{R}^i(g)$ are given by:

$$\begin{aligned}\mathcal{L}(g) &= \sum_l\sum_{h\in l}\mathbf{N}_g(h,l)\cdot\left(\mathbf{L}(h,l)^{-1} + x(h,l)x(h,l)^T\right) \\ \mathcal{R}^i(g) &= \sum_l\sum_{h\in l}\overline{\mathbf{X}}_g(h,l)[i]\cdot x(h,l)\end{aligned} \tag{4.14}$$

Excurse: We notice that $\mathbf{U}$ does not depend directly on something that depends local minimum

[6] The structure of Eq. 4.10 can be seen as roughly resolving the JFA formula to $\mathbf{D}\mathbf{y}_l$ and $\mathbf{U}\mathbf{x}_{(h,l)}$ respectively: i.e. $\mathbf{U}\mathbf{x}_{(h,l)} = \mathbf{m}_{(h,l)} - (m + \mathbf{D}\mathbf{y}_l)$.

[7] Seen in a rough (not so mathematical) way: $\mathbf{x} := \frac{\mathbf{B}}{\mathbf{L}} = \frac{\overline{\mathbf{X}}_g(h,l)}{\mathbf{U}} = \frac{\mathbf{U}\mathbf{x}}{\mathbf{U}} = \mathbf{x}$.

only on l. Thus, the steps of centering of $\overline{\mathbf{X}}(l)$ and the estimation of $y(l)$ are independent of the steps on the $\mathbf{x}$ side (estimation of $\mathbf{L}(h,l)$, $\mathbf{B}(h,l)$ and $x(h,l)$). They find influence only in the centering of $\overline{\mathbf{X}}(h,l)$ of the next iteration. It is thus crucial that $\mathbf{x}$ and $\mathbf{y}$ are not reset at the beginning of each iteration. However, during development of our systems, we noticed that this element has an effect on trapping the convergence of this algorithm to a local minimum[8].

4.2.4 Algorithm summary

Algorithm 1 presents the adopted strategy for estimating the nuisance matrix $\mathbf{U}$ with the above developments. The estimation of $\mathbf{U}$ is performed using preferably an independent data corpus with several utterances per language. In the model training and the testing phases, the components $\mathbf{x}$ and $\mathbf{y}$ are estimated using the same algorithm (Algorithm 1) where the $\mathbf{U}$ matrix is fixed (not re-estimated) and only one iteration is performed.

Algorithm 1: Estimation algorithm for $\mathbf{U}$

```
Initialization:
    y(l) ← 0 for each language l ;
    x(h,l) ← 0 for each language l and utterance h;
    U ← random (U is initialized randomly with values in [0.0, 1.0) );
Estimate statistics: N(l), N(h,l), X(l), X(h,l)    (Eq. 4.8 and Eq. 4.9);
for i = 1 to nb_iterations do
    for all l do
        for all h do
            Center statistics: X̄(h,l)    (Eq. 4.10);
            Estimate L(h,l)^-1 and B(h,l)    (Eq. 4.11);
            Estimate x(h,l)    (Eq. 4.12);
        end
        Center statistics: X̄(l)    (Eq. 4.10);
        Estimate y(l)    (Eq. 4.12);
    end
    Estimate matrix U    (Eq. 4.13 and Eq. 4.14) ;
end
```

closed-form restriction

Opinion: Originally, JFA made discrete assignments of the frames to the Gaussians of the GMM, that is one frame is assigned to only one Gaussian, as described in (Kenny et al., 2005a). This limitation on the statistics step is required in order to find a closed-form solution to the optimization problem

[8] In fact, probably significantly better results were observed on one setup by resetting $\mathbf{x}$ and $\mathbf{y}$ at the end of the third and eighth iteration. For that setup, this accidental observation could not be outperformed by any other (structured) resetting strategy. This can be explained by the local-minimum trap.

that has to be solved (MAP criterion in the M-step of EM). We may wonder that JFA continues to work considerably well even if the assignments are changed to the Gaussians' posterior probabilities γ_g, so the frames are assigned to different Gaussians according to their "proximity". In our eyes, the closed-form updates are still completely valid, since we may probably see the assignments or sufficient statistics as being one single Gaussian, because the individual Gaussian component parts are stacked into one SV. If we see the problem from the abstract SV space, without its partitioning into the components, there is only one *Super-Gaussian* and the assignment is trivially discrete. For us, this is the justification for the algorithm's validity.

4.3 JFA variability compensation

In the following, we will see that JFA may be applied either on the feature level (removing the variability from each feature vector) or to the model level (compensating the model with a nuisance term). And both cases may be applied or only during training or only during testing or even at both stages. The resulting four possible combinations will be discussed in this section.

As basis, we are given $\mathbf{U}$, which we estimated by Algorithm 1. Let h_{tar} and h_{test} be the training and testing utterances respectively and l_{tar} and l_{test} are the corresponding languages.

4.3.1 Model compensation during training

In this paragraph we present the strategy using models in which the effect of variability is compensated. We are given $\mathbf{U}$, which was globally estimated using Algorithm 1. Now, with the exact same processing (except last step of estimating the $\mathbf{U}$ matrix), for all utterances h_{tar} belonging to a given language l_{tar}, the per-session $x_{h_{tar}}$ vectors and subsequently $y_{l_{tar}}$ get estimated whereas $y_{l_{tar}}$ is tied among all training utterances of this language l_{tar}:

$$m_{(h_{tar},l_{tar})} = m + \mathbf{D}y_{l_{tar}} + \mathbf{U}x_{h_{tar}} \tag{4.15}$$

Finally, the compensated model for language l_{tar} is simply given by the two terms that are common to all training utterances and which thus do not include the (per-utterance) variability term $\mathbf{U}x_{h_{tar}}$. Our model is thus $m + \mathbf{D}y_{l_{tar}}$. This is transformed (from the SV space) to a GMM model by splitting the SV to individual mean components and using Gaussian weights and covariances unchanged from the UBM. This is then stored as a compensated (cleaned-up) language model for language l_{tar}.

model cleanup

4.3.2 Feature compensation during training

Instead of removing the variability from the model, it may be subtracted from the feature vectors. The channel- and speaker variability-free features may be reused in any algorithm or classifier of various nature and complexity without any additional restriction. In this approach, JFA can be seen as new front-end generating cleaned-up features or simply as an additional step in the front-end processing (FE, Sect. 2.3).

feature cleanup

The channel variability term is removed from the feature vectors by subtracting the **Ux** term, weighted by the individual Gaussian occupation probability γ_g (posterior probability) of Eq. 2.7:

$$\hat{f} = f - \sum_{g=1}^{G} \gamma_g(f) \cdot \mathbf{U}_g \mathbf{x}_h \tag{4.16}$$

where $\mathbf{x}_h$ is estimated for the utterance h, which f belongs to and $\mathbf{U}_g$ designing the part of $\mathbf{U}$ that corresponds to Gaussian g (analogous to the sections of an SV resulting from stacking). This compensation is usually also done for the testing utterances in the very same way.

Since this approach is independent of the language of the utterance, it is a global, blind processing (we are just given $\mathbf{U}$). Thus $\mathbf{x}_h$ is estimated based on the UBM through the equation

$$m_{(h,l)} = \quad m \quad + \mathbf{U}\mathbf{x}_h \tag{4.17}$$

Opinion: Even if (Castaldo et al., 2007c) states that this approach is only possible based on the UBM, we think it is possible to apply feature-domain JFA to the training data by estimating $\mathbf{x}$ using the information about the utterance's language. Thus including the $\mathbf{D}\mathbf{y}_l$ term into Eq. 4.17. This should perhaps be attempted.

4.3.3 Model compensation during testing

During testing stage, the utterances may also be compensated for the speaker and channel variability represented by the **Ux** term.

Accurate way

Ideally, $x_{h_{test}}$ has to be estimated based on the hypothesized language's model using the equation

$$m_{(h_{test},l_{test})} = m + \mathbf{D}y_{l_{tar}} + \mathbf{U}x_{h_{test}} \tag{4.18}$$

Think of the left hand side representing the observed testing utterance and $m + \mathbf{D}y_{l_{tar}}$ comes from the (compensated) language model stored in the training step.

Under the assumption that the hypothesized language l_{tar} is the correct one ($l_{tar} = l_{test}$), we try to set the whole difference between the observed utterance

$m_{(h_{test},l_{test})}$ and the stored model $m + \mathbf{D}y_{l_{tar}}$ as channel variability — that is what Eq. 4.18 expresses. If this difference can be well mapped to the variability sub-space (opened by $\mathbf{U}$), the assumption about the language is likely to be true. If only a little of this difference can be described in the variability sub-space, it is more likely not the right language and the posterior Likelihood described hereafter will likely be poor.

In order to perform the verification test under the model-space approach, $\mathbf{U}x_{h_{test}}$ is added to the model's mean SV, which is converted back to a GMM and used to test the utterance against. We may express this under the form

$$f \quad | \quad m + \mathbf{D}y_{l_{tar}} + \mathbf{U}x_{h_{test}} \tag{4.19}$$

We thus compensate the stored language model with an utterance- (and language-) dependent variability term in order to test the observed features against. model perturbing

Compared to what happens in model-space compensation during training, we may also see this as follows: During training, the target model is transformed by removing the session component of the training data and during testing, the variability of the testing data is added. Thus in total, the training variability gets replaced by the testing variability.

As side-note, if we calculate the LLR against a UBM, the UBM itself has also to succumb to the same compensation.

Since the stored model of the hypothesized language is used (in Eq. 4.18), the whole process has to be restarted for every utterance–language combination, and thus for every single testing trial.

Approximation
In order to speed up calculations, the channel factors $x_{h_{test}}$ may also be estimated using only:

$$m_{(h_{test},l_{test})} = m + \mathbf{U}x_{h_{test}} \tag{4.20}$$

which is an approximation to Eq. 4.18. This $x_{h_{test}}$ is then used in the setup described by Eq. 4.19. The channel factors for the test utterance are thus obtained using the UBM instead of the hypothesized language's model. This is a good approximation (and speedup) to the accurate way described above (see also Vair et al., 2006; Castaldo et al., 2007c; Glembek et al., 2009).

This speeds up testing since the channel variability component $\mathbf{U}x_{h_{test}}$ has to be estimated only once for every testing utterance. In the case of LLR scoring, the UBM has also to be compensated only once per utterance. Running protocols with 23 languages (as in NIST LRE 2009), this contributes a considerable speedup! speedup

Opinion: In our eyes, (Vair et al., 2006) mix up or do not distinguish the independence of $\mathbf{x}_{(h,l)}$ (resp. $\mathbf{x}_h$) with respect to l and the approximation of estimating $\mathbf{x}_h$ on the UBM only instead of the full target model. The former being a precondition to the latter.

4.3.4 Feature compensation during testing

Even if the feature-domain compensation during training is a self-contained and theoretically sufficient process, feature-domain compensation can also be applied at testing stage.

feature cleanup

For this, the variability term $\mathbf{U}x_{h_{test}}$ is estimated using one of the methods mentioned in Sect. 4.3.3 (the accurate or the approximate way). But then, instead of appending this term to the stored model in order to match the channel distortion of the testing utterance, the troublesome variability is removed from the feature vectors. These cleaned features are then tested against the clean models stored during training stage. We may similarly express this under the form

$$f - \mathbf{U}\mathbf{x}_{\mathbf{h_{test}}} \quad | \quad m + \mathbf{D}y_{l_{tar}} \tag{4.21}$$

This approach is usually combined with model-space compensation during training. So this is called the *hybrid-domain* approach since it involves model domain (training) and feature domain (testing).

A small nuance can be found for feature-space compensation: The compensated features may undergo a final CMS step or not. Our original implementation included this step, but it has been disabled in an early stage, based (amongst others) on the results of Table 4.1 presented in the next section.

4.4 JFA results

This section investigates the effect JFA has in the context of Language Recognition under the diverse basic hypotheses and using the approaches presented in last sections.

First we analyze the possibilities that JFA offers when applied to the training and/or the testing phase. We then also investigate the effect of some tuning parameters. These include on one hand the rank of the matrix $\mathbf{U}$ to be used and on the other hand how JFA reacts when changing the sizes of the GMM models themselves.

4.4.1 Comparing JFA strategies

The objective of the following analyses is to compare the different strategies enumerated in Sect. 4.3. At the same time, we compare the JFA approach to JFA-less MAP adapted models.

— ◇ —

JFA vs. bare MAP

In a first part, Table 4.1 compares the baseline MAP system to systems featuring JFA. Similar comparisons have already been included into Table 3.2 of Sect. 3.3.3.

The same Table 4.1 compares also different testing-time JFA compensation strategies (and even no testing-JFA). They build all on model-space JFA during training. model-space training

Setup: The results presented in Table 4.1 are based on following two systems: Both systems are based on SDC features and have mixtures of 256 and 2048 Gaussians. One system is trained on the *train* part of CallFriend; The second system is trained on all three parts of CallFriend. The **U** matrix has a rank of 40 and performance is measured following the NIST LRE 2005 protocol using divSum-K normalization (with $K = 35$) and pooled EER (Sect. 2.6.2.1).

Table 4.1: *Comparing different testing-time JFA approaches and baseline MAP trained models, trained on CallFriend, tested on NIST LRE 2005, in %pooled EER*

JFA approach used for		256 Gaussians CallFriend		2048 G. CallFriend
training	testing	*-train*	*-all*	*-train*
MAP		18.48	19.51	18.31
model-space	*without JFA*	11.54	11.29	10.17
model-space	feature-space (hybrid), fast	10.46	–	8.27
model-space, no CMS	feature-space, fast, no CMS	10.09	10.17	7.52
model-space	model-space, accurate	10.17	9.64	7.49

From Table 4.1, we can read that JFA is most important during training. Performing JFA also on the testing data gives another reasonable 9% relative improvement. We observe also, similarly to what was already touched in Table 3.2 of Sect. 3.3.3 and will also be stated in Sect. 4.4.3, that JFA reveals its power on bigger models, the cost also drops from 10.17%$pooledEER$ for training-only JFA to 7.49%$pooledEER$ for training and testing stage JFA (-26.4% relative). The results seem to indicate that testing-time JFA helps particularly when the models are big.

The second part of analyses focuses on JFA applied to the feature domain during training. Table 4.2 shows a result into this direction, but this mitigated performance has to be taken with some precaution. Other researchers obtain results similar to model-domain JFA though (Vair et al., 2006). feature-space training

Opinion: This compensation in feature space has been tried and a cleaned set of utterances (as files containing feature vectors) have been generated. These have then been used to train a new UBM and then language models using MAP adaptation.
A first try yielded very poor performances. A later fresh attempt gave the results presented in Table 4.2, with very moderate performances, halfway between (JFA-less) EM models and model-space JFA (described in Sect. 4.3.1). We are still not completely convinced that either the tools provided by ALIZE/MISTRAL (Sect. 2.7) or our scripting solve this task fully correctly. To verify is for instance if the occupation probability γ has to be normalized to

sum up to 1.0 over all Gaussians (as it is implemented) or not. There is no hint about this in (Vair et al., 2006; Castaldo et al., 2007c).

Setup: The experiment with results in Table 4.2 is based on 256 Gaussian models trained on PLP type features of all CallFriend data. Measured using divSum-K normalization (with $K = 35$) and $min\text{-}C_{avg}$.

Table 4.2: *Comparison of feature- and model-space JFA approaches during training phase to baseline EM-ML trained models, in %$min\text{-}C_{avg}$*

EM-ML	28.91
feature-space FA	**18.56**
model-space FA	9.23

4.4.2 Variability matrix rank

We have seen that the speaker and channel variability is assumed to live in a sub-space of low dimension. This dimension corresponds to the rank of the **U** matrix. This section now analyzes the effective size of this sub-space. The results will allow to choose a good value for the **U** matrix rank.

— ◇ —

We present an analysis on the number of channel factors represented by the **U** matrix, which is its rank. The results of this analysis, shown in Table 4.3 indicate that we can gain some small enhancement by increasing the number of channels in/the rank of the **U** matrix. We can easily go up to around 100 channels, but required computation power increases considerably for just 4% relative improvement. We considering a rank of 40 as appropriate tradeoff between rather good performance and adequately light computation. Therefore, results of all other JFA systems presented in this work feature 40 compensation channels. Analog researches in the domain of speaker verification show a very similar behavior (Matrouf et al., 2007).

Setup: The systems used for obtaining the results of Table 4.3 have a slightly different setup than most other systems presented here (since this analysis has been done in an early stage of the development). They are based on LFCC-SDC features (instead of MFCCs, see Sect. 3.3.1). Further, only the *train* part of the CallFriend corpus was used for **U** matrix and language model estimation and the models are built with 521 components. The evaluation is done using divSum-K normalization (Sect. 2.5.2.2, $K = 35$) and pooled-EER performance measure (Sect. 2.6.2.1) on the NIST LRE 2005 data.

Table 4.3: *Factor Analysis performance on 512 Gaussian models using different channel variability matrix ranks*

rank	40	60	80	100	200
% EER	11.18	11.04	10.90	10.73	11.15

4.4.3 JFA effect on different model sizes

This section presents JFA results on different GMM-UBM model sizes. The objective of this series is to investigate how JFA behaves with respect to the GMM model size. Compared with the analysis in the previous section about **U**'s rank, it may lead to answering the question where computation power has to be invested: in augmenting the rank (JFA side) or the number of Gaussian components (GMM side). It also compares the JFA performances to the ones obtained on non-JFA, MAP adapted GMM-UBM models.

— ◇ —

While system development has been done on GMMs with 256 Gaussians, results are also presented featuring full systems using up to 2048 Gaussians. All the results are for 30 second segments, according to the NIST LRE 2005 primary condition. The GMM-UBM language models are obtained from the UBM with 10 iterations of MAP adaptation (Sect. 2.4.4), where only the mean values are updated (neither Gaussians' weights, nor variances are updated).

GMM-UBM results

While seeing the GMM-UBM system as baseline, it obtains 22.40 %$min\text{-}C_{avg}$ with 256 Gaussians and 19.44 %$min\text{-}C_{avg}$ with 2048, which represents about 13% relative gain for increasing the number of Gaussian components.

Table 4.4: *GMM-UBM and GMM-UBM-based JFA systems with respect to model size, NIST LRE 2005 task, 30 seconds; ratings in %$min\text{-}C_{avg}$.*

system	256	512	1024	2048	3072
GMM-UBM	22.40	21.25	20.07	19.44	–
JFA	8.57	8.38	6.99	**5.41**	5.75
gain, % relative	*-61*	*-60*	*-65*	*-72*	

JFA results

For the JFA system, the **U** matrix is set to have a rank of 40 (which is also the number of session factors). The matrix is iteratively estimated during 20 iterations using Algorithm 1. This JFA system performs at 8.57 %$min\text{-}C_{avg}$ using mixtures of 256 Gaussians. As Table 4.4 shows, the cost jumps to 5.41 %$min\text{-}C_{avg}$ with 2048 Gaussians, which is a far bigger improvement (37% relative) than observed for baseline GMM-UBM systems without JFA. For 2048 Gaussians, the JFA system outperforms the GMM-UBM one by 72% relative. While the capacity of GMM systems seems slowly to exhaust with about 512 Gaussians, JFA systems reveal their power on

increased model size (in the range of evaluated setups, the error rate reduction seems to be linear in the number of Gaussians added to the models). Observing this big performance impact of JFA over the baseline GMM-UBM system validates the profit of JFA for LR. Amongst others, we benefit from the fact that we have far more data to train each class than is the case in Speaker Recognition.

Other series including results on different model sizes can be found in Table 4.1 above and in Table 3.2 of Sect. 3.3.3, where also the reaction of JFA on different types of parameters is shown. The results they contain show similar trends.

Opinion: A quick attempt on a bigger model has been carried out, where the system ran with 3072 Gaussians. But the obtained result of 6.71 %$min\text{-}C_{avg}$ did not give additional improvements. This could be caused by the models getting too complex. The power of the UBM may get exhausted at such sizes, since there may be too many parameters to estimate.

4.5 Joint Factor Analysis conclusions

principle summary

The basic principle of JFA consists in decomposing each utterance into three parts (Eq. 4.5): A globally valid part (the general average, the UBM), a language-dependent part and the language-independent, per-utterance leftover. The UBM part is common to all training utterances. The language part is tied among all training utterances of the same language. And finally, the third part represents the various variabilities. It is supposed to live in a low-dimensional sub-space, and is thus constrained by a global low-rank matrix (named $\mathbf{U}$).

The (co-)variance of the observed data is separated into a part that is due to the language, $\mathbf{D}\mathbf{D}^{\mathrm{T}}$, and a part which is the variability of the speaker, channel etc., $\mathbf{U}\mathbf{U}^{\mathrm{T}}$.

Traditionally, JFA works in the mean Super-Vector space. With SDC 7-1-3-7 features and working with 2048 Gaussians, this yields vectors of 114 688 dimensions.

algorithm

After a thorough description of the decomposition, we presented a step-by-step algorithm, which may be implemented without further developments (Algorithm 1). The algorithm is mainly divided into three consecutive steps: First, general (posterior) statistics of the training data on the UBM have to be gathered. Then, MAP Point-Estimates of latent variables are calculated, based on these statistics and possible previous estimates. As a final step the compensation matrix $\mathbf{U}$ is estimated. It describes the sub-space, in which the variability factors live. These two last steps are usually iterated to allow the $\mathbf{U}$ matrix to converge.

usage strategies

We presented and investigated different ways to apply JFA in a running system, once the global $\mathbf{U}$ matrix has been estimated. The strategies differ along two axes: JFA may be applied in feature- or in model-space. Further, JFA may be used only during training or only during testing — or during both stages.

structural results

We arrive at following conclusions on JFA: The number of channels (rank

of **U**) is not very decisive. We chose a rank of 40, which we think to be a good tradeoff between performance and computation time. More crucial in contrast is the size of the GMM models, which are transformed to mean SVs. While system developments were mainly achieved with 256 Gaussians, full systems are run with typically 2048 Gaussians.

Further, our results tend to attest that the variabilities of diverse kinds (such as the speaker and the channel) can be caught at the same time in the same sub-space. The need for a complete JFA model separating these two variabilities can not be approved.

performance results

As overall conclusion, we can say that JFA helps, whatever strategy is adopted. However, JFA compensation during training-time helps most, whilst testing-time JFA enhances with growing model size. Reductions of a system's cost up to 72% relative can be achieved with JFA (comparing 2048 Gaussian models adapted through MAP with and without JFA). Observing this big performance impact of JFA over the baseline GMM-UBM system validates the profit of JFA for Joint Factor Analysis.

4.6 Similar methods

This section will briefly spend a few words on other dimensionality reduction methods that may be compared to JFA.

feature mapping

We may just indicate *feature mapping*, which uses the a priori information of a set of models trained in known (channel) conditions to map the feature vectors towards a channel independent feature space (Vair et al., 2006; Campbell et al., 2008).

NAP

Also, we touched already the Nuisance Attribute Projection (NAP), proposed by Campbell (Campbell et al., 2006b). NAP works in SV space and can thus be combined with SVMs, whereas JFA works on GMM-UBM and frame level and allows thus for instance joint modeling of different kinds of information.

4.6.1 Comparison between JFA and PCA

PCA

Principal Components Analysis (PCA) has first been introduced by Pearson in 1901 and is the oldest multivariate analysis technique. "*The objective of PCA is to perform dimensionality reduction while preserving as much of the randomness in the high-dimensional space as possible*" (Gutierrez-Osuna). For the randomness, we can think of the spread of the data points. Further, compared to JFA, PCA does not consider class separability, but only data reconstruction.

From a visual point of view, a full PCA (without reducing the number of dimensions) can be seen as centering the data distribution around the origin and rotating

it in such a way that the maximal variance directions become the coordinates.

— ◇ —

criteria

The hypothesis used in the PCA algorithm is to minimize the squared errors between the (current) model[9] and the training data. This error is also called *representation error* or *reconstruction error*. PCA thus uses a signal representation criterion (Gutierrez-Osuna).

PCA minimizing the reconstruction error, supposes that the data can be reconstructed without errors. On the other side, JFA assumes that some error will be made upon reconstruction. JFA further supposes that these errors follow a certain law. The error or residue (also called *noise source* is thus thought to be probabilistic, and the JFA can be seen as a probabilistic version of the PCA[10].

Supposing the reconstruction error following a single law, we are in the case of homogeneity of variance, also called *homoscedasticity*.

way of working

PCA finds a projection matrix, which maps the observed data points to a new (low-dimensional) space. JFA however finds a transformation (factor loadings, **U** matrix) and latent factors, which are able to reconstruct the observed data points under a certain error assumption. Thus JFA, in a certain manner, works in the opposite way compared to PCA.

In a visual manner, JFA stretches, rotates and shifts an initial standard normal distribution (the latent factors) in order to produce (describe) the input data distribution.

observed data point

In PCA, the observed data point is thought to come from a weighted combination of basis vectors (the weights being the values of the different dimensions, thus the low-dimensional representation, and the basis vectors originating from the eigenvectors). In JFA however, they are supposed to be drawn from a Gaussian distribution.

repartition on dimensions

In JFA (i.e. when i-vectors, Sect. 4.1.1, are used), all output dimensions have the same importance and potentially carry the same amount of information. But in the PCA case, the different output dimensions do not have the same importance. Since they are issued from eigenvectors or Singular Value Decomposition of the covariance matrix, the first dimension (the one which corresponds to the biggest eigenvalue) has more power in separating the classes than the other dimensions. The relative importance of the dimensions is probably linked to the magnitude of the eigenvalues.

The different repartition of the information on the dimensions between JFA and PCA can also be seen from its variances: In JFA, we suppose the variance to be the

[9] More accurately the data reconstructed by the model

[10] Many thanks to Pierre Michel Bousquet, LIA, for the related discussions.

identity (**I**), but in PCA, the variances are the eigenvalues.

— ◇ —

A more thorough description of PCA and its comparison to JFA can be found in (Tipping and Bishop, 1999a,b), as well as in (Roweis, 1998).

4.6.2 LDA

Linear Discriminant Analysis is based on SIR RONALD AYLMER FISHER's work (Fisher, 1936) in which he projects the data points into a (sub-)space in which the points of distinct classes are far apart, but at the same time keeping the data points of the same class in a close neighborhood. In contrast to PCA, LDA can only be used when class labels are available for the training data. introduction

— ◇ —

FISHER defined thus a criterion which, being maximized, maximizes the *between class scatter* and which minimizes the *within class scatter* using a ratio of covariances: Fisher criterion

$$\mathcal{F} = \frac{\sigma^2_{between}}{\sigma^2_{within}} \tag{4.22}$$

This criterion may also be interpreted as some kind of signal-to-noise ratio for the class labeling.

In the following, we very briefly show the LDA in different context, as when working with two or with more classes.

Two class problem — Equal covariance case

For this part, we assume the covariances of the two classes being the same (*homoscedastic*) and having full rank. detection using Fisher's discriminant

We will find a hyperplane with norm **p** that separates these two classes. If the points of each class are assumed to be normally distributed and if the class priors are equal, this hyperplane lies half-way between the two centroids (the mean vectors of the distribution). The class centroids being μ_i and the global mean μ being at half-way between the class-means, we have for the between class scatter[11]:

$$\sigma^2_{between} = E[(\mu_i - \mu)(\mu_i - \mu)^{\mathrm{T}}] = \frac{1}{2}2(\mu_0 - \mu)(\mu_0 - \mu)^{\mathrm{T}} \tag{4.23}$$

So, we write out $\sigma^2_{between}$ and for the (co-)variance (unique, since homoscedastic) of a class's points, we write Σ:

$$\mathcal{F} = \frac{\sigma^2_{between}}{\sigma^2_{within}} = \frac{(\mu_0 - \mu)(\mu_0 - \mu)^{\mathrm{T}}}{\Sigma} \tag{4.24}$$

[11] Using: $(\mu_0 - \mu) = \mu_0 - \frac{1}{2}(\mu_0 + \mu_1) = \frac{1}{2}(\mu_0 - \mu_1) = (\mu_1 - \mu)$

During detection, the test vector $\mathbf{x}$ will be projected onto the norm $\mathbf{p}$ of the hyperplane separating the two classes. The magnitude of this dot product will then be compared to some threshold b (or b is subtracted to obtain a score): $\mathbf{p} \cdot \mathbf{x} < b$.

So we are seeking for the norm $\mathbf{p}$ which maximizes this criterion, once projected:

$$\mathcal{F}(\mathbf{p}) = \frac{\mathbf{p}^T(\mu_0 - \mu)(\mu_0 - \mu)^T\mathbf{p}}{\mathbf{p}^T\Sigma\mathbf{p}} \tag{4.25}$$

We may show that $\mathbf{p}$ can directly be calculated as: $\mathbf{p} = \Sigma^{-1}(\mu_0 - \mu_1)$. For the decision threshold b, we find, what can be interpreted as the value of the projection of the point laying half-ways between both centroids: $b = \mathbf{p} \cdot (\mu_0 + \mu_1)/2$.

Two class problem — Different covariances
In the case of each class having a different covariance (heteroscedastic, i.e. Σ_0 and Σ_1), $\mathbf{p}$ becomes: $\mathbf{p} = (\Sigma_0 + \Sigma_1)^{-1}(\mu_0 - \mu_1)$. This is known as the *Quadratic Discriminant Analysis (QDA)* classifier, which compares a *likelihood ratio* to the threshold b:

$$LR = \frac{\frac{1}{\sqrt{2\pi|\Sigma_1|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}}{\frac{1}{\sqrt{2\pi|\Sigma_0|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)}} < b \tag{4.26}$$

An alternative to the QDA is the kernel trick, which can be applied to LDA as it is for SVM (Chapter 5). Its basic idea is to expand the test vector to a higher dimensional space so that a linear classifier in that space equals a non-linear one in the original space.

C **classes — Equal covariance case**
When extending LDA to C classes, the between class variability is the covariance of the class means μ_i:

$$\mathbf{B} = \frac{1}{|C|}\sum_{i \in C}(\mu_i - \mu)(\mu_i - \mu)^T \tag{4.27}$$

$$\mathcal{F}(\mathbf{p}) = \frac{\mathbf{p}^T\mathbf{B}\mathbf{p}}{\mathbf{p}^T\Sigma\mathbf{p}} \tag{4.28}$$

The separating hyperplane has norm $\mathbf{p}$, which is an eigenvector of $\Sigma^{-1}\mathbf{B}$, and a distance according to the corresponding eigenvalue.

derivation

In order to find the maximum, we may differentiate the criterion fraction (using the rule $\frac{\partial \mathbf{p}^T\mathbf{A}\mathbf{p}}{\partial \mathbf{p}} = \mathbf{p}^T(\mathbf{A}^T + \mathbf{A})$) and equate it with zero.

We may also simply put the denominator to the other equation side and introduce an identity term $\mathbf{W}\mathbf{W}^{-1}$ to the right hand side: $\mathbf{p}^T\mathbf{W}\mathbf{p}\mathcal{F} = \mathbf{p}^T(\mathbf{W}\mathbf{W}^{-1})\mathbf{B}\mathbf{p}$ and left-multiply both sides by $\mathbf{W}^{-1}(\mathbf{p}^T)^{-1}$, which has the effect of "shortening" the left $\mathbf{p}^T$ and $\mathbf{W}$, in order to leave over $\mathcal{F}\mathbf{p} = \mathbf{W}^{-1}\mathbf{B}\mathbf{p}$.

So the result of the criterion optimization (substituting back Σ for $\mathbf{W}$ for the present case) is a typical eigenvector ($\mathbf{p}$)/eigenvalue ($\mathcal{F}$) problem: $\mathbf{W}^{-1}\Sigma\mathbf{p} = \mathcal{F}\mathbf{p}$.

C classes — Different covariance
The C-class LDA with class-specific covariance matrices is called *Heteroscedastic Linear Discriminant Analysis (HLDA)* . It will not be handled here.

Detection/Classification
In multi-class context, classification is commonly done using a one-against-the-rest approach where the target class is opposed to a second macro-class, formed by all other classes. Very similar concepts of thresholding and using binary classifiers for the multi-class problem will be explained more in detail in Chapter 5.

— ◇ —

limits

While the limitation from theory is the number of output dimensions, being linked to the rank of the processed matrices, the practical limitation lies in the potentially huge processing power needed to compute the scatter (covariance) matrices if we have a lot of data points.

variants

Other variants of LDA are for instance the *Power Linear Discriminant Analysis (PLDA)* (Sakai et al., 2008) or also the *Non-parametric LDA* by Fukunaga(-Koontz), which removes the uni-modal Gaussian assumption so the projection preserve the data structure more closely.

La curiosité n'est que vanité. Le plus souvent, on ne veut savoir que pour en parler.

— Blaise PASCAL, Pensées de M. Pascal, 1670 [0]

Chapter 5

Support Vector Machines

Contents

origin

As opposed to Gaussian Mixture Models (GMMs), a Support Vector Machine (SVM) is a supervised two-class (binary) classification system (but it can also be applied to regression). SVMs are one of the algorithmic results of the Vapnik-Chervonenkis (VC) theory of statistical learning, which was developed by Vladimir Vapnik and Alexey Chervonenkis between 1960 and 1990.

discriminative modeling

GMMs (with or without JFA applied) are statistical modeling techniques, which try to mold at best the clusters (distributions) of points in multidimensional space. This is called generative modeling. Another kind of techniques is discriminative classification as Artificial Neural Networks (ANNs) or the SVMs presented in this chapter. They focus on tracing decision boundaries between the clusters instead of reproducing the distribution of the points. Discriminative techniques are thus working on the difficult part of space where most of the confusions occur. Whereas

[0] Pensées de M. Pascal sur la religion, et sur quelques autres sujets, Chapitre XXIV: Vanité de l'homme, § 9; 1670: p. 186 ; 1688: p. 123

generative techniques focus more on the "easier" part of space, where points have higher concentrations.

SVM +JFA

In speaker verification, SVM systems are roughly at the same performance as GMMs with JFA (Matrouf et al., 2011). Herein, we will try to verify how the combination of SVMs with JFA behaves in the context of Language Recognition. Further, the SVs on which JFA works could directly be used as SVs for an SVM classifier. The hypothesis behind the research related to this chapter is that the association between JFA and SVM should allow to benefit from the JFA decomposition (and variability compensation) power together with the SVM classification power. After an introduction to SVMs, this interaction between JFA and SVM will be analyzed in Sect. 5.4.

SVM structure

This chapter has also the objective to explore how an SVM has to be structured to obtain best results in Language Recognition. SVMs solve two-class problems. We thus analyze how these two classes (the positive and the negative one) have to be composed.

5.1 Linear SVM

In 1963, Vladimir Vapnik proposed the following linear classifier using an optimal hyperplane algorithm. In a general way, SVMs build on the structural risk minimization principle, defined by Vapnik in 1979.

super-vectors

Instead of feature vectors, SVMs operate with points in space, that are of higher dimension, usually called Super-Vectors. Since SVMs are binary classifiers, we need SVs of two classes for their training. In an SVM, this signifies that one class is discriminated against the other. So one class is said to be the positive and the other the negative one. The set of SVs representing the negative class is usually called the *blacklist*.

blacklist

An SVM is a discriminative classifier, meaning it focuses on the information distinguishing the two classes. It tries to represent or to model the boundary between them instead of trying to mold the distributions (clouds of points in space) of the two classes, as generative models like GMMs do.

linear SVM

The simplest SVM setup is the *linear SVM*, where the frontier between the positive and the negative points in space is a hyperplane in the same space. In the 2-dimensional space, this can be depicted as a straight line. This hyperplane can be expressed as:

separating hyperplane

$$\mathbf{w} \cdot x - b = 0 \tag{5.1}$$

where $\mathbf{w}$ is the normal (perpendicular) vector of the hyperplane and $\frac{b}{|\mathbf{w}|_2}$ is the offset of the hyperplane from the coordinate origin in that direction ($|\blacksquare|_2$ being the Euclidean norm of $\mathbf{w}$).

5.1.1 Maximum margin SVM

For determining the separating hyperplane between the two classes, only the few SVs that lay on this boundary are used. They are called *support vectors* and give the name to the SVM.

support vectors

— ◇ —

The SVs on each side of the boundary verify, for the positive and for the negative class respectively:

margin

$$\begin{aligned} \mathbf{w} \cdot x - b &= +1 \\ \mathbf{w} \cdot x - b &= -1 \end{aligned} \tag{5.2}$$

These define two hyperplanes that are parallel to the separating hyperplane and they span up a *margin* between the two classes (which does not contain any points) of width $\frac{2}{|\mathbf{w}|_2}$ (resulting from $\frac{b+1}{|\mathbf{w}|_2} - \frac{b-1}{|\mathbf{w}|_2}$).

In order to obtain a robust SVM, this margin has to be maximized. Since the width is $\frac{2}{|\mathbf{w}|_2}$, this involves minimizing $|\mathbf{w}|_2$. For mathematical ease, this can be replaced by the equivalent $\frac{1}{2}|\mathbf{w}|_2^2$, which is a quadratic programming optimization problem. The result of this optimization is the set of support vectors to be used and at the same time the normal vector of the separating hyperplane:

maximum margin

optimization

$$\mathbf{w} = \sum_i^n \lambda_i \mathbf{1}_\pm(x_i)\, x_i \quad ; \quad \mathbf{1}_\pm(x_i) = \begin{cases} +1 & \text{if } x_i \text{ is of positive class,} \\ -1 & \text{if } x_i \text{ is of negative class.} \end{cases} \tag{5.3}$$

where λ_i are the non-negative Lagrangian multipliers of the SVM optimization problem. We see also that the values of the class-indicating function $\mathbf{1}_\pm(\blacksquare)$ can be found again as the right hand side of Eq. 5.2. The SVs that do not lay on the margin's boundary (but are correctly classified by it) are not of any interest since their λ_i is zero.

And b is determined such that $\frac{b}{|\mathbf{w}|_2}$ is the offset of the hyperplane. This can be done solving Eq. 5.2 to b. It would be sufficient to calculate it for one support vector, but usually the mean over all support vectors S is taken:

$$b = \frac{1}{|S|} \sum_{i \in S} \mathbf{w} \cdot x_i - \mathbf{1}_\pm(x_i) \tag{5.4}$$

— ◇ —

The maximum margin SVM as generalized linear classifier maximizes at the same time the margin (geometrically), as it minimizes the empirical classification error.

5.1.2 Soft-margin SVM

In 1995 (Cortes and Vapnik, 1995) spun the (maximum margin) linear SVM further to allow training sets that are not entirely separable. This means to allow and to take into account SVs that lay inside the margin or even in the area of the other class.

slack variables (penalties)

Their proposed method not only takes into account the SVs on the boundaries of the margin, but also the SVs that fall into the false area by applying a penalty. They use (non-negative) slack variables ξ_i that will be non-zero for those penalizing SVs. The problem that has to be minimized (that was $\frac{1}{2}|\mathbf{w}|_2^2$ for the clean case) becomes:

$$\frac{1}{2}|\mathbf{w}|_2^2 + C\sum_i^n \xi_i \tag{5.5}$$

with C controlling the tradeoff between the penalty applied to ξ_i and the size of the margin. In this case, not only $|\mathbf{w}|_2$ has to be estimated, but also the different ξ_i.

non-linear

This defines a linear penalty with the drawback of outliers having a big impact. Non-linear penalty functions have also been developed, but they need bigger computational and mathematical efforts.

5.1.3 Testing with SVMs

We have seen how to train a Support Vector Machine. Now it is the turn of looking at the testing phase. More insight will be given in Sect. 5.3 since results will be presented at the same time for raw and for JFA-based GMM-SVMs.

— ◇ —

Test utterance points (SVs) are also mapped to the higher-dimensional domain, as training SVs are. They originate from GMMs of UBM-based MAP adaptation towards the testing utterance's data.

score

This testing SV of unknown class is then classified by the SVM according to the trained normal vector $\mathbf{w}$ and the offset b:

$$s = \mathbf{w} \cdot x - b \tag{5.6}$$

If the score s is positive, the testing utterance can be classified into the positive class, otherwise into the negative one.

multi-class

Since LR is a multi-class problem and SVMs are binary classifiers, there are two possibilities how they can be applied: 1) testing one class (positive) against all other ones (negative set) or 2) testing pairs of classes two-by-two. Here, we chose the former, since it matches well our detection task, which consists in detecting one class against the others.

LLR interpretation

The scores produced by the SVM systems are Log-Likelihood Ratio like, so the same score processing procedure as for the other GMM based systems may also be applied, although the effect of score normalization is far less crucial.

— ◇ —

System setups and results for SVMs will be presented in next section, together with results of SVM-UBM-JFA systems.

Opinion: By defining the margin to have a width of $\frac{2}{|\mathbf{w}|_2}$, we see that the value of a classification score becomes meaningful: It is well known that the sign of the score indicates the class to decide (Eq. 5.6), since the offset b plays the role of threshold. Additionally, if the absolute value of the score is bigger than 1, then the SV could be classified clearly, since it falls outside the margin (Eq. 5.2). If its absolute value is inferior to 1, then it falls into the margin. But this easy interpretation will probably not hold any more for soft-margin SVMs. score interpretation

5.2 Non-linear SVM

In 1964, Aizerman, Braverman and Rozonoer (Aizermann et al., 1964) found the *kernel trick* for machine learning algorithms. Its effect is to put the SVs to a high-dimensional space in order to apply linear SVMs. This corresponds to applying a non-linear SVM in the original input space.

There is an infinite number of possible kernels that may be applied. The design or architecture of kernels may be a field of research of its own (Moschitti, 2010a,b).

5.2.1 Kernel trick

The kernel trick maps each (multidimensional) point into a higher-dimensional space, which might even be infinite. The idea is that classes with a complex (non-linear) structure in the original space can be described with linear classifiers in the mapped high-dimensional space.

The *trick* consists in the fact that, if an algorithm depended on the inner product of two input vectors, then it depends also only on the inner product of the two vectors mapped to this higher-dimensional space. Thus, this mapped space does not need to be explicitly known. This allows also this space to be potentially of infinite dimension. Further, this non-explicitness makes it computationally easier or even feasible.

A *kernel* is thus a function, which maps a combination of two input vectors to a kernel

real value:

$$\kappa : X \times X \to \mathbb{R} \tag{5.7}$$

where an inner product space exists with:

$$\kappa(x, x_i) = \langle \varphi(x), \varphi(x_i) \rangle \tag{5.8}$$

where $\varphi(\blacksquare)$ is the mapping of an input vector to the high-dimensional inner product space.

SVM kernel

Only in 1992, Boser, Guyon and Vapnik (Boser et al., 1992) applied the kernel trick to the linear maximum-margin hyperplane SVMs in order to build nonlinear classifiers. The maximum margin hyperplane classifier, which is linear in the kernel space appears thus in the original input space as a nonlinear classifier.

conceptual proof

Compared to the linear SVM, the kernel function replaces the dot products (e.g. Eq. 5.1). This makes the separation of complex classes easier and it is still linear (in the mapped space). We can also use the definition of a hyperplane to feel that the mapping does not need to be defined: A hyperplane is defined (by Eq. 5.1) as the set of points (SVs), for which their inner product with a fixed vector (in occurrence $\mathbf{w}$) is constant. For a higher-dimensional space, this is the same and it is sufficient to be able to calculate inner products in that space in order to define hyperplanes therein.

5.2.2 Kernel types

mean super-vector

In our GMM-UBM context, the Super-Vectors consist of all means μ_g of a GMM stacked to form a vector of considerable dimensionality: $G \cdot d$, with G being the number of Gaussians in the model and d being the dimensionality of the feature vectors. For example with 2048 Gaussians and a feature vector dimensionality d of 56 (for typical SDCs), this results in SVs of dimension 114 688.

kernel

In (Campbell et al., 2006a), a probabilistic distance kernel that computes a distance between GMMs was proposed. This distance is well suited for a Support Vector Machine classifier. Let $\mathcal{X}_l$ and $\mathcal{X}_k$ be two sequences of speech data corresponding to the languages l and k. The kernel formulation we use is given by:

$$\kappa(\mathcal{X}_l, \mathcal{X}_k) = \sum_{g=1}^{G} \left\langle \sqrt{\alpha_g} \Sigma_g^{-\frac{1}{2}} \mu_g^l \,,\, \sqrt{\alpha_g} \Sigma_g^{-\frac{1}{2}} \mu_g^k \right\rangle \tag{5.9}$$

where α_g, μ_g^l and Σ_g are the weight, the mean and the covariance matrix of the g^{th} Gaussian in the GMM. This kernel is valid when only the means of the GMM models are varying. This is true since the weights and covariances are taken from the UBM. All the SVM experiments presented hereafter use the linear kernel shown in Eq. 5.9. We will stick to this kernel for our SVM experiments.

other types of kernels

Other kernels may be divided into several classes. They may for instance be categorized according to the following basic structures:

Linear kernels: $\kappa(x, y) = \langle x, y \rangle$

Polynomial kernels: $\kappa(x, y) = \langle x, y \rangle^d$

Radial Basis Function (RBF) kernel: $\kappa(x, y) = e^{-\frac{|x-y|^2}{2\sigma^2}}$

Further types of kernels include for instance sigmoidal or inverse multi-quadratic ones. A more exhaustive research on SVMs and different kernel types in the SR context can be found in the thesis (Louradour, 2007).

5.3 SVM structure

training sets

For training SVMs, we need a set of positive SVs and a set of negative ones, the blacklist. There is two possibilities for the positive SVs:

one: One unique SV is obtained for a given language to train. For this, the GMM language model of a previous MAP or JFA experiment is converted to a SV using not only the Gaussians' means, but also their weights and covariances (cf. Eq. 5.9):

$$x_g = \sqrt{\alpha_g}\Sigma_g^{-\frac{1}{2}}\mu_g \tag{5.10}$$

These per-Gaussian vectors are then stacked to a SV.

multi: A multitude of SVs are used. For each training utterance of the current language, the UBM is adapted by MAP with a regulation factor τ (of 14.0 in our case) towards the utterance's data. At this stage, a JFA step may be inserted. These per-utterance GMMs are then transformed using Eq. 5.10. Thus, one SV for each utterance is obtained.

So all utterances of the training language are used for the positive set, either as one centroid SV or as distinct (utterance-wise) SVs.

blacklists

For a given training language, the negative labeled examples are recordings belonging to all the other languages. Also for this blacklist, different configurations can be imagined. We define three different blacklist compositions:

Set A is composed of one SV for each of the non-target languages — similar to the one-SV setup for the positive case. This type of blacklist thus contains $|L| - 1$ SVs. Each SV is obtained by Eq. 5.10.

Set B is composed of one SV for each training utterance of the non-target languages. As example for the NIST LRE 2005 setup, the training corpus comprises all three CallFriend parts, and thus these blacklists contain 960 or 1080 SVs, issued from utterances containing about 12 minutes of speech each.

Set C is the same as *set B*, augmented by additional (non-target) SVs. In the NIST LRE 2005 context, this adds all SVs of non-target language utterances of NIST LRE 2003 evaluation data, which contains utterances of 3, 10 and 30 seconds.These blacklists count thus between 2640 and 3240 SVs.

Fig. 5.1 depicts on the left hand side the two enumerated possibilities for the positive SVs (targets) and on the right the three blacklist sets defined above.

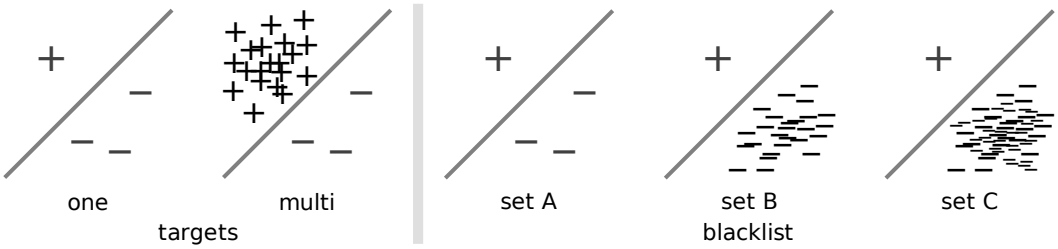


Figure 5.1: *Illustration of different setups for the targets (+, one vs. multi) and for the blacklist (−, Sets A, B, C) with the line symbolizing the SVM hyperplane*

combinations

Exploring these combinations is motivated by the fact that estimating a separating hyperplane using only one point on the target side seems amendable and for the impostor side, by using more SVs, we expect better generality (similar to a UBM adducting robustness).

5.3.1 Results

training sets composition

Table 5.1 shows the results of the 256 Gaussian SVM systems with all different target SVs and blacklist combinations. All SVs are the mean Super-Vectors obtained of JFA compensated GMMs. The systems are tested on NIST LRE 2005 and the scores are normalized with divSum-K (Sect. 2.5.2.2, $K = 35$). As comparison, one test has been conducted on 2048 Gaussians: For one target SV and blacklist set C, the pooled EER is at 5.61 %.

Table 5.1: *SVM-UBM-JFA results for 256 Gaussians, in % pooled EER*

blacklist	one target SV	multiple target SVs
set A	8.16	8.74
set B	7.32	7.88
set C	7.12	6.93

discussion

The results primarily indicate that the blacklist should be composed of as different SVs as possible (*set C*). In addition, we obtain better results using multiple target SVs, given we also have enough impostor SVs. But SVM systems are more difficult to tune, since there is more parameters that are quite sensitive, as for instance the soft-margin penalty controlling parameter C, Eq. 5.5 in Sect. 5.1.2 (Hsu et al., 2003), and the profit may be lost.

5.4 JFA-based SVMs

The JFA decomposition algorithm leads to compensated SVs defining GMM models. Since in this SV domain, each utterance is represented by a single (big, fixed-size) vector (denoted M or $\mathbf{m}$ in Chapter 4), these SVs can directly be used in an SVM classifier to evaluate data Likelihood in the usual way (Campbell et al., 2006a). But with the kernel we use (Eq. 5.10 and Eq. 5.9), this is only possible taking over the the GMM weights and covariance from the UBM (since SVs issued from JFA are purely the means stacked, whereas the kernel also uses the covariance and the weights). This association between the JFA and SVM allows one to benefit from the JFA decomposition power and SVM's (discriminative) classification power.

By using Algorithm 1, the JFA modeling leads to SVs that contain both additive terms: the language component and the channel component. All SVs are compensated by discarding the channel component. The retained SVs are of the form $m + \mathbf{Dy}$. The resulting compensated SVs are directly used in the SVM classifier described here. All the SVs used by the SVM are compensated: the SVs corresponding to the target language, the set of negative examples (blacklist) used in training SVMs, and to the SVs of the test utterances.

5.4.1 Results

GMM to SVM comparison

Further, we compare SVM systems to GMM-UBM systems, as well with as without JFA applied. For these SVM systems, every training utterance of the target language is represented by an associated positive SV (multi-target). The blacklist is using blacklist *set C*.

In Table 5.2, we show the results of 256 Gaussian and 2048 Gaussian SVM systems using mean SVs obtained without and with JFA. These are compared to MAP adapted GMM-UBM systems with a corresponding setup. All systems build on MFCC-SDC features and are run on NIST LRE 2005 with *llkMax0* score normalization (Sect. 7.2.1) and expressed as *min-*C_{avg}.

Table 5.2: *Results for SVM systems (one-target and blacklist set C setup) without and with JFA compared to the generative GMM systems. NIST LRE-2005, in %min-*C_{avg}

system	256 G	2048 G
GMM-UBM	22.40	19.44
GMM-UBM-JFA	**8.57**	**5.41**
SVM-UBM	**11.55**	—
SVM-UBM-JFA	8.91	7.21

The results primarily indicate that the SVM system without JFA largely outperforms the GMM-UBM system, but the JFA-based systems are at about the same

level with or without JFA for a model size of 256 Gaussians and do not improve that well for 2048 Gaussians. This could be due to the fact that the tuning of SVMs is more delicate.

— ◇ —

discussion

In this section, we investigated the use of the JFA method in the context of SVMs. For the JFA-based SVM system evaluated on the NIST LRE 2005 task, the result is at 8.91 %*min-*C_{avg} for 256 Gaussians, which is a relative gain of 60% over the GMM-UBM system. Using SVs coming from 2048 Gaussian models, the cost of the SVM system is 7.21 %*min-*C_{avg} (a gain of 63% relative over bare GMMs). We observed that the SVM systems with JFA do not show any improvement over generative GMMs with JFA, but the JFA approach proofs its usefulness also in this SVM context (−23% relative for 256 Gaussian).

Similar observations can still be made while changing the type of front-end parameters from MFCC-SDCs (dimension 56) to PLPs (dimension 39), as it is shown in Table 5.3.

Table 5.3: *Results for different system setups, based on PLP-12 features with/without JFA and with/without SVM. NIST LRE-2005, llkMax0 score normalization, in %min-*C_{avg}

base system	no-SVM	SVM setup		
		blacklist	one target SV	multiple target SV
EM-ML	28.97	—	—	—
MAP	28.24	set A	34.08	31.22
		set B	20.93	20.05
		set C	20.17	14.95
JFA	8.85	set A	15.25	11.60
		set B	11.89	10.42
		set C	10.78	9.03

It is to note, that the training data should be as similar to the testing data as possible. This is a general rule for pattern classification, but it has shown of particular importance on a quite fine level for SVMs, where (in the context of LR) this holds particularly for the speech utterance duration[1].

Opinion: Perhaps, it should be tried to obtain training SVs through MAP adaptation not from the UBM, but from the language model, which was trained on all training utterances of that language. We thus think of a language-model-based instead of UBM-based MAP. Perhaps, this "triangle" adaptation would be the same as a UBM-based MAP with a different regulation factor regulation factor. Perhaps, the performances could even be slightly worse,

[1]Some NIST LRE participants fork off specific handling for the three nominal durations (3, 10, 30 s). Other append an additional parameter containing the segment's duration as workaround.

if we remember of the SVMs being sensitive to the utterance duration (see just above), since the testing SVs will still be MAP'ed from the UBM, which would induce a discrepancy from the training utterances' adaptation.

5.5 SVM conclusion

In a short constructive approach, we presented the working of SVMs, starting with linear SVMs and subsequently applied the kernel trick to end up with non-linear SVMs (in the input space). The scores output by an SVM indicate the location (distance) of the testing SV with respect and relative to the margin. Different SVM configurations, as well on the target (positive) as also on the blacklist (negative) side can be imagined. In the LR context, these are crucially different, since we have a lot of (training) utterances for every class.

We observe that exploding the per-language centroids into individual utterance-wise SVs is, at least for the blacklist, of some benefit. Further, we analyzed the effect of SVMs in the JFA context. While the SVM approach is of considerable use (-48% relative) on bare GMM-UBM models, their advantage in the JFA context can not be affirmed. The direct usage of JFA's SVs as inputs to the SVM may justify this procedure for its simplicity avoiding the need to retransform the SVs to GMM models for scoring.

The novelty about combining SVMs (directly) with JFA compensated models (represented by SVs), which we presented in this chapter were among the first in their kind in the domains of Speaker Recognition and Language Recognition. novelty

Let's also recall, that the SVM technique is not limited to scoring, but may also be used in more complex processing, as shown in Sect. 3.4.2 about GMM-SVM pushback.

La nature a des infinis mystérieux, une puissance d'imagination. Elle se manifeste en variant toujours ses productions.

— Eugéne Henri Paul GAUGUIN,
'Oviri, écrits d'un sauvage, 1889 [0]

Chapter 6

Variability Between Databases

Contents

data sources

In 2009, it was the first time that NIST's Language Recognition Evaluation (LRE) (NIST LRE, 2009) included data of two rather different channel categories: The classical *Conversational Telephone Speech (CTS)* and the major part coming from

[0] Chapitre premier, Huysmans et Redon (fin 1889, Extraits), p. 60.
English: *Nature has mysterious infinities and imaginative power. It is always varying the productions it offers to us.* The Writings of a Savage, translated by Eleanor Levieux, edited by Daniel Guérin, 1978; Part one, Huysmans and Redon (end of 1889), p. 39

telephone bandwidth parts of *Voice of America (VOA)* radio broadcasts. This LRE disposed of a total of 23 target languages, of which some were very similar, as Ukrainian and Russian or Hindi and Urdu.

section's objective

As pointed out in Chapter 4, JFA systems presented here will show a very important gain over traditional MAP adaptation systems. These JFA systems require a projection matrix (**U**) that keeps track of the session and channel variabilities observed in the training data. The NIST 2009 evaluation setup contains two rather distinct channel categories (CTS and VOA). The objective of this part of study is to analyze the impact of working in an environment containing such hybrid data sources. In this section, we will research how these two categories may be handled separately up to a certain point, under the hypothesis that a unique system is not able to cope conveniently with data of very different kinds, as is the case in the environment pointed out.

incomplete data

Further, the considerable number of languages to be handled in the NIST LRE 2009, which is 23, leads to the fact that there are some languages where there is only data of one channel category. As well the training as also the testing stage are hit by the possibly strong bias introduced by this fact. This particular aspect of LRE 2009 is discussed in Sect. 6.1 and an unbiased way of evaluation using a subset of 8 languages is proposed.

— ◇ —

system setup strategies

In regard to system setup, different organizations are conceivable. One possibility is to build one unique system including data from both categories pooled together. But initially, it may be a better idea to cope with these two categories separately and then putting them together at some later stage. For this, there are several possible setups: We may estimate one JFA session variability matrix (**U**) using both, CTS and VOA data and then train category dependent models using this common compensation matrix. Another strategy could be to estimate session variability matrices separately and train the language models using data pooled over both channel categories. This case has not been researched because it does not seem optimal to use some training material of a category which the compensation was not designed for[1]. Finally, we may estimate separate matrices and estimate the models using only data of the same category as used for the matrix and then put the systems together on the score level.

This section will thus have a look at systems with a global approach putting all data together and systems handling these categories separately. These systems will be evaluated following the NIST LRE 2009 protocol (Sect. 2.6.3.2) with its 23 languages, as well as on a common subset of 8 languages since we do not have training and testing data for both channel categories in all 23 languages (more on this problem can be found in Sect. 6.1). Each system will also be evaluated on the CTS segments of these 23 and 8 language evaluation sets only and on the VOA

[1] If the session variability was really independent of the channel category, this could work. But in this case, a separation into two parallel systems would not be needed.

utterances only.

For the different JFA systems we analyze in the following sections, the channel compensation matrices are obtained using the unique UBM and are set to have a rank of 40 (the number of session factors and the dimensionality of $\mathbf{x}$; they are iteratively estimated during 14 iterations).

For training each language model, statistics over training data against the UBM are collected and $\mathbf{x}$ and $\mathbf{y}$ are estimated. The model, being the $m + \mathbf{Dy}$ part of the JFA formula (Eq. 4.5), gets then fashioned in one step.

6.1 NIST LRE 2009 particularity

The data provided by NIST in 2009 for system development and training is a heterogeneous composition of CTS and VOA data. CTS data comes from historic recordings from the CSLU of the OHSU.

incomplete training data

CTS training data is available only for 11 of the 23 languages of NIST LRE 2009: Spanish, English, Korean, Mandarin, Hindi, Indian English, Cantonese, French, Persian (Farsi), Russian and Vietnamese. VOA data comes from broadcast recordings transmitted over satellite or over the internet. This source comprises 22 of the 23 languages, where the missing language is Indian English. The fact that there is not training data of both category conditions for every language poses some troubles. If we follow an approach where both data parts are pooled together, some language models will have been trained exclusively on one channel category (the source of the recordings). On the other hand, if the two sets are handled separately, we can not estimate two sets of 23 single-source models. To be still able to obtain a full set of 23 language models, we will, where it is necessary, use recordings of the other category. Since otherwise, there would be no data for training some of the models.

incomplete testing data

Similarly, on the testing segments side, some category–language combinations are missing. This presents even a more consequent problem. CTS test utterances are available for 10 languages only: Cantonese, English, Hindi, Indian English, Korean, Mandarin, Persian, Russian, Urdu and Vietnamese. Of these, 9 correspond to the available CTS training data languages, the exception being Urdu. VOA test utterances are present for the same 22 languages as in the training data. When analyzing systems separately on the CTS and on the VOA testing utterances, we will end up comparing system performances on a different number of languages. But this is not a big problem since the evaluation officially is a *detection* task, which answers binary questions, in contrast to an identification task that has to select one language out of a set. Being in a detection context, the results remain comparable and the average expected cost of a detection system delivering random decisions is 50%, independently of the number of classes in the evaluation.

overview

Table 6.1 shows the above mentioned facts and gives some statistics for the dif-

ferent languages, as well as the indication about the number of languages available for each set.

Table 6.1: *Data availability for the different languages in the training and the testing sets, number of utterances and total duration, the 8 languages with full coverage are highlighted*

	training data				testing data	
language	CTS		VOA		CTS	VOA
Amharic			400	25.38 h		398
Bosnian			400	10.03 h		355
Cantonese	**39**	**3.04 h**	**388**	**2.98 h**	**316**	**62**
Creole Haitian			400	14.04 h		323
Croatian			400	4.58 h		376
Dari			400	13.07 h		389
English Amer.	**1906**	**58.90 h**	**400**	**21.80 h**	**522**	**374**
English Indian	474	4.88 h	–	–	574	–
Farsi=Persian	**159**	**29.75 h**	**400**	**19.97 h**	**52**	**338**
French	119	27.63 h	400	20.30 h	–	395
Georgian			400	6.43 h		399
Hausa			400	18.84 h		389
Hindi	**403**	**25.53 h**	**386**	**11.40 h**	**270**	**397**
Korean	**786**	**28.81 h**	**350**	**5.37 h**	**145**	**318**
Mandarin	**2253**	**64.84 h**	**400**	**15.32 h**	**625**	**390**
Pashto			400	17.36 h		395
Portuguese			400	17.11 h		397
Russian	**40**	**2.70 h**	**361**	**27.82 h**	**257**	**254**
Spanish	1535	27.85 h	347	63.78 h	–	385
Turkish			400	7.78 h		394
Ukrainian			400	8.20 h		388
Urdu	–	–	400	23.85 h	32	347
Vietnamese	**156**		**400**	**13.36 h**	**288**	**27**
total	7870	336.94 h	8632	332.84 h	3081	7490
number of languages	11		22		10	22

6.1.1 8 language task

There are only 8 languages that are available in both data sources, in CTS and in VOA, and that are contained as well in the training as in the testing data sets. Consequently, results will also be presented for the subset of these 8 common languages in order to be able to take conclusions on a somehow cleaner setup (unbiased with this respect).

languages and datasets

The languages of the 8-*language task* are: Cantonese, English, Hindi, Korean, Mandarin, Persian, Russian and Vietnamese. Table 6.2 gives some overall statistics

about this 8-language task. The datasets involved in this task are the *CTS* and the *VOA* datasets, as defined in Sect. 2.6.1.1.

Table 6.2: *Number of languages and number of test utterances in the 8-language task for all 30 s test segments and on a per-condition basis*

NIST LRE 2009 — 8 language task, closed-set 30s tests			
	all tests	CTS tests only	VOA tests only
number of testing languages	8	8	8
total number of test files	4635	2475	2160
files per language	315–1015	52–625	27–397

6.1.2 23 language task

The 23 language task represents the same systems under the official NIST LRE 2009 condition, where there are language–condition combinations that are missing either in the training or in the testing set. It also shows to which extent the systems are robust enough to recognize certain languages on a channel condition for which no training data is available.

As described in Sect. 2.6.1.2, NIST's LRE 2009 evaluation concerns 23 languages. Hence, we need some data to train each of the 23 language models in the case we focus on one channel category only. For this, we extend the clean data sets (defined in Sect. 2.6.1.1) used for training as follows: datasets

***CTS*+ extended data set**
The *CTS*+ data set comprises the *CTS* dataset (Sect. 2.6.1.1) for its 11 languages, extended by the *VOA* parts for the other 12 languages.

***VOA*+ extended data set**
This set analogously contains the *VOA* dataset for the 22 languages contained in it plus the *CTS* part for Indian English, which is missing VOA data.

Table 6.3 gives some statistics of the testing data for this 23-language task.We observe for instance that CTS tests exist only for 10 languages, whereas there are 22 for the VOA test utterances. For this testing data, there is nothing we can do. Only for the training data, we can add utterances from the other category to cover all languages.

6.2 Baseline MAP results

Results in subsequent sections describing JFA systems will be compared to MAP adapted GMM-UBM systems (Sect. 2.4.4) without JFA, whose results are given in

Table 6.3: *Number of languages and number of test utterances in the 23-language task*

NIST LRE 2009 — 23 language task, closed-set 30s tests			
	all tests	CTS tests only	VOA tests only
number of testing languages	23	10	22
total number of test files	10571	3081	7490
files per language	315–1015	32–625	27–399

this section. All models in this and subsequent sections are mixtures of 2048 Gaussians. The GMM-UBM language models are obtained by simple MAP adaptation (still with a factor τ of 14.0), where only the mean values are changed (neither Gaussians' weights nor variances are adapted). The remaining system setups match those of the subsequent JFA systems and can thus directly be compared.

6.2.1 Evaluation on 8 languages

While seeing the GMM-UBM system as baseline, it obtains, evaluated on the 8-languagetask, 18.21 %$min\text{-}C_{avg}$ when trained on *CTS* data only and 18.31 %$min\text{-}C_{avg}$ with *VOA* data only. When using all data for training, its mean average cost is at 16.91 %$min\text{-}C_{avg}$. Table 6.4 indicates the MAP performances of the single channel category evaluations as well.

Table 6.4: *MAP adapted GMM-UBM systems evaluated using only the 8 mutual languages on all test segments and on a per-category basis, in %$min\text{-}C_{avg}$, the highlighted values indicate results with matching category*

NIST LRE 2009 — 8 language task, closed-set 30s tests			
data for MAP model estimation	all tests	CTS tests only	VOA tests only
CTS &VOA	**16.91**	19.69	14.27
CTS	18.21	**19.64**	19.27
VOA	18.31	24.62	**12.27**

Opinion: The MAP systems presented here did undergo just one MAP adaptation iteration. Either the MAP regulation factor τ should be tuned or several iterations should be done (which should have a similar effect). For 10 iterations for instance, the 8-language VOA system would have a performance of 15.09 %$min\text{-}C_{avg}$ over all test utterances, 21.91 %$min\text{-}C_{avg}$ on cross-condition and 9.36 %$min\text{-}C_{avg}$ on the matching VOA only tests. Even if this signifies an enhancement of 11 to 23% relative, these results are still far from JFA based results, as will be shown later on.

6.2.2 Evaluation on 23 languages

Also on the 23-language task, the performances shown in Table 6.5 are not that good and thus there is not so a big difference between the different evaluation parts, but slightly better results can still be observed on testing utterances matching the training data.

Table 6.5: *MAP results of the 23 language task on all test segments and on a per-category basis, in %min-C_{avg}*

NIST LRE 2009 — 23 language task, closed-set 30s tests			
data for MAP model estimation	all tests	CTS tests only	VOA tests only
CTS &VOA	**20.60**	22.41	21.21
CTS+	23.54	**21.50**	27.89
VOA+	21.62	35.48	**17.30**

6.3 Pure single category JFA systems

In order to analyze the impact of the channel category in the JFA context, we build two completely distinct parallel systems. One system is trained exclusively on CTS data, the other exclusively on VOA data. The used datasets are thus truly distinct. This allows not only to investigate the category problem, but it serves also as basis for subsequent approaches. We define the two "pure-category" systems as follows:

Pure CTS system
The pure CTS system uses only standard phone data (the *CTS* data set, Sect. 2.6.1.1) for estimating the session compensation matrix **U**. The language models are then also estimated using this pure CTS session compensation matrix and the *CTS* training data set.

Pure VOA system
The pure VOA system estimates a session compensation matrix on VOA data only (*VOA* data set), which are phone calls transmitted over broadcast. This matrix serves then for estimating language models using this same *VOA* data set.

23 language context

Due to the reasons given in Sect. 6.1, systems working on the whole set of 23 languages, require some foreign data, namely the *CTS*+ and *VOA*+ datasets defined in Sect. 6.1.2. We will see that the **U** matrices will still be trained on the pure data, but these extended datasets are required for model training.

matching and cross tests

To compare now their performance, we define tests of *matching category* as CTS test segments tested on the CTS system or VOA segments tested using the VOA system. Similarly, *cross category* (or channel category mismatch) tests are trials where the test segment category does not match the channel category on which the system

was trained, thus testing CTS on VOA and VOA on CTS. This allows to evaluate the impact of the channel categories.

6.3.1 Evaluation on 8 languages

The results for the clean 8-language task are shown in Table 6.6. As expected, we see a striking difference between evaluations with matching category and those on cross category: If the category of the testing utterances match the one used in training, we observe excellent 2.71 %*min*-C_{avg} for CTS and 1.90 %*min*-C_{avg} for VOA. If the category does not match, the results degrade to 10.15 %*min*-C_{avg} and 4.28 % *min*-C_{avg} respectively. This means, when changing the condition of the training data (while still testing on the very same utterances), the systems yield 3.8 and 2.3 times more errors! When looking at the pure VOA system, it presents a striking difference between matching tests with 1.90 %*min*-C_{avg} and cross category tests with a cost as high as 10.15 %*min*-C_{avg}! On the other hand, the CTS **U** matrix seems to be much more robust against big changes in the channel condition (the channel category). This is perhaps a result of the CTS data being of a little lower quality or being more heterogeneous than the phone-over-broadcast VOA data.

Table 6.6: *UBM-based JFA systems trained on single channel categories, evaluated on all test segments and on a per-category basis, matching category tests are highlighted, in %*min*-C_{avg}*

NIST LRE 2009 — 8 language task				
data for estimating		closed-set 30s tests		
U matrix	models	all tests	CTS only	VOA only
pure CTS	CTS	3.05	**2.71**	4.28
pure VOA	VOA	6.78	10.15	**1.90**

We observe that both training data parts (*CTS* and *VOA*) have about the same total amount of training data, but for *CTS* distributed on only half the number of languages, thus benefiting in average from twice as much data each (30.6 *h* vs. 15.1 *h*). The VOA tests seem to be easier. By analyzing the matching category evaluation, VOA gives a far better performance, even if the models are trained on less data in average.

6.3.2 Evaluation on 23 languages

The observations on the 23-language task, presented in Table 6.7, are very similar: when testing on the very same utterances, but changing **U** matrix estimation and training data condition, the systems yield 2.8 and 2.6 times more errors!

What we see is that the overall performance (tested on all utterances) of the CTS system degraded massively compared to the 8-language task (from 3.05 %*min*-C_{avg} to 9.22 %*min*-C_{avg}). It has now a lower performance (higher error rate) than the pure

Table 6.7: *Single category systems for the 23-language task, matching category tests are highlighted, in %min-C_{avg}*

NIST LRE 2009 — 23 language task				
data for estimating		closed-set 30s tests		
U matrix	models	all tests	CTS only	VOA only
pure CTS	CTS+	9.22	**7.75**	10.06
pure VOA	VOA+	7.95	21.51	**3.91**

VOA system, which did not change that much. This can not really be explained, since the *CTS+* data set used for training includes some VOA data for the missing languages (see Sect. 6.1) and for these languages, there is also only VOA test data, which thus would correspond to the data these particular models were trained with. This indicates that it is not good to train models on data of a channel category that was not present for estimating the **U** matrix.

The degradation of the matching CTS tests evaluation from 2.71 %*min-C_{avg}* (8-language task) to 7.75 %*min-C_{avg}* can only be caused by **A)** adding 3 more languages (Indian English, French and Spanish) to the **U** matrix estimation, which should not have a negative impact and **B)** testing on 2 additional languages (Indian English with very little training data and Urdu without CTS training data at all). The analysis of the error probabilities for the single languages (not shown herein) gives the explanation: It is Urdu that degrades the whole system performance since it has a P_{miss} [2] as high as 81.25% (and an average P_{FA} of 11.79%). The Urdu model was trained on VOA data only since no CTS data is available and Urdu CTS tests consist of only 32 utterances. Using all utterances for testing (the *all tests* column), the few Urdu utterances for the CTS category disappear under the number of Urdu VOA utterances[3].

The biggest part may thus come from the fact that the **U** matrix has been estimated purely on data of one category. For the CTS system, it has been estimated using only data of the 11 CTS training languages (as shown in Table 6.1). So even the models which have only VOA data available for training and which are also tested on VOA data only (refer to Table 6.1) suffer from the "false" compensation matrix. This means that the kinds of (fine grained) variabilities tracked by the **U** matrix depend up to a certain level on the channel category. The channel variability matrix does not seem as universal as in its initial definition.

Opinion: Future investigations should also analyze cross-category combinations of the data used for the **U** matrix and the models. Thus, estimating the **U** ma-

[2] These values for P_{miss} and P_{FA} are obtained in identification mode and are potentially slightly different from the ones obtained by the global threshold in *min-C_{avg}*. So the presented analysis has still its importance.

[3] We speak about utterances of the same language pooled together for testing, thus no "number un-biasing" takes place.

trix with one data set (for instance CTS) and training the language models on the other data set (VOA— or the other way around). This would not only show the trivially negative impact of training data of the false category, but (seen from another perspective) it would clearly show the impact of the channel category for the estimation of **U**, how much the **U** matrix is dependent on the channel category.

— ◇ —

pure systems conclusion

This section principally showed the big impact of the channel category. If the category of the testing utterances does not correspond to the data used for training (as well the **U** matrix, as also the language models), performances degrade very rapidly. On clean setups with matching category, we may achieve excellent performances (1.9 %$min\text{-}C_{avg}$ for the best case)[4].

The analyses of this section highlighted the fundamental problem when working on data of different (big) channel categories. It prompts for more extended research on this topic and it lays the foundations for the subsequent analyses presented in this chapter.

6.4 Data level fusion: Global approach with pooled data

The most straightforward approach in handling multiple channel categories consists in pooling all available training data together, in our case the *CTS* and the *VOA* datasets, without distinguishing or keeping track of the source. There is thus only one global channel compensation matrix (**U**) to be estimated (as described in Chapter 4). Further, one plain set of 23 language models is estimated (as described in the same chapter). In order to speak of "merging" the two categories at different levels, we can name this approach a fusion at the data level, since the two data sets are pooled together.

6.4.1 Evaluation on 8 languages

Table 6.8 shows the results on the JFA system pooling all data together to estimate the **U** matrix. The number of tests is indeed the same as indicated in Table 6.4. Evaluated on the 8 language subset, the base JFA system pooling together the training data of both categories performs at 2.23 %$min\text{-}C_{avg}$. Evaluating only on the CTS test utterances yields 3.03 %$min\text{-}C_{avg}$ and on the VOA files 1.75 %$min\text{-}C_{avg}$.

Interesting is that VOA tests slightly benefit either from the additional (model) training data or from the enhanced **U** matrix since the cost measure is reduced by nearly 8% relative compared to the pure VOA system. But this does not hold for the

[4]But this 8 language task has to be regarded with a certain caution since a good share of languages had to be removed and it does not contain "hard" language pairs any more.

CTS tests (with +11% relative). Perhaps the channel variabilities coming from VOA overwhelm those of CTS so that they are more present in this pooled **U** matrix. The question that arises is, if the VOA channel variabilities are more "stable" or more characteristic of the variability than the CTS ones, then they could emerge and get caught more easily by the **U** estimation algorithm than the CTS ones and make up a bigger part of **U**.

Since the merged (stacked) **U** matrix described in the next section has a rank of 80 (originating from twice 40), we also tested a **U** matrix having 80 channels, but using the pooled approach (denoted *pooled 80 ch*). Here, it is slightly less powerful than its corresponding matrix of rank 40.

Table 6.8: *Results of the pooled data systems, varying the model training data, compared to the single category systems (grayed out), in %min-C_{avg}*

NIST LRE 2009 — 8 language task				
data for estimating		closed-set 30s tests		
U matrix	models	all tests	CTS only	VOA only
pure CTS	CTS	3.05	**2.71**	4.28
pure VOA	VOA	6.78	10.15	**1.90**
pooled	CTS &VOA	**2.23**	3.03	1.75
pooled	CTS	2.73	**4.08**	1.51
pooled	VOA	5.30	7.28	**3.19**
pooled 80 ch	CTS &VOA	**2.99**	3.71	2.37

6.4.2 Evaluation on 23 languages

The testing conditions (as the number of utterances etc.) are still those shown in Table 6.5. Let us look at the matching condition trials while moving from the 8-language to the 23-language task: In the case of pure systems (Sect. 6.3), we observed for CTS a change from 2.71 %min-C_{avg} to 7.75 %min-C_{avg} (2.9 times more errors) and for VOA a change from 1.90 %min-C_{avg} to 3.91 (2.1 times more). When using a pooled **U** matrix (described here), for CTS the change is smaller: from 4.08 % min-C_{avg} to 8.50 %min-C_{avg} (2.1 times more). Also for VOA, the change is smaller: from 3.19 %min-C_{avg} to 5.62 %min-C_{avg} (1.8 times more). In conclusion, the costs are higher when using a pooled matrix, but the more difficult 23-language task seems nevertheless to be able to take some benefit from this pooling.

— ◇ —

data level summary

As global conclusion, pooling of the **U** matrix slightly flattens out the difference between matching and cross-category tests: matching condition tests perform worse and cross-condition tests considerably better. So that the overall performance (by testing with all test utterances pooled together, *all tests*) improves about 20% relative.

Table 6.9: *UBM-based JFA systems with pooled data, compared to the single category systems (repeated from Table 6.7), in %min-C_{avg}*

NIST LRE 2009 — 23 language task				
data for estimating		closed-set 30s tests		
U matrix	models	all tests	CTS only	VOA only
pure CTS	CTS+	9.22	**7.75**	10.06
pure VOA	VOA+	7.95	21.51	**3.91**
pooled	CTS &VOA	**4.79**	6.71	4.57
pooled	CTS+	7.23	**8.50**	6.93
pooled	VOA+	6.27	11.96	**5.62**
pooled 80 ch	CTS &VOA	**5.36**	7.24	5.04

Until this stage, the best system (on whatever test set) is the system where the **U** matrix is estimated on all data together and where the language models are also estimated on all data of both channel categories.

6.5 Model level fusion: Merged channel compensation matrices

After having trialed pooling all training utterances together on the data level, the next level would be to put together the two systems on the model level.

One approach is thus to estimate separate channel compensation matrices (**U**) and to "merge" these two matrices together by stacking one on top of the other. We will call it *merged*, which is not to be confound with the *pooling* of data described in previous section. The new session compensation matrix is built by concatenating the two category dependent matrices we have already estimated for the pure CTS and the pure VOA system described in Sect. 6.3. This forms a new **U** matrix of double rank (in our case 80 instead of 40).

The difference to a compensation matrix with the same (double) rank estimated on all data is that, in the present case, there are half of the channels assured for each category, whilst for the pooled case, the channels are allotted dynamically — some channels may even be similarly present/found in both categories.

Two sites participating in NIST LRE 2009 have used a similar strategy of stacking session compensation matrices (ATVS NIST LRE, 2009; IFLY NIST LRE, 2009).

Using this merged (thus dual-category) matrix, we may train the language models using either training data pooled together or we may build category specific models using either data set, *CTS* or *VOA*. Thus the training step corresponds to what was also done in last section.

6.5.1 Evaluation on 8 languages

In the case of concatenated compensation matrices, we have a rank of 80. The results using this stacked compensation matrix and different training set compositions are shown in Table 6.10. It indicates that the results are rather similar to the pure systems. In reference to what has been said in last section's conclusion (that the difference between matching and cross-condition is lessened), the results using merged-**U** very slightly deepen this gap. This insinuates that it is sufficient if the **U** matrix is estimated on data of the channel category that is also used for model training. Adding data of an other channel category does not have a big impact (it has just a slight helping tendency) if no similar data is used for the models.

Table 6.10: *Systems featuring a merged matrix, models trained on all data or on only one category, in %min-C_{avg}*

NIST LRE 2009 — 8 language task				
data for estimating		closed-set 30s tests		
U matrix	models	all tests	CTS only	VOA only
pure CTS	CTS	3.05	**2.71**	4.28
pure VOA	VOA	6.78	10.15	**1.90**
pooled	CTS &VOA	**2.23**	3.03	1.75
pooled	CTS	2.73	**4.08**	1.51
pooled	VOA	5.30	7.28	**3.19**
pooled 80 ch	CTS &VOA	**2.99**	3.71	2.37
merged **U**	CTS &VOA	**2.33**	3.06	1.68
merged **U**	CTS	3.32	**2.61**	4.76
merged **U**	VOA	6.78	10.34	**1.92**

evolutive sequence: CTS

Let us have a look at following sequence for the CTS tests only, in the optics of a somehow pessimistic approach: Cross-category MAP is at 24.62 %min-C_{avg} (Sect. 6.2.1), pure cross-category JFA drops to 10.15% (−59% relative), adding some data that matches the testing category (merged-**U** and still adverse *VOA*+ data for the models) stays similarly at 10.34 %min-C_{avg}. Adding some data that matches the model training category improves the performance to 3.06 %min-C_{avg} (another −70% relative, which gives a total of −88% relative over the MAP setup).
So we see that JFA is very useful, even for channels purely estimated on channel/sessions of a quite different category. These results tend thus to show that the channel (and speaker) variability captured by the compensation matrix is for a certain part of some global nature, which is independent of the channel category. We further see the importance to have training data as similar to the testing data as possible — this seems to be even of more benefit for JFA (the above −70% relative) than for MAP (−20% relative).

VOA

For the analogous (pessimistic) approach on VOA tests only, we observe: Cross-

category MAP 19.27 %*min-*C_{avg}, pure cross 4.28 %*min-*C_{avg} (−78% relative), adding matching channels gives 4.76 %*min-*C_{avg} and adding matching training data yields 1.68 %*min-*C_{avg} (another −64% relative, or over −91% relative compared to the baseline MAP approach).

For matching category, CTS test utterances, we have a cost reduction of 86% relative between simple MAP and the pure JFA system. For matching VOA, the cost reduction is of about 84% relative.

— ◇ —

conclusion

Globally, we see that performance does not vary radically by changing the way the session compensation matrix is estimated (at least when evaluating on cross-category condition). Thus the most important, besides having some compensation matrix (thus using JFA), is to have at one's disposal a lot of training data, with a part of it as similar to the testing data as possible.

6.5.2 Evaluation on 23 languages

The corresponding results for the 23-language task are displayed in Table 6.11.

Table 6.11: *Merged systems, different data sets for model training, in %min-*C_{avg}

NIST LRE 2009 — 23 language task				
data for estimating		closed-set 30s tests		
U matrix	models	all tests	CTS only	VOA only
pure CTS	CTS+	9.22	**7.75**	10.06
pure VOA	VOA+	7.95	21.51	**3.91**
pooled	CTS &VOA	**4.79**	6.71	4.57
pooled	CTS+	7.23	**8.50**	6.93
pooled	VOA+	6.27	11.96	**5.62**
pooled 80 ch	CTS &VOA	**5.36**	7.24	5.04
merged **U**	CTS &VOA	**4.47**	7.16	4.14
merged **U**	CTS+	6.23	**6.96**	6.86
merged **U**	VOA+	6.16	13.13	**3.72**

23-L analysis

For this 23-language task, systems using a merged channel/speaker variability compensation matrix and that are still trained on only one or the other channel category clearly benefit from the extended channel modeling (80 instead of 40 and more diverse channels): an improvement of −32% relative and −39% relative for cross-category tests and also slightly (−5% relative to −10% relative) for tests with matching category. Also for the official NIST LRE setup (evaluating by taking all testing utterances together), the improvement is in the order of −32% relative and −23% relative (for CTS+ and VOA+ training respectively).

comparison to 8-L

This observation about the merged-**U** setup improving in every case for the 23-language task could not be confirmed in the "cleaner" 8 language setup, where the performances with the merged **U** matrix were very similar to the category specific ones. In that setup, we have all required language–channel combinations in our training data and thus the additional channels are not of much use. An interpretation of this difference between the 8- and the 23-language setups may be that the data used for estimating the **U** matrix has at least to cover the language–channel combinations used for model training[5].

evolutive sequence: CTS

As we did for the 8-language task, let us have a look at following (pessimistic approach) sequence for the CTS only tests: Cross-category MAP has a $min\text{-}C_{avg}$ of 35.48% (Sect. 6.2.2), pure cross-category JFA reduces the errors/costs to 21.51 % $min\text{-}C_{avg}$ (−39% relative), adding some matching channels (merged **U** matrix and still adverse *VOA*+ model data) drops to 13.13 %$min\text{-}C_{avg}$ (another −39% relative, or −63% relative from the MAP system) and then adding some channel-matching data for model training improves to 7.16 %$min\text{-}C_{avg}$ (additional −45% relative, that is about −80% relative compared to the adverse MAP system).
As difference to the 8-language task, we notice here that adding some channels of matching category to the **U** matrix helps. This means also that JFA does not really help for channels of non-matching category. The case described here reveals also that it is of the same importance to have data of corresponding category in the channel variability matrix as to have such data for training the models.

VOA

Analogously for VOA only tests: Cross-category MAP 27.89 %$min\text{-}C_{avg}$, pure cross 10.07 %$min\text{-}C_{avg}$, adding matching channels gives us 6.89 %$min\text{-}C_{avg}$ (32% relative) and adding matching training data yields 4.18 %$min\text{-}C_{avg}$ (another 39% relative, with totally 85% relative).

CTS degradation

The drastic degradation compared to Table 6.10, occurring in a general manner on the *CTS* systems (doubling or tripling the number of errors) can be explained by the fact that in this 23 language case, not even half of the models could really be trained with *CTS* data (the *CTS*+ data set containing VOA data for the 12 languages that do not have CTS training data).

Urdu

For the system where *CTS*+ data was used for model training, the cost for the CTS-only tests (matching) is slightly higher than the one of the VOA-only test utterances (cross, on the same models). This can again be explained by the high P_{miss} of the Urdu language: 65.63% for the CTS&VOA trained system and 71.88% for the CTS+ one. This finally gives an Urdu C_{DET} of 38.42% and 41.12% respectively (for C_{DET}, see Sect. 2.6.2.2).

— ◇ —

model level conclusion

Comparing the results between pooled-**U** and merged-**U** matrices seems to highlight that the channel variability mapping in the compensation matrix is fairly

[5] And above analyses seem to indicate that there should be a match between model training and testing data. The direct link between compensation matrix and testing data seems to be far less important.

on a per-category basis. So adding additional (foreign–category) channels does not help in a context where training data of the same category as the test data is available for every language (as the more controlled environment of the 8-language task). But the additional channels can prove useful when correct channels are missing from the training data for some languages (23-language setup). Thus the merged-**U** system proves well when a certain robustness is required.

Looking at the performance on all 30 seconds tests, no single channel dependent system performs as well as systems with models trained on all data together.

6.6 Score level fusion: Fully separate category modeling

system selection

In Sect. 6.3, we observed that performance is very good when channel compensation matrix, training data and test part are of the same channel-category (matching condition). This leads to the question if we could not fuse two completely separate category-dependent systems. One of the easiest ways to merge two such completely separate systems is by selecting the score that matches the correct channel category. By taking the corresponding system for each testing utterance, this tries to obtain the benefits of both systems. Thus, in this last approach, the fusion is delayed until reaching the score level.

category detector

So, fully separated CTS and VOA systems are trained and the testing utterances are scored against both systems separately. The idea is then to take, for every single utterance, only the scores of one or the other system. Ideally, the choice is done according to the real channel category of the testing utterance. In a first run, this ideal choice can be simulated using an oracle, which returns the channel category of the utterance.

oracle

This section presents and discusses the results using an *oracle* type selector. The oracle tells us of which category the test utterance really is. Our fusion unit then selects the scores of the corresponding channel-dependent system for that utterance. Since this selector is perfect in terms of channel detection, we can interpret the result as being the best performance a fusion unit based on automatic channel detection may approach. We employ an oracle to investigate the feasibility and the use of a system combination at this score level.

The separate Chapter 7 will then replace this oracle by automatic channel category detectors. It will principally dwell on the design of such detectors.

Also for this approach, one problem is the missing training data described in Sect. 6.1, which inhibits training a full set of 23 language models for both categories. But we tried to solve this issue by taking data of the other category, where really necessary.

6.6.1 Evaluation on 8 languages

Table 6.12 evaluates the fusion for the 8-language task. Once, it combines the pure CTS (2.71 %$min\text{-}C_{avg}$ on the matching tests) with the pure VOA (1.90 %$min\text{-}C_{avg}$ on VOA) systems. This yields a global cost of 2.06 %$min\text{-}C_{avg}$ for the merged system. This result is interesting since it is better than the global results of the system merging the channel compensation matrices (merged-**U**, −12% relative) or the system pooling all data together since beginning (−7.7% relative). We note also that the per-category testing has to produce the same results as the corresponding single system (due to the oracle).

The same kind of fusion applied on the category-dependent systems featuring a merged channel compensation matrix performs insignificantly better at 2.04 % $min\text{-}C_{avg}$, which is 8.6% relative better than the best single system.

Table 6.12: *Fusion of JFA systems using an oracle type system selector, compared to best single systems, 8-language task, in %$min\text{-}C_{avg}$*

LRE 2009, 8 languages						
system 1		system 2		closed-set 30 s tests		
U matrix	models	**U** matrix	models	all tests	CTS only	VOA only
pooled	CTS &VOA			**2.23**	3.03	1.75
merged **U**	CTS &VOA			**2.33**	3.06	1.68
CTS	CTS	VOA	VOA	2.06	2.71	1.90
merged **U**	CTS	merged **U**	VOA	2.04	2.61	1.92

We note that the result on all 30 s tests (grouping CTS and VOA segments) can differ from the value that would be obtained by averaging the matching category performances of the two corresponding single systems (weighted by the number of tests). This happens since the indicated performance uses the average detection cost function (C_{avg}) and is thus a mean of language pair (mis-) detection costs, based on a global threshold which may also change from experiment to experiment.

6.6.2 Evaluation on 23 languages

The oracle fusion is also applied on the whole set of 23 languages to match the LRE 2009 protocol. The results using an oracle as selector for fusion are shown in Table 6.13.

The fusion of the two pure systems is with 4.16 %$min\text{-}C_{avg}$ slightly outperformed by the fusion of the systems built on a common, merged eigensession matrix, which runs at 3.90 %$min\text{-}C_{avg}$ (probably significant −6% relative). This is similar to the difference that has been observed between unfused pooled and merged-matrix systems in Table 6.11 (and repeated in Table 6.13). A big part of the easier recognition on VOA tests is still due to the fact that nearly all language–channel combinations

Table 6.13: *Fusion of UBM-based JFA systems using an oracle type selector, compared to best single systems, 23-language (NIST LRE 2009) task, in %min-C_{avg}*

LRE 2009, 23 languages						
system 1		system 2		closed-set 30 s tests		
U matrix	models	**U** matrix	models	all tests	CTS only	VOA only
pooled	CTS &VOA			**4.79**	6.71	4.57
merged **U**	CTS &VOA			**4.47**	7.16	4.14
CTS	CTS+	VOA	VOA+	4.16	7.75	3.91
merged **U**	CTS+	merged **U**	VOA+	3.90	6.96	3.72

are present in the training data. This fused system with a minimal average cost of 3.90 %*min-C_{avg}* has a cost reduction of 12.6% relative compared to the best single system with its 4.47 %*min-C_{avg}*.

6.7 Channel category conclusion

problem The data used for the NIST LRE 2009 contains utterances of two different sources: Conversational Telephone Speech (CTS) and Voice of America (VOA). We called these two kinds being of different *channel categories*. We suspected that these two sources have considerably different structures or variabilities. We tried to design an approach which allowed these two categories to be separated. The subsequent analysis revealing striking differences between these two categories, most notably in the context of Joint Factor Analysis.

solutions We tried to cope with this category problem by designing parallel category-specific systems. These systems can be combined together on different levels: **1)** On the data level, **2)** on the model (or more accurately the JFA **U** matrix) level and **3)** finally on the score level. The data level merging consists in naively pooling all data together. With 4.78 %*min-C_{avg}* on 2048 Gaussian on the LRE 2009 setup, it achieves already acceptable performances. The combination on the model level features a compensation **U** matrix, which is obtained by stacking two category-specific ones. This systems runs at globally at 4.47 %*min-C_{avg}* (-7% relative). Finally, the fusion on the score level analyzed here uses an oracle channel category "detector". Depending on this detector's output, the scores of one of two category-specific systems are chosen. The best such combination runs at comparable 3.90 %*min-C_{avg}*, which correspond to the middle-field of NIST LRE 2009's participants. This shows that it is possible to improve over the naive pooled (JFA) approach (-18% relative). Compared to the baseline MAP approach, this is a reduction of 81% relative.

observations One important observation of the presented results is that the fact of adding more data to the **U** matrix estimation does not necessarily mean an improvement. Certainly, the robustness against testing utterances of a channel category which

was not included in the model training data (cross-category) increases. But this enhancement comes at the expense of reducing the channel compensation power on the data of known (matching) category.

But when imposing a certain (a priori) structure on the **U** matrix, as is for instance implicitly done when concatenating category-dependent matrices, significant improvements may be obtained[6]. This way allows to benefit of the additional information about the other channel category, which can be included into the corresponding part of the **U** matrix. We have further the impression that the VOA part of the compensation matrix slightly overwhelms the CTS one.

One finding was also that the data used for estimating the **U** matrix has at least to cover the language–channel combinations used for model training.

In general, the most important bit is to have model training data that is as close to the testing data as possible. The data used for model training is more crucial than the data used for estimating the **U** matrix.

[6] *Opinion:* We think that the two matrices that get stacked may need some proper balancing in order to benefit fully from both. We tried to normalize each column (corresponding to one channel factor) individually by its norm. But this strategy was not useful...

Les machines un jour pourront résoudre tous les problèmes,
mais jamais aucune d'entre elles ne pourra en poser un !

— Albert EINSTEIN

Chapter 7

Channel-Detectors for System Selection

Contents

If we have rather different channel categories (as analyzed in Chapter 6 for NIST LRE 2009), one possibility is to handle these categories completely separately in a parallel manner and merge such category-dependent systems at the score level. This combination may be done by a system or score selector taking the "better" system for each utterance. This chapter extends this idea of a channel category detector, already presented in Sect. 6.6. Here, we replace the oracle based detector by automatic detectors in different designs.

— ◇ —

7.1 Designing channel category detectors

In previous chapter and in (Verdet et al., 2010b), we investigated the idea of modeling the CTS and the VOA conditions separately. We have shown that the merging of such separate systems may be done at different levels:

1. Pooling all data together since the beginning (having thus just one common system),
2. stacking channel-category dependent JFA channel compensation matrices (**U**) in order to have a matrix with a CTS specific and a VOA specific part and
3. merging two completely channel-dependent systems only at the score level.

A simple, but nevertheless effective way to merge such systems at score level is using a system selector. This means that, for each test utterance, the scores of one or the other system are taken (selected). Typically, such a system selector acts according to the channel category detected in the test utterance and selects the scores of the corresponding system.

In previous chapter, we employed an oracle category detector. The work presented in this section investigates different ways to design automatic channel detectors in the context of such a system selector. More specifically, we design channel detectors based on SDC features (Sect. 3.3.2), as well as detectors working on the JFA level. In the latter variant, the term containing the session variability (the **x** vectors of the JFA formula Eq. 4.5) is used. This is a simple, but novel idea, which at the same time also validates the fundamental idea behind the JFA approach.

The impact of the different channel detectors are evaluated on systems with two different structures, already discussed in Chapter 6:

Pure channel-category systems where the compensation matrix **U** has a rank of 40 and is estimated exclusively on data of one or the other channel category.

Merged-U systems that use a common **U** matrix, which is obtained by stacking the two channel dependent **U** matrices and has thus a rank of 80. The category distinction lies only in the language model training.

— ◇ —

The following sections present the different channel category detectors we designed. It is then followed by a comparative analysis presenting results on the two category-dependent type of systems.

7.1.1 Simple-sum

The *simple sum* fusion is not a channel detector, but a baseline replacement for the system selector. For each test, the scores of both channel-category dependent systems are summed together (without special weighting). This gives a minimal system performance which acts as a baseline for the different channel detectors.

7.1.2 Feature-based MAP

As a first automatic approach, a Maximum A Posteriori (MAP (Gauvain and Lee, 1994)) adapted model is estimated for each of the two categories using all training data (feature vectors) of that channel category. Since the same UBM as for training the channel-dependent systems is used, these channel models are mixtures of 2048 Gaussians. Let us identify this detector by *f-MAP* . It has a channel identification rate of 87.63% when evaluated on the 30 second NIST LRE 2009 segments.

7.1.3 SVM on channel variability

Since JFA tries to model the session and channel variability separately and expressively, it is obvious to try to use this information alone for a channel detector. The channel variability part of the JFA formula (Eq. 4.5) is the term $\mathbf{Ux}$. Since $\mathbf{U}$ is globally fixed, the vector $\mathbf{x}$ represents the channel variability (the channel factors). In the present setup, these $\mathbf{x}$ vectors are obtained in every case using the UBM and the stacked $\mathbf{U}$ matrix (Sect. 6.5), since they have to be obtained on a common $\mathbf{U}$ matrix in order to be comparable.

These $\mathbf{x}$ vectors (here with a dimension of 40) can directly be used as input SVs for an SVM (Singer et al., 2003; Campbell et al., 2004; Verdet et al., 2009b). The $\mathbf{x}$ vectors of the target category are taken as positive SVs and the $\mathbf{x}$ vectors of the other category as blacklist (negative examples). We notice that the two SVMs we get for our two-category case are symmetric (in theory, for a given test, just the sign of the output score changes). We shorten this detector as *x-SVM* and it has a channel identification rate of 87.41% on LRE 2009 30 seconds.

7.1.4 MAP on channel variability

These FA $\mathbf{x}$ vectors can also be used as new features (thus as new front-end) on which a new channel-UBM can be estimated. This can then be adapted through MAP to obtain channel-dependent models working on these $\mathbf{x}$ vectors. For the work presented here, we use models of 64 mixtures (since each utterance is represented by one frame only). This detector returns the channel category corresponding to the model with the bigger likelihood We write this channel category detector *x-MAP* and it has an accuracy of 75.29% on the 30 second NIST LRE 2009 segments.

7.1.5 Oracle

The *oracle* represents the error-less channel detector. It returns the true channel category of an utterance. Evaluating the systems using the oracle as channel detector,

gives the performance we want to approach by automatic channel detectors.

— ◇ —

The performances of automatic data-based channel detectors are thus expected to lay between the one of a simple-sum fusion and that of the oracle.

7.2 Novel score normalizations

We saw different score normalization strategies in Sect. 2.5.2 and in Sect. 3.5. What concerns inter-utterance normalization, we saw the *divSum* and the *divSum-K* approach. The latter requires a constant K to be estimated or empirically chosen beforehand. This section presents novel approaches to replace this cumbersome technique. The hypothesis is not to rely on any tunable parameter and to understand what is really going on.

7.2.1 llkMax0

The *llkMax0* score normalization presented in these lines is also an inter-utterance normalization, meaning the scores are normalized separately for each test utterance.

Each Likelihood score is divided by the maximum of the scores the utterance obtains against all language models. Expressed in Log-Likelihood domain, this uniformly shifts all scores in order to assign a Log-Likelihood of 0 to the hypothesized language yielding the biggest score (hence the name of *llkMax0*):

$$s_l(\mathcal{X}) = \log\left(\frac{e^{LLk_l(\mathcal{X})}}{\max\limits_{k\in L} e^{LLk_k(\mathcal{X})}}\right) = LLk_l(\mathcal{X}) - \max_{k\in L} LLk_k(\mathcal{X}) \tag{7.1}$$

where $LLk_l(\mathcal{X})$ is the Log-Likelihood of the utterance $\mathcal{X}$ and the hypothesized language l, whereas L is the set of all languages.

DET step

This procedure has the effect of aligning the top-scores of all utterances to a Log-Likelihood of 0 (or 1 in Likelihood domain). De facto, this accepts all highest scores, producing a minimal level of false acceptance rate[1].

focus

The problem of setting the threshold thus exploits principally the information about the second-biggest score. If the second-biggest score of an utterance lies near to the biggest one (thus rather near to 0), it will pass to the positive side of the threshold sooner than if its distance to the biggest one was larger. Hence the basic idea behind this approach is to focus mainly on the second-biggest scores. The

[1] While lowering the threshold from the high end (imagine the DET plot), the progress of the false accept rate will jump from 0 to this minimal level. Then, it potentially stays at this level for some range. This produces a rectilinear step on the DET plot. The threshold at this edge point lies hence at (normalized) score 0.0.

assumptions are the following: (i) If the biggest corresponds to the true language, we are on the safe side (de facto acceptance). (ii) If the true one is not the biggest one (the recognition was not optimal), the chances that it is the second-biggest is higher if the spacing between these two top scores is relatively small.

Fig. 7.1 (a) shows the distribution of *llkMax0* normalized scores of a typical JFA experiment. Note that the big (target) peak as result of the 0.0 alignment is cut off since it extends to some big value[2]. We see that the false acceptance peak of the non-targets is far smaller[3]. Compared to the non-targets, the targets that did not make it to 0 (thus second-biggest and beyond, here accounting for 16.8% of the targets) are all relatively proximate to 0. In fact, the distributions resemble to the ones of *divSum-K* in Fig. 2.7 with a K high enough.

— ◇ —

We looked for a simpler scheme than the division by the sum put to a power of K (*divSum-K*) described in Sect. 2.5.2.2 in order to avoid having to tune the parameter K. The simple division proposed here has the big advantage not to depend on any tunable parameter, nor on the availability of a separate calibration dataset (as required for more evolved back-ends like the GBE). Nevertheless it still performs as well as the *divSum-K* approach.

The basic idea behind this *llkMax0* score processing is mainly to exploit the constellation of the few highest scores of each utterance (mainly the second-highest). This kind of score normalization is used throughout the current chapter.

7.2.2 llkInt01

In last section, we presented the *llkMax0* approach with the big advantage to perform as well as *divSum-K* , but not requiring any parameter (K). Here, we present even an other approach, which can be logically derived from *divSum-K* . But the chronological development was inverse since this *llkInt01* normalization process as designed by Jasmin Wei Ming Liu[4], whose idea was then spun further.

— ◇ —

The principle of *llkInt01* is not only to put the highest score to some fixed value, but doing similarly for the minimum. This transforms the set of (Log-Likelihood) scores of one utterance to a fixed interval, namely 0 to 1. (We note that in this case, we can not see the result as being Log-Likelihood-like any more, but a more appropriate interval could have been chosen.)

[2] In the present case up to over 8300
[3] Here fitting on the plot with 35.3
[4] Affiliated to Ultra Electronics AudioSoft and to the University of Swansea.

The scores are thus shifted and scaled as follows:

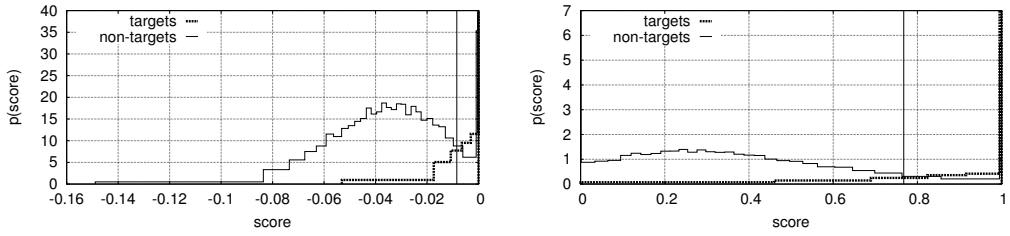
$$s_l(\mathcal{X}) = \frac{LLk_l(\mathcal{X}) \quad - \min_{k \in L} \left[LLk_k(\mathcal{X}) \right]}{\max_{k \in L} \left[LLk_k(\mathcal{X}) \right] - \min_{k \in L} \left[LLk_k(\mathcal{X}) \right]} \tag{7.2}$$

where $LLk_l(\mathcal{X})$ is the Log-Likelihood of the utterance $\mathcal{X}$ and the hypothesized language l, and L is the set of all languages[5].

Fig. 7.1 (b) gives an idea of the score distribution after *llkInt01* normalization for the same system as in (a) and in Fig. 2.7. Here again, the y-axis is truncated[6].

— ◇ —

The approach presented here formed the basis for the thoughts, which lead to the *llkMax0* technique described in Sect. 7.2.1. The results using this approach here are slightly better than the bare *divSum* approach, but not as good as *divSum-K* . This can principally be funded on the problem induced by aligning the minimum. The minimal scores do not have (or has only very little) influence on thresholding. The minimum more depends on the set of languages in our task. It is more the high-scoring range which is decisive.



(a) llkMax0 (b) llkInt01

Figure 7.1: *Sample score distributions for llkMax0 and for llkInt01 normalized scores for our JFA system*

7.3 Evaluation on 8 common languages

setup The results presented subsequently are based on SDC features. All models are obtained through MAP adaptation from the UBM with JFA channel and speaker normalization. All models are composed of 2048 Gaussians. Data parts and general system structures are the same as in Chapter 6, following the NIST LRE 2009 protocol and adding an 8-language task. The results presented in this section are based on scores which are normalized using the llkMax0 approach (Sect. 7.2.1)[7] and

[5] As degenerate case of all scores happening to be the equal (possible consequence of an utterance containing no speech), the denominator is set to 1 — so that the resulting scores are all equal to 0.

[6] At the 0 edge, where the minima are aligned, the non-targets reach about 137 in the present case. At the 1 edge of the maxima, the targets reach $2.3 \cdot 10^9$ and the non-targets as indicated 5.6.

[7] This change in normalization strategy accounts for the very slight change of the values compared to last chapter.

evaluated under the *min-C_{avg}* measure.

As explained in Sect. 6.1, there are some language–channel combinations which lack training or testing data. This section thus evaluates the systems on the NIST LRE 2009 30-second segments of the 8 common languages only, as well as solely on the CTS and solely on the VOA subset. 8-L task

7.3.1 Pure systems

Table 7.1 presents the results of the two pure channel-category dependent systems and their fusion. The results of all automatic channel detectors fall in between those of a simple-sum fusion and the oracle. We observe that the best results among the automatic channel detectors are obtained by the ***x*-SVM** detector. They approach the ones of the oracle, which represents ground truth, quite well (only $\sim$6% relative from the oracle away). The weakest channel detector is the one where the same **x** vectors are modeled by MAP.

Table 7.1: *8 languages, pure per-channel systems, in %min-C_{avg}*

		LRE 2009 closed-set 30s		
base system	fusion	all 30s tests	CTS tests only	VOA tests only
CTS	—	2.34	2.11	2.94
VOA	—	6.34	9.88	1.28
—	simple-sum	2.58	3.67	1.30
—	oracle	1.63	2.11	1.28
—	*f-MAP*	1.88	2.49	1.38
—	***x*-MAP**	2.31	3.06	1.73
—	***x*-SVM**	**1.73**	**2.27**	**1.26**

7.3.2 Systems with merged-Umatrix

The results shown in Table 7.2 are obtained by systems featuring a common merged **U** matrix. This matrix is obtained by stacking the two channel-dependent **U** matrices used for the pure systems. The observations for the merged-**U** systems are similar to those for the pure systems. Except for the VOA only evaluation, we identify a clear general tendency of slightly better performances for these merged-**U** systems compared to the pure category systems (i.e. with 4.3% relative gain for the ***x*-SVM** channel detector). When evaluated on the channel-categories separately, the feature based MAP detector (*f-MAP*) is slightly better than ***x*-SVM** with 2.30 %*min-C_{avg}* for CTS tests and 1.44 %*min-C_{avg}* for VOA tests.

Evaluated on all 30-second segments, system selection based on the ***x*-SVM** channel detector have a minimal average cost of 1.65%, which is respectable $-$90%

relative over the most basic MAP system (trained on all data pooled together).

Table 7.2: *8 languages, merged-systems, in %min-C_{avg}*

		LRE 2009 closed-set 30s		
base system	fusion	all 30s tests	CTS tests only	VOA tests only
CTS	—	2.53	2.04	3.63
VOA	—	6.30	6.64	1.48
—	simple-sum	2.40	3.48	1.60
—	oracle	1.55	2.04	1.48
—	*f-MAP*	1.75	**2.30**	**1.44**
—	*x-MAP*	2.27	2.96	1.87
—	*x-SVM*	**1.65**	2.39	1.45

7.4 Evaluation on all 23 languages

This section now presents the same systems under the 23 language NIST LRE 2009 task. It also shows to which extent the systems are robust enough to recognize languages of a channel category for which no training data is available.

7.4.1 Pure systems

The results in Table 7.3 show that the automatic channel detectors achieve results that are a bit further off the oracle (about 12% relative) compared to the 8-language protocol, but they remain clearly closer to the oracle than to the simple-sum performance.

Table 7.3: *23 languages, channel-dependent , in %min-C_{avg}*

		LRE 2009 closed-set 30s		
base system	fusion	all 30s tests	CTS tests only	VOA tests only
CTS	—	9.87	7.44	11.05
VOA	—	8.59	25.40	3.73
—	simple-sum	8.70	16.73	6.39
—	oracle	3.95	7.44	3.73
—	*f-MAP*	**4.47**	**8.24**	**4.02**
—	*x-MAP*	5.94	9.84	5.51
—	*x-SVM*	4.65	8.35	4.36

7.4.2 Systems with merged-Umatrix

The performances of the systems using a common stacked **U** matrix are given in Table 7.4. As for the 8-language task, they also indicate that these systems perform better than the pure systems. The *f-MAP* channel detector performs best with its 3.85 %*min-*C_{avg}. Its enhancement over the channel-category dependent **U** matrix (pure) systems is 14% relative. Compared to the simplest MAP system, this shows an improvement of over 81% relative for all 30-second test utterances.

Table 7.4: *23 languages, merged-systems, in %min-*C_{avg}

		LRE 2009 closed-set 30s		
base system	fusion	all 30s tests	CTS tests only	VOA tests only
CTS	—	6.59	6.63	7.51
VOA	—	5.80	13.77	3.43
—	simple-sum	4.32	7.17	3.92
—	oracle	3.64	6.63	3.43
—	*f-MAP*	**3.85**	**7.01**	**3.54**
—	***x****-MAP*	4.78	7.58	4.51
—	***x****-SVM*	4.44	7.58	4.09

7.5 Discussion

All these results show that channel detectors may be designed in different ways and that they may approach the performance of oracle based ground truth fusion up to 5–6% relative.

Of the analyzed channel detectors, ***x****-SVM* and *f-MAP* achieve similar performances. Whereas the former performs marginally better when training data for all channel-categories (and languages) is available and the latter seems more robust to incomplete data.

The results on the **x** vector based detectors confirm the basic idea behind Joint Factor Analysis, in which the channel variability is captured by the **Ux** term of Eq. 4.5.

perspectives

The validation of **x** vector based channel detectors opens the important and interesting perspective of completely data driven systems that automatically cluster and identify channel categories in the training data (instead of having the labels about CTS and VOA). This is not possible on the feature level, since the information about the channel (and thus about the channel category) is mixed up with the information about the language itself. But the channel category detectors based on the **x**-vectors make this possible.

Opinion: Even better results may be obtained upon employing more sophisticated back-ends. But these require separate development data for estimating their parameters.

L'ampule électrique n'a pas été inventée
en perfectionnant la bougie !

— Unknown colleague assistant,
on the why of research, CIES, 2009 [0]

Chapter 8

Perspectives

Contents

In this chapter, we will give some thoughts about what additional elements may have been tried and which are likely to enhance results or solve spotted problems. We will also discuss some bigger research directions, which may deserve further investigation.

8.1 Development set

inter-language normalization

We saw two kinds of normalization on the score level: inter-utterance and inter-language normalization (Sect. 2.5.2), the latter including more elaborate back-ends. Such inter-language techniques try to even out the strengths of the different language recognizers (models). They usually depend on some previously estimated normalization parameters. The different analyses on this topic conducted during our works turned out that this style of normalization made certain demands on the data set used for estimating these parameters.

— ◇ —

[0] English transition: *The electric bulb has not been invented by perfecting the candle!*

dev set

Ideally, the data which is run through the system to obtain scores on which these parameters are gathered comes from a spare data set. This is usually done this way because we assume that a certain structure, which is captured by the modelization step, gets also expressed in the scores. For instance, data which is hard to model or outliers to the modelization is also likely to have bad scores. Score normalization with parameters estimated on the same data set as the models may even reduce the universality or robustness (appropriateness to slightly different data) for the testing step.

The parameters for such score calibration should be estimated using a separate data set, commonly called *development set*. In the current setup, we did not keep apart some part of the data for this and included all available parts for estimating the models and the variability matrix at best. In early stages, we were not completely aware of the importance of score calibration (or inter-language normalization). Only with concentrating our attention to the score processing step, we saw that better results could probably have been obtained using a more evolved back-end (BE). In the context of LRE 2009, (Jancik et al., 2010) investigated different compositions for the development set.

analyses

Several tests and analyses[1] were carried out to put into action an inter-language normalization BE using the training set instead of a development set. But all results yield performances at the same level or marginally poorer.

— ◇ —

A proper score normalization is crucial for obtaining good results. Even if we spent enough time on this part of the system, which does not lie in the core topic of this thesis, a better approach should be chosen. Some development data set should be defined and typically a Gaussian Back-End (GBE) based processing pipeline (Sect. 3.5.2.1) should be applied. This could potentially lead to a system with increased performance, to a system with less loss due to calibration (see Sect. 3.6.1 for this concept).

8.1.1 Chunked training files

When trying to add a back-end (GBE or similar) to our system by estimating its parameters on the training data itself, we run the training utterances as if they were testing segments through our system. Having thus the scores of the training utterances, we may analyze the "quality" of training (possibly biased by overfitting) by calculating the system performance on these (known) utterances. Doing so, we observe that only a very few (typically 0, 1 or 2) errors are made on the whole set and that all other utterances are recognized correctly. This happens on the LRE 2005 protocol, where only CallFriend data is used in the training phase.

number of utterances

There seems to be two reasons for these good results on the training data itself:

[1] These range from a kind of simple Tnorm, over the use of FoCal (FoCal, 2007), to Logistic

The first is that there is only a very limited number of utterances in this (CallFriend) training set, namely 120 or 240 per language, which is a total of 1 200 over all seven languages. This is potentially too few for a good estimation of BE parameters.

duration

As second reason, these CallFriend files contain each about 14 min of speech, since they originate from half-hour telephone conversations. This is far more data that is at the recognizer's disposal than in the real testing utterances with 30 s (or 10 or 3 s resp.). On this side, the fact that an elaborate BE does not give any improvement may also be due to this difference in length between the training (or calibration training) and testing utterances. A difference in duration from the training to the testing data may also introduce some bias to the results — this is particularly true for SVMs (see discussion of SVMs in Sect. 5.4.1).

solution/ results

One solution (while sticking to the training data) would be to cut each utterance into smaller parts to be handled individually. This procedure was attempted in some early stage by extracting segments containing at least 30 s of speech from the CallFriend utterances. The individually applied mean and variance normalization (feature level) did not show any effect to the modelization (and hence to the results). Also, the JFA step estimating the **U** matrix using these shorter utterances showed itself insensitive to this change. Due to these reasons, this procedure chopping the long utterances into shorter ones was left apart in later development and it was not re-attempted for the BE normalization. But this surely constitutes a possible way if we want to add a more evolved BE.

As a reference to other works choosing this chopping approach, we may cite (Castaldo et al., 2007c), which split the same CallFriend utterances into 8172 chunks of about 150 s each.

8.2 Higher level approaches

In this work, we focused on GMM based acoustic modeling. This could form the basis of a vast diversity of further researches exploring higher level information, but keeping the basic core of the system simple and based on such short time acoustics.

The following sections will shortly present some ideas of such short time acoustics based higher level systems.

8.2.1 Signal dynamics on Gaussian indices

multi-dimensional path

We saw that each feature vector can be seen as a point in multidimensional space. A speech utterance, which consists of a series of feature vectors, can be represented as a sequence of points. Since the signal is thought to be quasi-stationary and continuous on a very short term, the values of adjacent feature vectors change only

Regression or LDA + Gaussian modeling.

slightly (i.e. this is exploited by the deltas). So an utterance can be seen as a (more or less continuous) path through this multidimensional space.

— ◇ —

Gaussian indices

We assume that this path can be discretized by keeping for each frame only the index (identity) of the Gaussian of a previously trained model, whose distance to the frame is minimal (also called the *top-1 Gaussian*). For this, typically the UBM is used.

top-N

First analyses show that using only the *top-N* Gaussians in a usual JFA based system is enough to achieve equal or similar performances as when using all Gaussians. These results are sketched in Table 8.1 for a 256 Gaussian system using the accurate JFA strategy. We observe that even when limiting to only one Gaussian, the systems does not loose too much of its power. This is a first validation step for the described assumption.

Table 8.1: *Effects of using only the top-N Gaussians, tested on NIST LRE 2005, in %pooled EER*

Top-N	1	3	6	10	15	20	30	50	100	256
%EER	11.43	9.00	8.86	8.85	8.86	8.86	8.86	8.86	8.86	8.86

transition histograms

Having "decoded" all utterances into sequences of Gaussian indices, we may analyze these sequences. For instance, we may calculate, for each language separately, the frequencies of transitions from one index to another along these paths. During testing, we may re-use these statistics to calculate a Likelihood score (in a straightforward manner).

dwelling durations

As a next step in development, we may build statistics on the duration a path "dwells" around a given Gaussian. This tracks the number of consecutive indices of the same value. For each Gaussian of each language, we will obtain a histogram of the frequency as function of the dwelling duration (this is illustrated in Fig. 8.1 (a)). We assume that the histograms for some Gaussians sensibly differ from language to language. The problem here is surely the jitter in the index sequence. We may perhaps apply some smoothing to the sequence beforehand (slightly similar to the smoothing in the SAD step, Sect. 2.3.2).

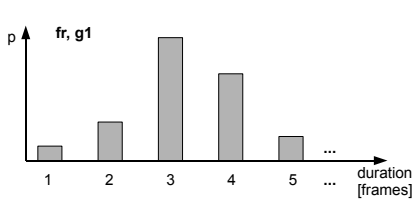
minimum duration

These duration histograms may be transformed to complementary cumulative distributions, which indicate how much time (in number of frames) the path dwells *at least* on a given Gaussian (thus the sum from n to positive infinity in the above histogram). This is called *Minimum Duration Modeling (MDM)* and other applications of it (without references here) indicate that such a distribution can typically be divided into two zones: (i) a high (cumulative) probably zone for low minimal durations (meaning that only a very few sequences are that short) and (ii) after a certain edge point, the complementary cumulation seems to decrease in an asymptotic manner. More accurately, it is most likely a complementary Gumbel

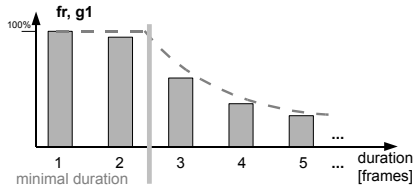
cumulative distribution (also Gumbel survival function for the maximum case)[2]. This is illustrated in Fig. 8.1 (b). We assume that this minimal duration threshold potentially changes from language to language. The modeling is thus built around this threshold[3].

As consequence, we may use this MDM to force the path to dwell on a given Gaussian for (at least) the number of frames, which corresponds to the minimal duration threshold. This forcing can be used for instance in conjunction with classic Likelihood calculation (Sect. 2.5.1), where the posterior probability γ of this possibly forced Gaussian is used. For the true language, this constraint should not induce severe penalties, but for false languages, this is likely to introduce potentially severe loss in Likelihood. MDM enforcing imposes the use of bad or very bad Likelihood values for some frames if the forced Gaussian does not correspond to the top-1 Gaussian. Fig. 8.1 shows in (c) the *top-1* Gaussian for each frame (dark-gray boxes) and the Gaussians forced by MDM (black connected boxes). forced MDM

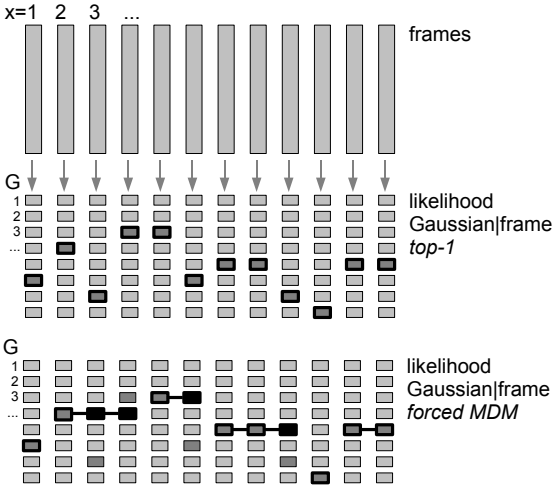
One problem of such an approach is the dependence of the starting point for the forcing — which frame we start with. We propose for instance to apply a threshold on the frame Likelihood, which has to be exceeded for the MDM to apply (on the following frames). problem



(a) Duration histogram



(b) Minimum duration



(c) Forced MDM

Figure 8.1: *Illustration of the different statistics for a given language and a given Gaussian in (a),(b) and the MDM forcing imposing penalizing Likelihoods in (c)*

[2]The Gumbel distribution is also known as the log-Weibull distribution or the type I extreme value distribution. See also http://en.wikipedia.org/wiki/Gumbel_distribution and http://www.itl.nist.gov/div898/handbook/eda/section3/eda366g.htm

[3]Apparently, some researchers choose as minimal duration threshold half the value of the average duration.

8.2.1.1 Possible implementation

In the following paragraphs, we give a sketch how MDM enforcing possibly could be implemented.

A GMM may be translated into an equivalent (ergodic) Hidden Markov Model (HMM). This HMM will have one state for each Gaussian of the GMM. The Gaussian weights will become the transition probabilities *to* the corresponding state. The path that the frames describe thus becomes a path through the HMM. Using the *top-1* Gaussian, this corresponds to a decoding using the Viterbi algorithm[4].

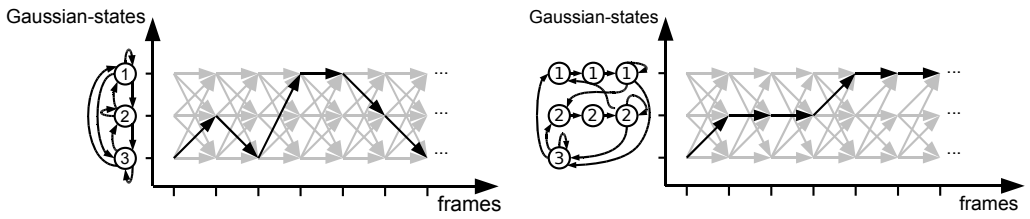
— ◇ —

implementation

Based on this concept, the MDM enforcing could be implemented in a very simple manner by duplicating each state until reaching a number equal to the minimal duration threshold. There will be only one unidirectional transition (of weight 1.0) between the duplicated states. This will hence enforce the minimum number of frames.

solution

Using a HMM based implementation has the big advantage of being independent of the starting point since the Viterbi algorithm will choose the best possible path under the MDM constraints implicitly introduced by the state duplications. Fig. 8.2 shows in (a) a three-state HMM and a possible Viterbi decoding. In (b) it shows the effect of the MDM constraint implicitly implemented in the transformed HMM.



(a) Usual Viterbi decoding (b) MDM constraint decoding

Figure 8.2: *Illustration of a GMM-corresponding HMM with an utterance path (a) unconstrained and (b) with MDM forcing*

— ◇ —

Working with Gaussian indices opens a large variety of possible approaches. Even if they move towards techniques used in phonotactics, the suggested approach has its basis in short time acoustics. Here, the system would heavily be based on the duration aspect, instead of mainly on the sequence of the index labels themselves.

min-max

The proposed MDM approach (i.e. with an HMM implementation) could even be extended to a *min-max* modelization, where there is an additional constraint

[4]Whereas the all-Gaussians approach would use the Baum-Welch algorithm.

indicating the maximum duration. As a hint, this could be implemented by replacing the state which has a (non-zero) self-transition by a series of (left-to-right) states with jumping (or by-passing) transitions (removing the self-transition avoids looping of uncontrolled length).

An extended discussion of all ideas introduced in this section has been formulated in (Verdet and Hennebert, 2010).

8.2.2 Pair-wise SVMs[5]

We investigated SVMs in Chapter 5. Since SVMs are binary classifiers, but we are working in a multi-class environment, we took the approach were we estimate the hyperplanes using one target language and putting all other languages to the negative class.

We may likewise train a set of pair-wise SVMs, meaning that their hyperplanes are estimated using SVs of two languages only: the hypothesized as positive and one other language in turn as negative class. The hypothesis is that such SVMs can be estimated more robustly than if we had SVs of $|L|-1$ languages on the negative side. pair-wise SVMs

Working with L languages, we end up with $\frac{|L| \cdot (|L|-1)}{2}$ pair-wise SVMs, if we do all combinations. As a first approach, the Likelihoods of an utterance run against all these SVMs may get concatenated into one big score vector, which can be input to a back-end (BE) to obtain a series of $|L|$ final scores. This will probably exploit the constellation of the utterance in the multidimensional space of all languages in a better way as can be achieved with one-against-others SVMs. ergodic vector

The vector of pair-wise SVM scores may also be augmented for instance by classical GMM (-JFA) scores before being input to the BE. The expected effect would be the SVM scores giving additional guidance in the case where the GMM ones are not clear enough (i.e. the highest Likelihood not standing out against the others). In order to avoid to have too large score vectors, we may also take only some selected pair SVMs. For instance those of often confound languages. As a development, we may also think of a more complex architecture which uses only a few automatically chosen SVMs. I.e. taking only those with big Likelihoods, thus those in which we have a sufficient confidence in order to enhance the GMM based detection.

These pair-wise SVMs may for instance also be exploited in some hierarchic manner, deciding at each level towards the more likely language or by introducing some bonus system, giving more weight to the languages that obtain better likelihoods against many other languages. hierarchy

[5]These pair-wise SVM based findings were sketched in an exchange with Jasmin Liu, affiliated to Ultra Electronics AudioSoft and to the University of Swansea.

8.3 More technical directions

RaSta-PLP-SDC
In Sect. 3.3.3, we compared parameterizations with different compositions of individual processing steps. From Table 3.3, we see that we missed out one interesting and promising combination. As basic type of features, PLPs seem very promising. We saw also that as well RaSta filtering normalization, as also the addition of delta-blocks (to obtain SDCs) show beneficial. The idea is then to combine these three elements to RaSta-PLP-SDC features.

Analysis of accurate JFA approach
While looking back to our work, we noticed that it would be interesting to analyze the effects of the **U** matrix rank, as it is done in Sect. 4.4.2, but for the accurate JFA approach (described in Sect. 4.3.3). The results are potentially different from those using the approximated JFA. We may guess that while lowering the number of channels (**U** rank), the accurate JFA will still work good enough for longer (lower ranks) than the approximate one. If this is the case, the justification may be that the accurate way is able to exploit the (few) channels in a better way.

Enhanced category detector

f-MAP +JFA

On the channel category detector side, the next potential step would be to include the JFA technique to the *f-MAP* detector. The disturbing variability we want to remove will then be the intra-category differences. Here, it will be constituted of the information about the language, the speaker and the fine-grained channel fluctuations.

x-* from full decomposition

As a second future direction may be imagined to design a system, which obtains cleaner category-dependent factors (**x**) for the ***x***-*MAP* and ***x***-*SVM* detectors. More category-specific **x** factors may for instance be obtained (i) by the JFA approach using intra-category variability, as described in last paragraph or (ii) maybe even better would be to use a full decomposition, as described in Sect. 4.1.1. This would separate the nuisance information into multiple terms, each of which lives in a separate subspace. One of these (specialized) subspaces will then correspond to the channel category, whereas the other disturbing information is constraint to different subspaces.

problems

The full decomposition is likely to enhance the accuracy of the channel category detectors, but at the cost of much more engineering (i.e. estimating a different matrix for each subspace). In addition, this complete decomposition will void the perspective about partitioning the training data into automatically determined, data-driven channel categories since the decomposition into multiple subspaces requires labels for the different subspaces. We may at most choose a data-driven approach as first layer (a kind of bootstrapping) to produce labels (of the channel category) and then to use a multi-subspace approach in a second turn in order to refine the categories and the recognizer.

8.4 Conclusion on perspectives

FE

Even though Feature Extraction was not part of the core subjects of this thesis, we think that some more enhancement could be reached in this domain. We may try more ingenious SDC building strategies in the neighborhood of silence segments, where frames labeled with silence are used for building delta-blocks. Likewise, we may give a try to T-DCT type features (Kinnunen et al., 2006; Castaldo et al., 2007c) since they operate comparably to the SDCs, but use all frames within the "lookahead" window and may avoid the second problem of SDCs, which is repeating delta-blocks.

SAD

We have also the feeling that we did not reach the best Speech activity detection (SAD). We showed in Sect. 2.3.2 that SAD can have a crucial impact to performance.

dev set

For being able to balance the different language recognizers among themselves, a development dataset, as well as more complex BE modules are necessary. First tries indicate however that a Gaussian Back-End alone is not able to do such inter-language normalization.

pair-wise SVMs

Although SVMs are nowadays outrun by other algorithms, we may also advance on this path. We suppose that pair-wise SVMs may give additional information in contexts of uncertainness.

Gaussian indices

A great share of research may be carried out on higher level approaches which build on the short time acoustic features we researched here. We may discretize the utterances into a sequence of Gaussian indices, representing the utterance as a trajectory in multidimensional space. Having partitioned this space into (discretized) areas, we can gather statistics on the number of consecutive frames spent in each area. This leads to Minimum Duration Modeling (MDM), which can then be used to force utterances of false languages to a penalizing trajectory.

This chapter presented some thoughts and first analyses of how future research directions, which are based on the present thesis, may look like.

Je ne vous dis plus rien ; la vertu, quand on l'aime, porte de nos bienfaits le salaire elle-même. Mon admiration, mon respect, mon amour, voilà ce que je puis vous offrir en ce jour.

— Pierre Carlet de Chamblain de MARIVAUX,
Annibal, 1720, Acte V, Scène 4

Chapter 9

Conclusion

Contents

9.1 Language recognition and variability

Language Recognition is the problem of discovering the language of a spoken utterance. If a person is familiar to the language, the recognition is quite easy. If he knows other languages of the same family, he may still guess which language it is.

A particular domain of computer science tries to recognize the language contained in a speech utterance by automatic means. Such research is part of the vast field of automatic speech processing, which also includes speech recognition and Speaker Recognition

field of research

9.1.1 Challenges and solutions

The definition of the Language Recognition (LR) task itself comes along with some challenges. Dialects, non-native accents and foreign words make it difficult.

challenges

Furthermore, it may also have to operate in a multilingual environment. To the main applications can be counted telephone based voice servers, video recording tagging and national security applications.

system characteristics

We presented some basic requirements of automatic speech processing systems and enumerated the conditions an automatic Language Recognition (LR) system should fulfill, together with the difficulties to meet them. An LR system should namely be designed in a way which makes it being independent of following points: the pronounced content, the application context, the style of speech, the acquisition and transmission, the signal quality, the set of targeted languages and the speaker. With respect to these basic requirements, it should first and foremost be easily extensible to additional languages and come along with substantial robustness to changing conditions like the speaker or the microphone. Keeping the extensibility of existing systems, this thesis focuses on the robustness of Language Recognition systems.

approaches

We gave an overview over automatic Language Recognition systems. A short outline of diverse existing approaches was given. Such systems can be distinguished by the linguistic level they operate on: acoustic systems analyzing the sound of a language, phonetic systems using occurrence statistics of phones and phonotactic systems tracking the sequence of phones/phonemes. Other approaches observe the prosody or operate on the morpho-lexical and syntactical level. For the works presented herein, acoustic systems (which operate on the lowest level) have been chosen, since they easily meet the extensibility condition.

9.1.2 Troublesome variability

variability

The robustness term is highly linked to the concept of variability. In an LR system, there is a lot of parts which may have considerable variability. This includes the inherent differences between speakers, the evolution of each speaker, the dialects or accents (in the case they are not required to be distinguished) and the variabilities due to the acquisition and transmission of the signal.

All these variabilities have a penalizing effect to automatic systems. Generally, the separability of the different classes (here the languages) is reduced by these nuisances. This is the main problem we observe in the data and it constitutes a challenging environment. The approaches investigated in this thesis follow the goal of taking into account this variability.

factor analysis

This variability problem is similar to the one that Speaker Recognition (SR) has to face, but its nature is slightly different. While the information about the speaker is the important piece in SR, it is part of the nuisance in the LR domain. Nevertheless, we adapted and applied the Joint Factor Analysis (JFA) concept, which works well in SR, to our domain. This solution consists in decomposing the information contained in the acquired signal into three components of different levels: a global component (an overall mean), a language-specific part and the language-independent nuisance part. The decomposition permits to model

explicitly this last variability part. This way, we can compensate our system for the estimated variability and obtain enhanced results.

9.1.3 Acoustic language recognition

acoustic modeling

In Chapter 2, we presented the whole processing flow of a system modeling languages on the acoustic level. As most of the typical pattern recognition systems, it is divided into two consecutive phases: the training where for each class the model parameters are estimated and the testing phase where these models are used to score data of unknown class.

Feature Extraction

In both phases, the first step always consists in Feature Extraction (FE), which cuts down the input signal into a more manageable representation. Traditionally, the frequencies contained in the signal are extracted and transformed to a cepstral representation, the feature vectors. A first step of normalizations may be carried out at this stage, like CMS, VTLN or RaSta.

training

In the training phase, one model for each class (language) is estimated. In the acoustic modeling approach we have chosen, these are Gaussian Mixture Models (GMMs), which are derived from the Universal Background Model (UBM) through a MAP criterion. For the development of the further approaches, we used GMMs of 256 components, whereas full evaluation models consisted of 2048 mixtures.

testing

In the second phase, the extracted features are scored against each model in turn. This produces a (Likelihood) score as output. Subsequent score processing steps like score normalization and calibration are generally applied before they are used to take a decision. This has the effect of putting the scores of the diverse testing utterances to a common range and to achieve some balance between the different language models.

evaluation

If the system is run in an evaluation objective, the outputs of a big number of tests are analyzed in order to give a performance measure. We also gave an overview over different ways to measure the performance. The Equal Error Rate (EER) measure is not well adapted for this, since it becomes biased if the different classes do not have the same number of testing utterances. One of the most suitable and unbiased methods is the minimal average cost ($min\text{-}C_{avg}$).

9.2 Novelties

The main challenge handled by the thesis consists in the unwanted variability contained in the data, which is mixed up with the useful language-dependent information. In a first step, we largely discussed the proposed Joint Factor Analysis (JFA) approach, which has been borrowed from researches in the Speaker Recognition domain.

We applied JFA to UBM-based GMMs using a Likelihood scoring, as well as to Support Vector Machines (SVMs) originating from such GMMs. This analyzes the possibly to use JFA as a global approach, which can get hold of diverse variabilities. In an other optic, we also probed to apply JFA by building differentiated systems, for telephone and for broadcast based sources.

9.2.1 Joint factor analysis

decomposition

In Chapter 4, we thoroughly described the concept behind JFA. The main assumption behind the JFA approach is that the information contained in the observed utterance can be separated into multiple components. The most important decomposition consists in separating the language-independent information from the language-dependent one. Additionally, this language-dependent part is further decomposed into a globally valid part and a purely language-dependent term.

decomposition implementation

Formally, this decomposition can be written as (Eq. 4.5):

$$\mathbf{m}_{(h,l)} = m + \mathbf{D}\mathbf{y}_l + \mathbf{U}\mathbf{x}_{(h,l)} \tag{9.1}$$

During the training phase, the different terms can be obtained with reasonable easiness: The term m is the global average over the data (here represented by the Universal Background Model (UBM)), $\mathbf{D}\mathbf{y}_l$ is the purely language-dependent part, obtained from the per-language average, and $\mathbf{U}\mathbf{x}_{(h,l)}$ is the language-independent term, which is the share which is neither contained in the global nor in the language part. This term depends on the utterance itself and represents the variability of the utterance (compared to the other utterances of the same language).

This per-utterance variability part is further composed of two parts: $\mathbf{U}$, which is a global (rectangular) matrix and $\mathbf{x}$, which contains the per-utterance (variability) factors. The dimensionality of $\mathbf{x}$ is low (40 in our case). $\mathbf{U}$ has the goal to constrain the session variability to this low subspace. The robustness of the $\mathbf{x}$ factor estimates is due to the global nature of $\mathbf{U}$. Since $\mathbf{U}$ is estimated on the variability parts of a large number of utterances of all languages, we can finally say that all these utterances contributed (by the fact of $\mathbf{U}$) to the estimation of the $\mathbf{x}$ factors.

During the testing phase, each utterance is considered separately. The real language-dependent information can not be identified directly. For this reason, the third (variability) term is estimated first. Due to the hypothesis that this nuisance term lives in a low-dimensional subspace, whose structure is given by $\mathbf{U}$, the variability of a single utterance can effectively be estimated[1].

algorithm

We also presented the detailed JFA algorithm, which is the basis of this work.

[1] In our eyes, it is the subspace hypothesis that makes JFA in the testing stage possible. If $\mathbf{U}$ was square (invertible) all the information would flow into this nuisance term and no information would be left for the language-dependent part (which is the term we target at). By the subspace method, the variability term can be estimated robustly and the remaining information is compared to the one of the hypothesized language.

The algorithm is presented in a manner that allows a direct implementation. It includes a preparation step where sufficient statistics are gathered, followed by an iterative cycle of centering these statistics, obtaining Point-Estimates for **x** and **y** and building the session compensation matrix **U**. This matrix is the core element of the proposed approach. Its column vectors form the basis of the variability subspace and thus it captures the structure of the variability. This structure is used again during testing to estimate the nuisance part of the data.

applying JFA

We also saw different ways to compensate our modelization system for this nuisance. Either the identified variability is removed from the feature vectors (they are cleaned up), or the general language models are compensated for the variability of the specific utterance (moved towards the data). Similarly, the compensation may take place only during the training phase, only during testing or, what works best, during both phases. The best strategy seems to require a model-space compensation during training. For the testing phase, the feature-space (and thus hybrid) method is only slightly outperformed by a model-space testing.

accurate vs. approximated

On the model-space testing side, we distinguish an accurate and an approximated way for obtaining the channel factors **x**. The accurate approach obtains **x** through the JFA formula (Eq. 4.5), where the $m + \mathbf{Dy}$ part is taken from the hypothesized language model (estimated during training) and **U** is fixed. While the approximate way just uses the UBM component m along with **U** under the form $m + \mathbf{Ux}$ for obtaining the **x** factors. The accurate model-space approach gives superior results, but at a nearly L-times higher computation cost (L being the number of possible languages). This is due to the fact that the accurate way has to estimate an **x** for each hypothesized language, whereas the approximation estimates it only once (for each testing utterance). This accurate method has not found its way to similar systems of other research groups (in the context of Speaker Recognition for instance).

results

The rank of the **U** matrix (the dimensionality of the subspace, commonly also called the number of channels) is not a very critical factor for the system's performance. Within some reasonable range, varying the rank does not have a big influence on the final system performance. However, the size of the underlying GMMs has a far higher impact. While the performance of systems using simple MAP adaptation begins to stagnate with about 512 Gaussians, the JFA approach allows to reliably estimate the means of even bigger models and thus to enhance performance. Compared to JFA-less GMMs, JFA contributes a relative cost reduction of about 61% for models with 256 components and 72% for 2048 mixtures.

conclusion

In a general manner, JFA allows to solve a severe problem of estimation theory, which is the tradeoff between the number of parameters and the amount of data required for their robust estimation. The results clearly support this fact, since JFA shows beneficial even on models of sizes where JFA-less MAP adaptation exhausts. This is also the influence of the global **U** on the robustness of the **x** factor estimates.

9.2.2 Support Vector Machines

We also analyzed the effects of JFA in the context of Support Vector Machines (SVMs). For this, the Super-Vectors (SVs) are principally obtained by stacking the mean values of a GMM one on top of the other to obtain one vector characterizing an utterance or a language. In the case of individual utterances, a GMM has been estimated (through MAP) for every utterance and transformed to an SV.

We trialed different ways to estimate SVM hyperplanes — with different combinations of target (positive) and blacklist (negative) sets. Since SVMs are put on top of the GMM approach (the GMMs becoming SVs are a new front-end), this approach does not interfere with the JFA scheme[2]. So we also tried to combine SVMs with JFA.

results

Since the SVM approach itself is rather powerful, compared to a standard GMM-UBM approach, the benefit of JFA in the context of SVM is limited. Nevertheless, this combination of SVM and JFA has been a novelty and has subsequently been attempted in the domains of Speaker Recognition and video genre recognition (Matrouf et al., 2011)

9.2.3 Channel category variability

Back to the more conceptual core of the JFA approach, we may analyze what happens when handling one kind of variability in a separate way up to a certain point. The data of the NIST LRE 2009 opened the opportunity to work on two rather different data sets, namely Conversational Telephone Speech (CTS) one one side and utterances coming from Voice of America (VOA) broadcast recordings on the other side.

seriousness

In order to analyze the severity of the problem induced by data coming from two quite different sources, we trained a dedicated system for each of these two sources (channel categories). This resulted in two completely separate pure-category systems (parts of them will be reused in later systems). By testing with one kind of data only, we observe severe performance differences between models trained on the same type of data and models trained on the other kind. The minimal average cost for such a mismatch is up to 3.7 times higher than for the corresponding case.

separate handling

In Chapter 6, we discussed how this data of two very different kinds of sources may be handled. After an explanatory description of the data and the particular challenge of the related setup, we analyzed different ways to cope with it.

feature level

A first approach corresponds to the JFA approach described until now. It pools together the data of the two sources since beginning (thus on the feature vector level). One $\mathbf{U}$ matrix on the pooled data is estimated, thus without category distinction, and one set of models are obtained.

[2]The SVs could potentially also be taken directly from JFA's internals, whose implementation operates in mean Super-Vector space.

Another approach works by taking this pooled **U** matrix, but to estimate two category-dependent sets of models. The results indicate that this approach slightly "flattens out" the discrepancy between matching and cross-category evaluations. In average, these systems are slightly better than the pure per-category ones. This is likely to be due to the extended or more robust estimation of this pooled-data **U** matrix. model level

A further attempted way takes the separate, channel category specialized **U** matrices that were obtained in the pure systems above. They can be combined together by stacking one on top of the other to form a single **U** matrix which has a CTS and a VOA dedicated part. In the same manner, the language models are trained by either channel category to obtain two sets of models. Results show that this approach has the effect of accentuating again the discrepancy between same- and cross-category evaluations. But overall, it has the big advantage of being more robust to the case where some languages have training data of only one channel category (which is the case for 15 of the 23 languages in the LRE 2009 closed-set). JFA level

As last analyzed possibility, two dedicated systems may as well be merged only on the score level. The merging is done in a very simple manner by choosing, for a given testing utterance, the score output by one or the other system. For this, we use in a first step an oracle, which reveals the correct system to look at. Here, we had also a look at two setups: using the above pure per-category systems or by merging the systems featuring a stacked **U** matrix. The latter slightly outperforms the former and it is also more robust to the problem of missing language–channel combinations. Compared to the standard pooled approach, the latter reduces the costs by 18.6% relative. This indicates that a separate handling of such big database differences (the channel categories) may be beneficial. The obtained results outperform the baseline JFA-less MAP adapted GMM-UBM system by 81% relative. score level

9.2.4 Channel category detectors

As described just above, we used an oracle for deciding which scores to choose. Even though this may be used in a study, a real system does not dispose of the correct answers to be returned by the oracle. For this reason, we also investigated ways to design automatic detection of the channel category.

Whilst the oracle, as best possible detector, dictates the ground-truth, we chose a simple-sum fusion of the two category-dependent systems as lower performance bound for comparisons. The different category detectors we designed are the following: In a first attempt, the bare feature vectors are used to train two channel category models, in a usual MAP adaptation process. These can then be used to detect the category in a classical way. category detectors

An innovative approach consists in using the channel factors issued from JFA for the task of channel category detection. These factors are the $\mathbf{x}_{(h,l)}$ vectors of the JFA formula (Eq. 4.5). So we apply the merged **U** matrix using the above formula on the speech utterances and retain only the **x** vectors. These can be used for instance channel variability based

in an SVM classifier, which discriminates between the CTS and the VOA data. It is obviously trained on the **x** vectors of the training data. Another way is to use these vectors as input features for a classical GMM MAP adaptation. Since solely one vector is obtained per utterance, models were chosen to have only 64 mixtures.

results The results show that the feature-based MAP detector and the SVM-based detector using the **x** vectors yield similar performances. The former being slightly more robust to missing category–language combinations in the training data.

On the other hand, the variability factors (**x**)-based detectors have a very important advantage: They may without any problem be used in future works to cluster and classify completely unlabeled data into some automatically determined channel categories. These categories do even not have to correspond to broad data sources like telephone or broadcast, but may also get hold of other types of variabilities, which allow to be separated into a small number of classes.

9.3 Wrap-up

Our works are based on purely acoustic Language Recognition (LR) for the simplicity to add more, possibly under-resourced languages. The main challenge of an LR unit is the variability of the data for each of the proposed languages. This variability comes under very diverse forms and from a lot of different sources, like intra- and inter-speaker variability, as well as acquisition and transmission caused fluctuations.

JFA The proposed solution to these variabilities is to decompose the total observed information into several parts: a universal (global average) part, a language-dependent and a language-independent part. This JFA approach is interesting, since it offers a way to handle all kinds of variabilities at the same time without the need to distinguish between them. This universal approach works well, but may not be the best one.

channel category For bigger types of variability, as the channel category (i.e. telephone or broadcast), we analyzed ways to work on parallel (JFA) systems and to combine them at different stages. Merging of channel category dependent systems on the score level in a fully automatic way using our designed detectors allows to approach the oracle ground-truth up to 5–6% relative. Some of the proposed detectors work by classifying the channel factors **x** into one of the two categories. Beyond interesting applications we may imagine with this technique (evoked above), it also validates the fact that the (channel) variability part is effectively caught by this **x** term.

Overall in the course of this thesis, we reduced the error rate (or cost) from 19.4 %$min\text{-}C_{avg}$ (for MAP) down to 5.4 %$min\text{-}C_{avg}$ (-72% relative) for the LRE 2005 protocol and from 20.6 %$min\text{-}C_{avg}$ down to 3.9 %$min\text{-}C_{avg}$ (-81% relative) for the LRE 2009 protocol. This shows as well the seriousness of the variability problem and at the same time it validates the JFA approach for Language Recognition.

Acknowledgments

As a slightly unusual side of this thesis can be highlighted the fact that it has been prepared under the supervision of three directors: All my gratefulness goes to the two main directors of this collaboration, M Jean-François Bonastre and M Jean Hennebert, who allowed me to conduct this thesis. Even more important to the daily evolution of this thesis has been the supervision by M Driss Matrouf, who took over the main lead at a later stage. I owe him a particular debt of gratitude.

On the financial side, the preparation of this thesis has been made possible by the French National Ministry of Education (MEN) with the contribution of the first three years and by the Department of Informatics of the University of Fribourg (DIUF) for a couple of additional months. Considerably sincere thanks to the Rectors' Conference of the Swiss Universities (CRUS) for the grant endorsing the collaboration and the joint preparation of the present thesis.

All distinct expressions shall also reach the reviewers M^{me} Régine André-Obrecht and M Pietro Laface for the serious work and the time they dedicated. Sincere appreciation also to the examiners M^{me} Lori Lamel and M Rolf Ingold. Many thanks as well to M^{lle} J.W.M. Liu for her valuable correction reading.

I enjoyed very much the outstanding ambiance at "our lab" (LIA) in Avignon with all friendly colleagues which have contributed during their time to, are still doing so or have recently joined this warm environment. All my best wishes also to the DIVA research group at the DIUF with its completely different, but also very convivial setup.

Une série de remerciements chaleureux se voient adressés à mes amis divers et variés du côté d'Avignon et de Fribourg pour leur indulgence en matière de temps que je n'ai pas su mobiliser pour eux. Ün grondischem grazcha fichun a meis genituors e mias sours pel sustegn permanent dürant quist lung temp da stüdis. Ich spreche meinen Dank auch an den erweiterten Familienkreis aus für deren Einsicht und deren Verständnis, dass diese Arbeit oft Vorrang erheilt. Tē ha'amāuruuru roa nei au ia tātou mau hoa no tō tātou taime 'ārearea o te hīmene, te 'ori e te mau paraparaura'a.

List of Symbols

C_{avg}	average detection cost, *see*: *min*-C_{avg}
C_{DET}	detection cost
C_{llr}	Log-Likelihood Ratio (average) cost
$\kappa(\blacksquare)$	kernel function used for instance in SVMs
$\varphi(\blacksquare)$	mapping to the kernel space, used for instance in SVMs
$\Sigma_{\blacksquare}$	covariance matrix of a Gaussian
$\Delta_{\blacksquare}$	delta-block constituting SDCs
$\blacksquare \cdot \blacksquare$	dot product/scalar product of two vectors, also written as $\langle \blacksquare, \blacksquare \rangle$
$e^{\blacksquare}$	exponential function
$\mathbf{1}_{\blacksquare}(\blacksquare)$	indicator function of type $\blacksquare$ for $\blacksquare$, returns 0 or 1 (or $-1/+1$)
$\blacksquare^{-1}$	inverse of matrix $\blacksquare$
min-C_{avg}	minimal average cost
$\lvert\blacksquare\rvert$	determinant of matrix $\blacksquare$
$\mu_{\blacksquare}$	mean (vector) of a Gaussian
$\lvert\blacksquare\rvert$	number of elements in/the size/cardinality of the set $\blacksquare$
$\mathcal{F}_{\blacksquare}$	objective function or criterion of type $\blacksquare$. Also called *auxiliary function*
$\pi_{\blacksquare}$	prior of language $\blacksquare$
$P_{\blacksquare}(\blacksquare)$	probability of $\blacksquare$
$\vartheta^1_{\blacksquare}$	sufficient statistics of first order
$\vartheta^2_{\blacksquare}$	sufficient statistics of second order
$\vartheta^0_{\blacksquare}$	sufficient statistics of order zero
$\blacksquare^{\mathrm{T}}$	transpose of vector or matrix $\blacksquare$
$\sigma^2_{\blacksquare}$	variance of a Gaussian
$\lvert\blacksquare\rvert_2$	Euclidean norm /2-norm of a vector $\blacksquare$
$\lvert\blacksquare\rvert$	norm of a vector $\blacksquare$
$\alpha_{\blacksquare}$	weight of a Gaussian
M	model
R	in JFA, the rank of matrix $\mathbf{U}$
Λ	model parameters
α	weighting factor for MAP, $= \frac{\vartheta^0}{\vartheta^0 + \tau}$
C	soft-margin SVM penalty controlling factor
G	number of Gaussians (in a GMM)
K	normalization exponent, *see*: Sect. 2.5.2.2
L	the set of all languages
$\mathbf{i}$	$= \sqrt{-1}$: the imaginary number
$\boldsymbol{\mathcal{X}}$	set of utterances
γ	a posteriori probability of a Gaussian for an observed feature vector

Σ^{-1}	inverse of the Co-Variance matrix
λ	Lagrangian multiplier (used for SVM margin optimization)
B	intermediary matrix in the JFA algorithm
L	intermediary matrix in the JFA algorithm
N	zeroth-order statistics for the JFA algorithm
X	first-order statistics for the JFA algorithm
$\mathcal{N}$	normal distribution
$\mathcal{X}$	utterance, represented by a set of feature vectors
σ	standard deviation, square-root of variance
D	in JFA, the square matrix, which has weighting effect since it depends on MAP's regulation factor τ: $\mathbf{D}\mathbf{D}^{\mathsf{T}} = \frac{\Sigma}{\tau}$
U	in JFA, the matrix, which projects the channel factors **x** into the SV space, has a low range
m	an SV obtained by stacking the means of one GMM
w	the normal vector of an SVM hyperplane
x	in JFA, the vector containing the channel factors and being of low dimension
y	in JFA, the vector, containing the language factors (same dimension as the SV space)
θ	threshold Upon testing, a hard decision is taken by comparing a score against the *threshold*
b	The offset of an SVM hyperplane
d	the dimension of a feature vector x
g	Gaussian of the set G of Gaussians of a GMM
h	an alternate Gaussian (as opposed to g)
k	an alternate language (as opposed to l)
l	a language (of set L)
s	score an utterance $\mathcal{X}$ obtains against a language model
t	time t
x	a feature vector
B	between class scatter matrix, in LDA
P	LDA projection matrix
V	total scatter matrix, in LDA
W	within class scatter matrix, in LDA
τ	regulation factor for MAP
$\vdash$	operator subtracting the right-hand vector from each column of the left-hand matrix
p	LDA projection vector
ξ	slack variable (used in soft-margin SVMs)
Hz	Hertz, the frequency of an occurrence [1/ s]
h	hour (unit of time, 60*60 seconds)
kHz	kilo- Hz [1000/ s]
min	minute (unit of time, 60 seconds)
ms	milli-second (1/1000 s)
s	second (unit of time)

Acronyms

ALI	Automatic Language Identification
ANN	Artificial Neural Network
APE	Applied-Probability-of-Error curve
BE	back-end
BUT	Brno University of Technology
CallFriend	LDC CallFriend corpus
cdf	cumulative distribution function
CMS	Cepstral Mean Subtraction
CSLU	Center for Spoken Language Understanding, of the OHSU
CTS	Conversational Telephone Speech
CWCC	common within-class covariance
DET	Detection Error Trade-off
DTW	Dynamic Time Warping
EER	Equal Error Rate
EM	Expectation Maximization
FE	Feature Extraction
GBE	Gaussian Back-End
GEM	Generalized EM
GMM	Gaussian Mixture Model
HLDA	Heteroscedastic Linear Discriminant Analysis
HMM	Hidden Markov Model
JFA	Joint Factor Analysis
KLD	Kullback-Leibler Divergence
LBE	Linear Back-End
LDA	Linear Discriminant Analysis
LDC	Linguistic Data Consortium, at University of Pennsylvania, http://www.ldc.upenn.edu/
LFCC	Linear Frequency Cepstral Coefficients
LIA	Laboratoire Informatique d'Avignon
LID	Language Identification
LIMSI	Laboratoire d'Informatique pour la Mecanique et les Sciences de l'Ingenieur (Computer Sciences Laboratory for Mechanics and Engineering Sciences), CNRS UPR 3251, Orsay
Lk	Likelihood
LLk	Log-Likelihood
LLR	Log-Likelihood Ratio
LPC	linear prediction coefficients
LR	Language Recognition

LRE	Language Recognition Evaluation
LVSR	Large Vocabulary Speech Recognition
MAP	Maximum A Posteriori
MDM	Minimum Duration Modeling
MFCC	Mel Frequency Cepstral Coefficients
MIT	Massachusetts Institute of Technology, Cambridge
ML	Maximum-Likelihood
MLE	ML Point-Estimate
MLR	Multi-class Logistic Regression
MLTS	Multi-Language Telephone Speech corpus, sometimes just called TS
MMI	Maximum Mutual Information
NAP	Nuisance Attribute Projection
NIST	National Institute of Standards and Technology
OGI	Oregon Graduate Institute, School of Science and Engineering, a department of OHSU, subsequently named Department of Science & Engineering, and now CSLU
OHSU	Oregon Health & Science University
PCA	Principal Components Analysis
pdf	probability density function
PLDA	Power Linear Discriminant Analysis
PLP	Perceptual Linear Prediction
pmf	probability mass function
PPR	language-dependent Parallel Phone Recognition
PPRLM	Parallel language-independent Phoneme Recognition followed by language-dependent Language Modeling
PRLM	language-independent Phoneme Recognition followed by language-dependent Language Modeling
QDA	Quadratic Discriminant Analysis
RaSta	RelAtive SpecTrAl transform
ROC	Receiver Operating Characteristics
SAD	Speech activity detection
SDC	Shifted Delta Cepstra
SR	Speaker Recognition
SV	Super-Vector
SVM	Support Vector Machine
SVR	Support Vector Regression
UBM	Universal Background Model
VOA	Voice of America
VQ	Vector Quantization
VTLN	Vocal Tract Length Normalization
WCCN	Within Class Covariance Normalization
WM	World Model

Glossary

This glossary has been gathered in different stages during the time of the works presented in this document. It may therefore lack of some homogeneity. Neither is it thought to be exhaustive nor very precise.
At the same time, it serves as index, the page numbers being listed at the end of each entry.

Artificial Neural Network
A discrimination-based method featuring one or multiple hidden layers of nodes, each containing a function which maps the input values to the output values. The nodes are interconnected between the (input, hidden, output) layers. 99, 102, 139

a posteriori
Based on the state of knowledge *after* new evidence has been observed. *see also*: a priori. 40, 42, 45, 49, 51, 52, 56, 57, 78, 89, 91, 92, 103, 109, 117, 119, 122, 123, 125–127, 132, 185, 205, 212, 214

a priori
Describing or based on a previous state, *before* making further observations which may lead to a changed view or new estimate of the problem. *see also*: a posteriori. 23, 53–56, 66, 119, 133, 169, 205

back-end
Commonly a *score back-end*.
It often tries to combine the scores an utterance obtains against all classes (instead of only the hypothesized one) in order to enhance the performance. *see also*: Linear Back-End (LBE), Gaussian Back-End (GBE). 59, 79, 98–102, 105–107, 110, 175, 180–183, 187, 189, 209, 212

Baum-Welch algorithm
also known as forward-backward algorithm. 52, 186

Bayes' Theorem
$P(B|A) = P(A|B) * P(B)/P(A)$;
$posterior = likelihood * prior/marginal\_likelihood$. 30, 41, 56, 103, 109, 205

bayesian
see: Bayes' Theorem and Sect. 2.1.
see: iterative estimation for the basic notations.
Let $\bar{\Omega}$ be the space of all possible universes $\bar{\omega}$.

(Y, μ) are two random variables or functions:
$(Y, \mu) : \bar{\Omega} \to \mathbb{R} \times \mathbb{R} \quad ; \quad (Y, \mu) : \bar{\omega} \mapsto (y, \mu)$
$P^{\bar{\Omega}}$ is the probability distribution of the $\bar{\omega}$.
P_μ is the probability distribution of y following the law $\mathcal{L}_\mu$ of μ.

$\mu \longrightarrow Y$.
a priori/prior distribution : $P(\mu = \mu)$; $\mu \sim \mathcal{N}(0, 1)$
a posteriori/posterior distribution : $P(Y = y|\mu = \mu)$; $Y \sim \mathcal{N}(\mu, \sigma^2)$

$\mathcal{L}_{\mu|Y} : A \mapsto P(\mu \in A|Y)$
prior: $\mathcal{L}_\mu : A \mapsto P(\mu \in A)$
posterior: $P(Y = y) \cdot P(\mu = \underline{\mu}|Y = y) \; = \; P(\mu = \mu, Y = y)$
$= \; P(\mu = \mu) \cdot P(Y = \underline{y}|\mu = \mu)$.

Categorical Variables
e.g. the class label.

characteristic function
Fourier transform of the probability density function (pdf). It always exists (in contrast to the pdf).
$\varphi_X(t) = E[e^{\mathbf{i}tX}] = \int_{-\infty}^{\infty} e^{\mathbf{i}tx} \, dF_X(x) = \int_{-\infty}^{\infty} e^{\mathbf{i}tx} f_X(x) \, dx = M_{\mathbf{i}X}(t) = M_X(\mathbf{i}t)$,
where $M_X(t)$ is the moment-generating function (mgf) (*see*: Generating functions). , 211, 218

Conditional Probability
$P(A|B)$; $= P(A, B)/P(B)$.

confusion matrix
A matrix presenting the percentages of identification decisions of the elected class vs. the true-class. It shows all combinations of elected class with respect of the true class.
On its diagonal, the elected class is the right one, these values thus are the per-class identification rate. The off-diagonal elements contain the false-positive (false-alarm, $P_{FA}(l, k)$) values.

Conversational Telephone Speech
Data source.
Recordings of plain old telephone system conversations. The used corpora do not contain any cellular data and they have been recorded in the USA with at least one side also located in the USA.. 10, 35, 63–66, 69, 71, 151–169, 172, 177–179, 196–198

covariance
the measure of how much two random variables vary together; $Cov(X, Y) = E((X - m) * (Y - n)) = E(X * Y) - m * n$; $m = E(X)$; $n = E(Y)$;
measure of "linear dependence". 104–106, 136

covariance matrix
matrix of covariances between elements of a vector.

cumulant
Cumulants of a pdf can be used instead of the moments for describing its shape since one determines the other (bijective relation between cumulants and moments). Often, working with cumulants is simpler (for instance because of their additive property in the context of independent random variables). The cumulants $\kappa_k = \langle X^k \rangle_c$ are defined through (the Maclaurin series of) the cumulant-generating function $g(t)$ (*see*: Generating functions):
$\kappa_1 = g'(0) \quad ; \quad \kappa_2 = g''(0) \quad ; \quad \kappa_3 = g'''(0) \quad \cdots \quad \kappa_n = g^{(n)}(0).$
In terms of central moments μ_k (and raw moment μ'_k):
$\kappa_1 = \mu_1 = \mu'_1 = \mu \quad ; \quad \kappa_2 = \mu_2 = \mu'_2 - \mu'^2_1 = \sigma^2 \quad ; \quad \kappa_3 = \mu_3 \quad ;$
$\kappa_4 = \mu_4 - 3\mu_2^2 \quad ; \quad \kappa_5 = \mu_5 - 10\mu_2\mu_2$. , 210, 215, 220

cumulative distribution function
Similar to the pmf or the pdf, but cumulating the probabilities, thus the probability of the random variable taking a value smaller as (or equal to) a given value (for the discrete and the continuous case):
$F_X(x) = P(X \leq x) = E[\mathbf{1}_{\{X \leq x\}}(X)] \quad ; \quad F(X) = \int_{-\infty}^{x} f(t)\, dt.$
see also: probability mass function (pmf), probability density function (pdf). , 218

Delta-Cepstrum
Synonym: speed and acceleration. first (and second) order polynomial cepstral coefficients in addition to the instantaneous cepstral coefficients. 88

detection
As opposed to identification. The *detection* task answers binary questions related to some hypothesized class. , 211

Detection Error Trade-off
A curve plotting the system's $P[FA]$ (false positives) against the $P[Miss]$ (false negatives) by varying the global system (score) threshold over the whole possible range. More verbosely, it shows the different system operation points with a tradeoff between high security (few false positives, but more false negatives) and high convenience (few false negatives, but more false positives). 67, 108, 109, 174, 214

distribution
An empirically or parametrically described set of values or points (either discrete observed or generated). To (parametric) probability density function count among others the normal distribution or the Gaussian distribution. 3, 9, 47, 89, 206, 209, 210, 212, 215, 216

emission
The observable outputs of probabilistic model. The hidden variables are the

parameters, which have to be estimated and which define a pdf from which the observations are thought to be drawn or emitted. 50

Equal Error Rate
see: Sect. 2.6.2.1.
Uses an a-posterior determined (global) threshold. Is particularly sensitive to unbalanced number of test segments for the different classes, since all results are pooled together, independently of the class. 67–72, 88–90, 108, 110, 129–131, 146, 184, 193, 215

estimate
The result of an estimation process or of an estimation step (e.g. iteration). It usually corresponds to updated hidden variables of a model. 52

Estimation Theory
estimating the values of parameters based on measured/empirical data; preferably, the estimator exhibits optimality; estimator estimates a models parameters.

Estimator
examples of common estimators: •Maximum likelihood estimators, •Bayes estimators, •Method of moments estimators, •Cramér-Rao bound, •Minimum mean squared error (MMSE), also known as Bayes least squared error (BLSE), •Maximum a posterior (MAP), •Minimum variance unbiased estimator (MVUE), •Best linear unbiased estimator (BLUE), •Unbiased estimators (*see*: estimator bias), •Particle filter, •Markov chain Monte Carlo (MCMC), •Kalman filter, •Ensemble Kalman filter (EnKF), •Wiener filter.

Expectation Maximization
(Algorithm) alternates between E and M steps:
•Expectation step: compute expectation of likelihood by including the latent variables as if they were observed,
•Maximization step: compute maximum likelihood estimates of the parameters by maximizing the expected likelihood;
Prominent instances of this algorithm: •Baum-Welch algorithm: applied to hidden Markov models, •inside-outside algorithm, •algorithm for fitting a mixture density model;
An EM algorithm can also find maximum a posteriori (MAP) estimates, by performing MAP estimation in the M step, rather than maximum likelihood; faster variant: OS-EM . 47, 50, 51, 53, 54, 56, 57, 91, 119, 125, 129, 215

expected value
Usually written $E[X]$. The average value we may expect from a random variable. *see also*: mean, (first) raw moment. 105, 215

Fast-Scoring
Synonym: Top-Gaussians.
For each feature vector, the C top-scoring mixtures in the background model

(UBM) are determined, then the Likelihood is computed by scoring the vector against the same C mixtures in the hypothesized target model.

feature
Parametric representation of the observed data (speech signal). 41, 71, 72, 77, 78, 80, 81, 83, 84, 87, 88, 91, 188, 189, 193, 198, 214

feature space
The domain of the transformed observed data, in contrast to model and score domain. 47, 133

feature vector
. 30, 41–46, 50–52, 56, 79, 80, 82, 83, 85–87, 90, 91, 100, 102, 106, 118, 125, 126, 129, 133, 144, 173, 183, 193, 195–197, 201, 202, 208, 212, 220, *see* vector

Fisher's Linear Discriminant
the separation between two distributions is the ratio of the variance between the classes to the variance within the classes; is, in some sense, a measure of the signal-to-noise ratio for the class labeling.

FoCal
A BE, normalization and score calibration tool developed by Niko BRÜMMER. *see*: (FoCal, 2007). *see also*: back-end (BE), normalization. 182, 212

Formants
peak in an acoustic frequency spectrum which results from the resonant frequencies of any acoustic system.

frame
Synonyms: feature vector, parametric vector. 45, 49, 61, 71, 82–87, 89, 92, 93, 95–97, 115, 118, 119, 122, 124, 125, 133, 184–186, 189, 214, 219

Gaussian
see: Gaussian distribution, Gaussian Mixture Model (GMM). 3, 9, 22, 42, 47–55, 57, 61, 81, 88–94, 97, 101–105, 118, 119, 122, 124–126, 129–133, 137, 144–148, 156, 168, 173, 176, 183–186, 189, 195, 201, 202, 209

Gaussian Back-End
A BE modeling the input scores (putting them to a vector) by means of a multivariate Gaussian. 101–103, 175, 182, 189, 205

Gaussian distribution
pdf concentrating the mass around the mean value. It can fully be described by two parameters: its mean and its (co-)variance. Named after Carl Friedrich GAUSS (1777-1855). *see also*: distribution, probability density function (pdf). 47, 134, 207, 209, 215, 216, 220

Gaussian Mixture Model
weighted combination of G unimodal Gaussian densities of dimension d. See Sect. 2.4.1 for a complete description and further pointers. *see also*: Gaussian.

9, 30, 33, 36, 37, 42, 45–47, 49–51, 53, 54, 56, 57, 76, 78, 81, 87, 90, 91, 93–97, 105, 108, 114, 117, 118, 124, 125, 127, 128, 131, 133, 139, 140, 142–149, 183, 186, 187, 193–196, 198, 201, 202, 209, 214, 215, 219, 220

Generalized EM
no need to find optimal (argmax) conditional expectation and parameters (m,s,a-priori-P), just some improvement over their current value will also ensure successful convergence. ex: improving conditional expectation with factorial distribution; improving parameters with hill-climbing method.

Generating functions
, 206, 207, 215, 220

probability-generating function (pgf)
In the context of a discrete random variable (univariate case):
$G(z) = E[z^X] = \sum_{x=0}^{\infty} p(x)z^x$;
see also: cumulant-generating function $g(t)$.

exponential generating function
$EG[a_n; x] = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$;
for the sequence $a_n = 1, 1, 1, \ldots$: $\sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = e^x$;
Note that this can be resolved to the factorial term to obtain: $n! = \int_0^{\infty} x^n e^{-x}\, dx$.
, 210

moment-generating function (mgf)
$M_X(t) = E[e^{tX}] = G(e^t) = \varphi_X(-\mathbf{i}t) = 1 + \sum_{m=1}^{\infty} \mu'_m \frac{t^m}{m!} \quad ; \quad t \in \mathbb{R}$,
where μ'_m are the raw moments. $M_X(0) = 1$;
Or viewed by the Maclaurin series (consecutive/n^{th} derivatives):
$M_X^{(n)}(0) = E[X^n] \quad ; \quad n >= 0$
(for the case the *mgf* exists on an open interval around t = 0), so it is the exponential generating function of the moments of the probability distribution. *see also*: exponential generating function (with its x replaced by tX):
$e^{tX} = \sum_{n=0}^{\infty} a_n \frac{(tX)^n}{n!} = \sum_{n=0}^{\infty} a_n X^n \frac{t^n}{n!} = 1 + tX + \frac{t^2X^2}{2!} + \frac{t^3X^3}{3!} + \cdots$;
see also: moments. , 206, 210

cumulant-generating function
Defined as the logarithm of the moment-generating function (mgf):
$g(t) = log(E[e^{tX}]) = \sum_{n=1}^{\infty} \kappa_n \frac{t^n}{n!}$.
see also: cumulant. , 207, 210, 215, 220

second cumulant-generating function
Logarithm of the characteristic function.

Generative Models
model for randomly generating observed data, typically given some hidden parameters;
defines a joint probability distribution over observation and label sequences;
examples:

- Gaussian distribution,
- Gaussian mixture model,
- Multinomial distribution,
- Hidden Markov model,
- Generative grammar,
- Naive Bayes,
- Latent Dirichlet Allocation;

If the observed data are truly generated by the generative model, then fitting the parameters of the generative model to maximize the data likelihood is a common method.;
However, data rarely truly arises from the generative models used. Therefore, it is often more accurate to model the conditional density functions directly: i.e., performing classification or regression analysis .

hidden variable
The non-observable part of a probabilistic model. The model is defined by some hidden variable parameters and possibly further fixed parameters. , 207, 208

hyperplane
A linear geometrical object extending to $(n-1)$-dimensions (a plane (2D object) for the 3D case). It is for instance used in SVMs for separating two classes. 94, 135, 136, 140, 141, 144, 146, 187, 196, 202, 219

identification
As opposed to detection, its task has to choose or elect one class among a set of possible classes. , 206, 207

iterative estimation
Let y_i be a random variable.
Y_i follows a normal distribution with unknown (hidden) mean μ: $Y_i \sim \mathcal{N}(\mu, 1)$. Y_i can be seen as function $Y_i : \Omega \to \mathbb{R}$, which maps $(\omega \mapsto y_i)$ our universe ω to concrete (the observed) values y_i.
– A Possible estimator may be $\hat{\mu} = Y_1$ or $\hat{\mu} = \frac{Y_1 + Y_2}{2}$ or $\hat{\mu} = \frac{Y_1 + Y_2 + Y_3}{3}$ or ... ($\hat{\mu}$ is also a random variable)
which associates, in the case of our universe, $\hat{\mu} \to y_1$ or ...

– Objective function: measures the goodness of an estimator:
e.g. $F(\hat{\mu}, \mu) = E[|\hat{\mu} - \mu|^2]$
Even if such a measure depends on the unknown mean, it is possible to use it because we only need to be able to compare objective function applications (i.e. from one iteration to the other), since the terms with μ possibly cancel out.
When using for instance the objective function $\frac{P_{\hat{\mu}}(Y_i = y_i)}{P_{\mu}(Y_i = y_i)}$, the denominator cancels out when comparing . , 205

Joint Probability

$P(A, B) = P(A|B) \cdot P(B)$ $\qquad$ $P(A|B) = \frac{P(A, B)}{P(B)}$.

Kullback-Leibler Divergence

Synonyms: information divergence, information gain, relative entropy.
A non-symmetric measure of the difference or distance between two probability distributions. However, it is not a true distance metric.
The problem of measuring the difference between two distributions is not easy. An often adopted replacement is the KLD.
It has been introduced by Solomon KULLBACK and Richard LEIBLER in 1951.

Approximate KLD: $D(i, j) = \sum_{g}^{G} \alpha_g \sum_{d} \left(\frac{\mu_{gd}^{i} - \mu_{gd}^{j}}{\sigma_{gd}} \right)^2$, d being the dimensions of the acoustic feature vector. 94

labeling

The action of assigning a token or class label to each slice of a segmented sequence — for instance indicating if the segment was detected as being speech or silence. 45

latent variable

synonyms: hidden variable, model parameter, hypothetical variable or hypothetical construct; variable which is not directly observed but is rather inferred from other variables that are observed and directly measured; one advantage of using latent variables is that it reduces the dimensionality of data. 50, 52, 53, 55, 118, 119, 122, 132

Likelihood

As opposed to probability, the *likelihood* is related to a conditional pdf. It thus can be seen as a kind of probability of a posterior observation (without taking into account the priors). 31, 41, 52, 53, 56, 57, 59, 72, 78, 91, 95–99, 101, 103, 106, 108–110, 127, 147, 174, 184, 185, 187, 193, 194, 209, 219

Linear Back-End

A BE featuring a linear combination of the scores input to the BE (score vector).
see also: FoCal. , 205

Linear Classifier

Often the fastest classifier; often works very well when the number of dimensions is large.
Parameter determination:

- by modeling conditional density functions:
  - Linear Discriminant Analysis (or Fisher's linear discriminant) (LDA): assumes Gaussian conditional density models; supervised learning;
  - Naive Bayes classifier: assumes independent binomial conditional density models
- by discriminative training: often yields higher accuracy:
  - Logistic regression: maximum likelihood estimation assuming that the observed training set was generated by a binomial model that depends on the output of the classifier;
  - Perceptron: attempts to fix all errors encountered in the training set;
  - Support vector machine: maximizes the margin between the decision hyperplane and the examples in the training set

.

Linear Discriminant Analysis

method to find the linear combination of features which best separate two or more classes of objects or events;
attempt to express one dependent variable as a linear combination of other features or measurements; as do: ANOVA (analysis of variance) or Regression Analysis;
closely related to:
- Principal Components Analysis (PCA): unsupervised learning; does not take into account any difference in class,
- Factor Analysis: builds the feature combinations based on differences rather than similarities;

the dependent variable is a categorical variable;
LDA explicitly attempts to model the difference between the classes of data;
a distinction between independent variables and dependent variables (also called criterion variables) must be made;
works when the measurements made on each observation are continuous quantities (vs. Discriminant Correspondence Analysis for categorical variables);
for 2-classes, LDA assumes, that both pdfs are normally distributed;
in many practical cases linear discriminants are not suitable. 49, 99, 101–104, 106, 121, 135–137, 183, 202

Linear Prediction

Mathematical operation where future values of a discrete-time signal are estimated as a linear function of previous samples. *see also*: Perceptual Linear Prediction (PLP), linear prediction coefficients (LPC). , 214

linear prediction coefficients

Synonym: Linear prediction coding.
Single-level or multilevel sampling system; predicted to be a linear function of the past values of the quantized signal. Some authors present it as being one of the most powerful speech analysis techniques and one of the most useful methods for encoding good quality speech at a low bit rate.
They estimate the formants, remove them (inverse filtering) and estimate the intensity and frequency of the remaining buzz, subtracting it gives the residue. Its values describe intensity and frequency of the buzz, the formants ("tube"), and the residue signal. They can be used to resynthesize the speech. But the encoding is very sensitive to errors. LPC usually work with generally 30 to 50 frames per second.
Other features that are more robust to errors and more stable prediction filter: Log Area Ratios (LAR), Line Spectral Pairs (LSP) decomposition and reflection coefficients.
see also: Linear Prediction. 77, 80, 81, 213

Log-Likelihood Ratio (average) cost
An empirical measure of the effective quality of information delivered to the user.
It is measured in bits of entropy. The cost is composed of *calibration loss*, which is related to the way the information is presented by the scores, and *refinement*, which effectively is the discrimination part or ability of the system.
The advantage of this performance is its independence of the application task. Hence it does not depend on a predefined operating point.
It can intuitively be seen as a measure similar to the area below the DET curve.
see: Sect. 3.6.1.

Marginal Probability
Prior Probability; probability of one event, regardless of the other event; obtained by summing (or integrating, more generally) the joint probability over the unrequired event (called marginalization); $P(A)$; $P(e) = P(R = 0, e) + P(R = 1, e) + \cdots = \sum_r P(e|R = r)P(R = r)$.

Maximum A Posteriori
Bayesian based Optimization criterion, which thus maximizes the a posteriori probability. Consequently, MAP adaptation is thought to meet the overall goal of the posterior probability, as indicated in Eq. 2.2. It can thus be seen as trying to estimate a good model for the testing phase. Further explanation on this and the ML criteria, as well as the optimization can be found in (Kamen and Su, 1999, Sect. 3.5, p. 94). Pointers to the working of MAP in the context of GMMs can be found in (Gauvain and Lee, 1994). 9, 37, 47, 50, 53–56, 72, 87, 88, 91–93, 99, 117–119, 122, 123, 125, 128, 129, 131–133, 142, 145, 147–149, 152, 155–157, 163–165, 168, 173, 176–179, 188, 193, 195–198, 201, 202, 214, 217

Maximum Margin Classifiers
see also: SVM, Chapter 5. , 219

Maximum Mutual Information
which feature(s) is (are) most discriminative for the classes; $H(C|X_i)$, H:relative entropy, C:classes, X_i:ith component of the feature vector. 37, 56, 61, 76, 87, 88, 91–93, 95

Maximum-Likelihood
Algorithms using the ML criterion require the evaluation of first and/or second derivatives of the likelihood function.
Some of such algorithms are: •gradient descent, •conjugate gradient, •variations of the Gauss-Newton method, •Expectation Maximization.
see also: Point-Estimate, Expectation Maximization (EM) Sect. 2.4.2. 47, 50–54, 56, 91–93, 95, 107, 204, 214, 215, 217

mean
Usually the *mean*/average parameter of a Gaussian distribution or a GMM. $\mu = E[X] = \langle X \rangle = \mu'_1 = \kappa_1 = g'(0)$, where $g(t)$ is the cumulant-generating function (*see*: Generating functions) and κ the cumulant. 9, 46–49, 51–54, 56, 72, 78, 87, 92, 104, 105, 117–120, 122, 125, 132, 133, 135, 144, 145, 147, 156, 183, 196, 201, 202, 208, 209, 211, 212, 216, 217, 220

Mel
The Mel scale is a non-linear mapping of the frequency scale based on perceptually equal pitch distances. Based on this fact, the word *melody* gave the name of *Mel*. It is defined as:

$$mel = \frac{1000}{\log(2)} \log\left(\frac{freq}{1000} + 1\right) = 2595 \log_{10}\left(\frac{freq}{700} + 1\right) = 1127 \log_e\left(\frac{freq}{700} + 1\right)$$

It has been developed by STEVENS, VOLKMAN and NEWMAN in 1937. 44, 72, 81–83, 85, 87, 88, 90, 204

minimal average cost
An alternative cost measure to the EER. It is insensitive to unbalanced amount of testing utterances among the different classes. As EER, it features also a global threshold. *see*: Sect. 2.6.2.2.

model
is composed of statistical samples, parameters, their probability density function (pdf) or probability mass function (pmf); ev. parameter probability distribution. 36, 66, 72, 115, 117, 125, 127

moments
Parameters measuring the shape of a distribution (usually a pdf).
see also: cumulant. , 207, 210, 220

raw moment
Synonyms: crude moment, non-central moment.
The k^{th} (raw) moment is the expected value of X^k, X being the points (discrete case) or the random variable of the distribution. This is roughly the location or translation of the distribution.

In the case f being a pdf:
$\mu'_k = E[X^k] = \langle X^k \rangle = \int_{-\infty}^{\infty} x^k \, dF_X(x) = \int_{-\infty}^{\infty} x^k f_X(x) \, dx$;
And in the discrete sample case: $\frac{1}{n}\sum_x x^k$,
where we easily can see that $\mu'_1 = \mu$ is the mean. , 207, 208, 210, 220

central moment
Are the (raw) moments about the mean μ, the (raw) moments of the differences to the mean. They describe the translation-independent shape of the distribution.
$\mu_k = E[(X - \mu)^k] = E[(X - E[X])^k] = \int_{-\infty}^{\infty} (x - \mu)^k \, dx$;
Some typical central moments: $\mu_0 = 1$; $\mu_1 = 0$; $\mu_2 = \sigma^2 = \mu'_2 - \mu^2 = \sigma^2$.
We note that the second central moment is the variance. The third central moment is the "lopsidedness" of the distribution. , 207, 216

normalized moment
Synonyms: normalized central moment, standardized moment.
Are the central moment, normalized (divided) by the "k-variance ":
$\hat{\mu}_k = \frac{\mu_k}{\sigma^k} = \frac{E[(X - \mu)^k]}{\sigma^k}$
Some typical ones have got own names:
$\hat{\mu}_1 = 0 \quad ; \quad \hat{\mu}_2 = 1 \quad ; \quad \hat{\mu}_3 = skewness \quad ; \quad \hat{\mu}_4 = Kurtosis.$

multivariate
Related to a multidimensional domain and thus working with vectors of values instead of simple (e.g. scalar) values. 47, 101

N-gram
Language model (grammar) represented by statistical syntactical rules. Containing the probability of occurrence of an ordered sequence of N symbols. Indispensable in LVSR systems and core of phonotactic systems.

Naive Bayes Assumption
assumption that the pieces of evidence are conditionally independent: $a1$ and $a2$ in: $P(B|a1, a2) = P(a1|B) * P(a2|B) * P(B)/P(a1, a2)$.

non-target
Related to a class, which is not the true one (training) or not the hypothesized one (testing). *see also*: target. , 219

normal distribution
Synonym: standard distribution. A distribution of Gaussian type having zero mean and unit variance. *see also*: distribution, Gaussian distribution. 47, 50, 117, 119, 134, 202, 207, 211, 218, *see* Gaussian distribution

normalization
Acoustic features: CMS, feature variance normalization, feature warping; BM:

Bayesian hypothesis test, likelihood ratio; Score Normalizations: 42, 58–61, 72, 78, 79, 84, 89, 99–102, 107, 108, 143, 148, 174–176, 181–183, 188, 189, 193, 209

Cnorm
Clustering used when there are several unidentified handsets; blind clustering of the normalization data; Hnorm algorithm using each cluster as a different handset. 78

Dnorm
distance normalization; normalize the score with the distance between the client and the world models; the distance is derived by comparing scores from client model scores on world model generated data with world model scores on client model generated data. 78

Hnorm
handset normalization; detect the type of handset used and apply its normalization; m,s: estimates of non-clients using the detected microphone type and client model. 78

HTnorm
handset variation of Tnorm; m,s estimated by testing the input utterance against handset-dependent impostor models. 78

Tnorm
test-normalization; matches the input utterance against a large number of non-client models. From these, the impostor mean and variance are estimated; avoids the possible acoustic mismatch between test and normalization utterances. 60, 78, 107, 182

WMAP
MAP approach on likelihood ratio World-model Maximum A Posteriori normalization. Produces a meaningful score in probability space.

Znorm
zero normalization; transform non-clients to $\mathcal{N}(0,1)$, assuming Gaussian distribution; the transformed score represents the number of standard deviations above the impostor average score. Same with the decision threshold; FAR directly defined by the decision threshold. 78, 107

Observations
also called features, attributes, variables or measurements.

Point-Estimate
The result of an estimation step, usually maximizing a certain criterion. *see also*: Maximum-Likelihood (ML), Maximum A Posteriori (MAP). 50, 53, 56, 117, 119, 122, 123, 132, 195, 204, 215

probability density function
Rarely also: probability distribution function, probability function.

It describes the probability that the continuous random variable ($f(t)$) takes a value in a given interval ($a..b$): $P(a \leq X \leq b) = \int_a^b f(x)\, dx$.
For the normal distribution : $f(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$.
It characterizes the random variable and is commonly noted $P(\blacksquare)$. It may not exist (in contrast to the characteristic function).
For the discrete case, *see*: probability density function (pdf). , 206–209, 212, 213, 215, 216, 218

probability distribution
Depending on the adopted definition or the context, this term may reference to: probability density distribution function (pdf), probability mass function (pmf) or cumulative distribution function (cdf). , 206, 211, 212, 215

probability mass function
A function returning the probability that the underlying discrete random variable takes the given value. It characterizes the random variable and is commonly noted $P(\blacksquare)$.
For the continuous case, *see*: probability density function (pdf). , 207, 215, 218

random variable
Synonym: stochastic variable.
The values of a random variable can be seen as the outcomes of a random process (an experiment).
It is a measurable function from a probability space. This means that the real outcome value of the experiment can not be told with certainty, but only its potential value. This also corresponds to possible future outcomes.
Random variables are either discrete or continuous.. 107, 206–208, 210, 211, 215, 218

range
The *range* of a matrix is the same as its *column space*, which is the set of all possible linear combinations of its column vectors. Thus, we are using the definition where the *range* of a function is the same as its image (as opposed to the definition sometimes used where it is the co-domain). The dimension of the *range* is called rank. , 218

rank
The *rank* of a matrix is at most the smaller of its dimensions (equal if the matrix has full rank). It is the maximal number of linearly independent row or column vectors, which compose the matrix. The row rank and the column rank are always the same. If we speak of *rank* in JFA, we mean the rank of the variability matrix $\mathbf{U}$, which is equal to its number of columns (the nullity is 0). It is at the same time the dimensionality of the nuisance sub-space and thus of the vectors $\mathbf{x}$. The *rank* of a matrix is thus also the dimension of its range. 104, 105, 115–118, 128–133, 135, 137, 153, 161–163, 172, 188, 195, 218

RelAtive SpecTrAl transform
A fnormalization method. It uses band-pass filtering in the log-spectral domain and removes slow channel variations in order to smooth over short-term noise variations and remove constant offsets. So it is a window-based filtering technique that adapts itself over time. RaSta filtering is described in (Hermansky et al., 1991, 1992; Hermansky and Morgan, 1994). The filter can be described as:
$H(z) = 0.1 * \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{z^{-4}(1 - 0.98z^{-1})}$
. 34, 71, 83, 84, 86, 88–91, 188, 193, 219

Spectral Envelope
Boundary of the spectral properties of a small portion of speech (usually a frame).

Support Vector
Samples along the maximum-margin hyperplanes. *see also*: Maximum Margin Classifiers.

Support Vector Machine
A discrimination-based vector classification method, featuring a non-linear decision boundary. A kernel function transforms the complex input space to a space, which is more suited for linear discrimination; Limitation: Can not handle the temporal structure of speech; A Combination with GMMs is possible: Use GMM's Likelihood values of each frame and each mixture component as an input vector for the SVM. 3, 5, 7, 9, 30, 37, 38, 46, 47, 56, 71, 76, 87, 91, 93–96, 99, 106, 133, 136, 139–149, 173, 177–179, 183, 187–189, 194, 196, 198, 201, 202, 211, 214, 219

Support Vector Regression
SVM only takes the support vectors; SVR only ignores any training data that are close to the model prediction.

target
During the training phase, an element related to the true class, and during testing, an element related to the hypothesized class. *see also*: non-target. , 209, 216

Universal Background Model
Synonyms: World Model, root model. Model commonly estimated by pooling together data of a rather large number of target class representatives. This model may then be used for being adapted to the target data or as reference for score normalization. 9, 36, 47, 50, 53–55, 70, 71, 88, 91, 93–95, 99, 100, 117–120, 122, 123, 125–127, 129, 131, 132, 142–149, 153, 158, 162, 168, 173, 176, 184, 193–195

utterance
The utterance designs an input element to our system seen from outside. It may be a phrase, speech of a defined length (i.e. some seconds) or even whole conversations. This data may be recorded beforehand, stored and handled in an offline manner or it may be recorded in an online (live) manner as stream or chunked on the fly. In the system, the utterance will be processed by sampling (windowing) it to smaller working units, the feature vectors. Commonly, the system will output a decision and/or a resulting score for a whole utterance only. 3, 9, 10, 15, 20–22, 27, 28, 31, 36, 40–42, 45, 46, 56–61, 63–67, 69–71, 77, 89, 92, 93, 97, 99, 100, 103, 107, 109, 114–122, 124, 126, 127, 129, 132, 142, 145, 147–149, 153, 155–162, 164–166, 168, 171–176, 179, 181–184, 186, 187, 189, 191, 193–198, 201, 202, 205, 215

variance
Usually the *variance* parameter of a Gaussian distribution or a GMM. More generally, the mean squared deviation (from the mean μ).
$\sigma^2 = \langle\langle X\rangle\rangle = \langle(X-\mu)^2\rangle = E[(X-\mu)^2] = E[X^2] - E[X]^2 = E[X^2] - \mu^2 = \mu_2' - \mu_1'^2 = \kappa_2 = g''(0) = \frac{1}{\sum_x p(x)} \sum_x p(x) * (x-\mu)^2$, where μ_k' are the raw moments (*see*: moments), κ_k the cumulants, and $g(t)$ is the cumulant-generating function (*see*: Generating functions). 45–47, 49, 52–54, 72, 84, 94, 103–105, 118, 131, 132, 134, 135, 156, 183, 201, 202, 209, 216, 217

vector
Synonyms: Feature vector. When speaking of a vector, in the pattern recognition domain, we usually mean a parametric vector, a multidimensional vector of parametric values, the smallest unit representing observed data. 43–46, 49, 51, 52, 81, 84–87, 96, 100–107, 117, 122, 128, 132, 135, 143, 144, 187, 196–198, 202, 209, 212, 219

Viterbi algorithm
Algorithm developed by Andrew VITERBI in 1967. For finding the most likely sequence of hidden states (Viterbi path) that result in a sequence of observed events. 52, 96, 186

Vocal Tract Length Normalization
A f-based normalization strategy to remove speaker particularities that are thought to be due to the length of its vocal tract. 34, 84, 114, 193

Voice of America
Data source.
Recordings of broadcasts emissions of the Voice of America station. Some stations feature only the language local to the geographical area, while other stations have languages mixed up.
For the NIST LRE, potentially only segments containing phone calls coming to the broadcast station are used. 10, 29, 35, 63–66, 69–71, 100, 152–169, 172, 177–179, 196–198

Bibliography

More detailed reference information such as links to abstracts and PDFs, as well as the bibliography source may be accessed online by appending the key given by the **go:** tag to the end of `http://florian.verdet.ch/thesis/bib/`.
DOI identifiers can be accessed by appending them to `http://dx.doi.org/`.
Recurrent conference proceedings have got own entries and are cross-referenced.
The section numbers where the work is cited are listed at the end of each entry.

(Abe et al., 1990) M. Abe, K. Shikano, & H. Kuwabara, 1990. Statistical Analysis of Bilingual Speaker's Speech For Cross-Language Voice Conversion. *Journal of the Acoustical Society of America (JASA) 90*(1), 76–82. go:AbeJASA90. 1.1

(Adda-Decker et al., 2003) M. Adda-Decker, F. Antoine, P. B. de Mareüil, I. Vasilescu, L. Lamel, J. Vaissiere, E. Geoffrois, & J.-S. Liénard, 2003, August 3–9. Phonetic knowledge, phonotactics and perceptual validation for automatic language identification. In (ICPhS, 2003). go:AddaICPhS03. 1.3.2

(Aizermann et al., 1964) M. Aizermann, E. Braverman, & L. Ronzonoer, 1964. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control 25*, 821–837. go:Aizermann64. 5.2

(ATVS NIST LRE, 2009) National Institute of Standards and Technology (NIST) (Ed.), 2009, June 24–25. *ATVS (Universidad Autonoma de Madrid, Spain) system*, In (NIST LRE, 2009). `http://www.itl.nist.gov/iad/mig/tests/lre/2009` (accessed: July 2010). NIST LRE 2009 participant system description., go:lre09atvs. 6.5

(Auckenthaler et al., 2000) R. Auckenthaler, M. Carey, & H. Lloyd-Thomas, 2000, January. Score Normalization for Text-Independent Speaker Verification Systems. *Digital Signal Processing 10*(1-3), 42–54, doi:10.1006/dspr.1999.0360. go:Auckenthaler00. 2.5.2.1

(Barry et al., 2003) W. J. Barry, B. Andreeva, M. Russo, S. Dimitrova, & T. Kostadinova, 2003, August 3–9. Do Rhythm Measures Tell us Anything about Language Type? In (ICPhS, 2003), 2693–2696. go:BarryICPhS03. 1.3.2, 1.3.3, 3.3.1.2

(Baum et al., 1970) L. Baum, T. Petrie, G. Soules, & N. Weiss, 1970. A maximization technique occurring in the statistical analysis of probabilistic

functions of markov chains. *Annals of Mathematical Statistics 41*, 164–171, doi:10.1214/aoms/1177697196. go:Baum70. 2.4.2

(Bayes and Price, 1763) T. Bayes & R. Price, 1763. An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S. *Philosophical Transactions 53*, 370–418, doi:10.1098/rstl.1763.0053. go:Bayes1763. 2.1

(Bellman, 1957a) R. E. Bellman, 1957. *Adaptive control processes: a guided tour*. New Jersey: Princeton University Press. go:Bellman61. 2.4.1

(Bellman, 1957b) R. E. Bellman, 1957. *Dynamic programming*. Princeton University Press. Republished: (Bellman, 2003) ([page ix]?)., go:Bellman57. 2.4.1

(Bellman, 2003) R. E. Bellman, 2003. *Dynamic programming*. Courier Dover Publications, ISBN 9780486428093. go:Bellman03. 9.3

(BenZeghiba et al., 2009) M. BenZeghiba, J.-L. Gauvain, & L. Lamel, 2009, April 19–24. Gaussian Backend design for open-set language detection. In (ICASSP, 2009), 4349–4352, doi:10.1109/ICASSP.2009.4960592. go:BenZeghibaICASSP09. 2.5.2.1, 3.5.2.2, 3.5.3

(BenZeghiba et al., 2008) M. F. BenZeghiba, J.-L. Gauvain, & L. Lamel, 2008, September 22–26. Context-dependent phone models and models adaptation for phonotactic language recognition. In (INTERSPEECH, 2008), 313–316. go:BenZeghibaIS08. 1.2.1, 3.3.4, 3.4.3.2

(Bernardo and Smith, 1994) J. M. Bernardo & A. F. M. Smith, 1994. Bayesian theory. *Statistical Methods & Applications 3*, 155–160, doi:10.1007/BF02589045. go:Bernardo94. 2.1

(Bimbot et al., 2004) F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-García, D. Petrovska-Delacrétaz, & D. A. Reynolds, 2004. A Tutorial on Text-Independent Speaker Verification. *EURASIP Journal on Applied Signal Processing, Special issue on biometric signal processing 4*, 430–451, doi:10.1155/S1110865704310024. go:BimbotASP04. 1.4.3, 2.4.3, 3.1

(Bonastre et al., 2008) J.-F. Bonastre, N. Scheffer, D. Matrouf, C. Fredouille, A. Larcher, A. Preti, G. Pouchoulin, N. Evans, B. Fauve, & J. Mason, 2008, January 21–24. ALIZE/SpkDet: a state-of-the-art open source software for speaker recognition. In Proc. of *Proceedings of Odyssey 2008 - The Speaker and Language Recognition Workshop*. International Speech Communication Association. go:BonastreOdy08. 2.7

(Bonastre et al., 2005) J.-F. Bonastre, F. Wils, & S. Meignier, 2005, March 18–23. ALIZE, a free toolkit for speaker recognition. In (ICASSP, 2005), 737–740, doi:10.1109/ICASSP.2005.1415219. go:BonastreICASSP05. 2.7

(Boser et al., 1992) B. E. Boser, I. M. Guyon, & V. N. Vapnik, 1992. A Training Algorithm for Optimal Margin Classifiers. In D. Haussler (Ed.), *5th Annual ACM Workshop on COLT'92*, 144–152. go:Boser92. 5.2.1

(Brümmer and du Preez, 2006) N. Brümmer & J. du Preez, 2006, April–July. Application-independent evaluation of speaker detection. *Computer Speech & Language 20*(2-3), 230–275, doi:10.1016/j.csl.2005.08.001. go:BrummerCSL06. 2.6.2.2, 3.6, 3.6.1

(Brümmer et al., 2009) N. Brümmer, A. Strasheim, V. Hubeika, P. Matějka, L. Burget, & O. Glembek, 2009, September 6–10. Discriminative Acoustic Language Recognition via Channel-Compensated GMM Statistics. In (INTERSPEECH, 2009), 2187–2190. go:BrummerIS09. 1.4.3, 4

(Brümmer and van Leeuwen, 2006) N. Brümmer & D. van Leeuwen, 2006, June 28–30. On calibration of language recognition scores. In (ODYSSEY, 2006), 1–8, doi:10.1109/ODYSSEY.2006.248106. go:BrummerOdy06. 2.6.2.2, 3.6.1

(Burget et al., 2009) L. Burget, M. Fapšo, V. Hubeika, O. Glembek, M. Karafiát, M. Kockmann, P. Matějka, P. Schwarz, & J. Černocký, 2009, September 6–10. BUT system for NIST 2008 speaker recognition evaluation. In (INTERSPEECH, 2009), 2335–2338. go:BurgetIS09lre. 4.1.1

(Burget et al., 2009) L. Burget, P. Matějka, V. Hubeika, & J. H. Černocký, 2009, September 6–10. Investigation into variants of Joint Factor Analysis for speaker recognition. In (INTERSPEECH, 2009), 1263–1266. go:BurgetIS09. 1.4.3

(Burget et al., 2006) L. Burget, P. Matějka, & J. Černocký, 2006, May 14–19. Discriminative training techniques for acoustic language identification. In (ICASSP, 2006), 209–212, doi:10.1109/ICASSP.2006.1659994. go:BurgetICASSP06. 1.4.3, 2.3.1, 3.3.2, 3.4.1, 3.4.1

(CallFriend, 2010) LDC, (2010). *CallFriend corpus, telephone speech of 15 different languages or dialects*. `http://www.ldc.upenn.edu/Catalog` (accessed: October 2010). go:webCallFriend. 2.6.1.1

(Campbell et al., 2006b) W. Campbell, T. Gleason, J. Navratil, D. Reynolds, W. Shen, E. Singer, & P. Torres-Carrasquillo, 2006, June 28–30. Advanced Language Recognition using Cepstra and Phonotactics: MITLL System Performance on the NIST 2005 Language Recognition Evaluation. In (ODYSSEY, 2006), 1–8, doi:10.1109/ODYSSEY.2006.248097. go:CampbellOdy06. 3.5.2.1

(Campbell et al., 2004) W. Campbell, D. Reynolds, & J. Campbell, 2004, June. Fusing discriminative and generative methods for speaker recogntion: experiments on switchboard and NFI/TNO field data. In (ODYSSEY, 2004), 41–44. go:CampbellOdy04tnorm. 2.5.2.1

(Campbell et al., 2006b) W. Campbell, D. Sturim, D. Reynolds, & A. Solomonoff, 2006, May 14–19. SVM Based Speaker Verification using a GMM Supervector Kernel and NAP Variability Compensation. In (ICASSP, 2006), 97–100. go:CampbellICASSP06svmnap. 1.4.3, 1.4.3, 4.6

(Campbell et al., 2006a) W. M. Campbell, J. Campbell, D. A. Reynolds, E. Singer, & P. A. Torres-Carrasquillo, 2006, April. Support vector machines for speaker and language recognition. *Computer Speech & Language* *20*(2–3), 210–229, doi:10.1016/j.csl.2005.06.003. go:CampbellCSL06. 2.5.2.1, 2.5.2.1

(Campbell et al., 2007) W. M. Campbell, F. Richardson, & D. A. Reynolds, 2007, April 15–20. Language Recognition with Word Lattices and Support Vector Machines. In (ICASSP, 2007), IV–989–IV–992, doi:10.1109/ICASSP.2007.367238. go:CampbellICASSP07. 3.5.2

(Campbell et al., 2004) W. M. Campbell, E. Singer, P. A. Torres-Carrasquillo, & D. A. Reynolds, 2004, June. Language Recognition with Support Vector Machines. In (ODYSSEY, 2004), 285–288. go:CampbellOdy04lrsvm. 1.2.1, 2.3.1, 3.3.2, 3.5.2.1, 3.5.3, 7.1.3

(Campbell et al., 2006a) W. M. Campbell, D. Sturim, & D. A. Reynolds, 2006, May. Support Vector Machines Using GMM Supervectors for Speaker Verification. *IEEE Signal Processing Letters* *13*(5), 308–311, doi:10.1109/LSP.2006.870086. go:CampbellSPL06srsvm. 5.2.2, 5.4

(Campbell et al., 2008) W. M. Campbell, D. E. Sturim, P. A. Torres-Carrasquillo, & D. A. Reynolds, 2008, September 22–26. A Comparison of Subspace Feature-Domain Methods for Language Recognition. In (INTERSPEECH, 2008), 309–312. go:CampbellIS08. 3.5.2.3, 4.6

(Castaldo et al., 2007a) F. Castaldo, D. Colibro, E. Dalmasso, P. Laface, & C. Vair, 2007, August 27–31. Acoustic language identification using fast discriminative training. In (INTERSPEECH, 2007), 346–349. go:CastaldoIS07. 2.5.2.1, 3.4.2, 3.4.2, 3.4.2

(Castaldo et al., 2007b) F. Castaldo, D. Colibro, E. Dalmasso, P. Laface, & C. Vair, 2007, September. Compensation of Nuisance Factors for Speaker and Language Recognition. *IEEE Transactions on Audio, Speech, and Language Processing* *15*(7), 1969–1978, doi:10.1109/TASL.2007.901823. go:CastaldoASLP07. 1.4.1, 1.4.3, 1.4.3, 2.3.1, 2.5.2.1, 3.3.2, 3.4.1, 3.4.1

(Castaldo et al., 2007c) F. Castaldo, E. Dalmasso, P. Laface, D. Colibro, & C. Vair, 2007, April 15–20. Language identification using acoustic models and speaker compensated cepstral-time matrices. In (ICASSP, 2007), IV–1013–IV–1016, doi:10.1109/ICASSP.2007.367244. go:CastaldoICASSP07. 2.5.2.1, 3.3.1.1, 3.3.1.2, 3.3.2, 4.3.2, 4.3.3, 4.4.1, 8.1.1, 8.4

(Ceppellini et al., 1955) R. Ceppellini, M. Siniscalco, & C. Smith, 1955, October. The estimation of gene frequencies in a random-mating population. *Annals of Human*

Genetics *20*(2), 97–115, doi:10.1111/j.1469-1809.1955.tb01360.x. go:Ceppellini55. 2.4.2

(Chang and Lin, 2001) C.-C. Chang & C.-J. Lin, 2001. *LIBSVM: a library for support vector machines*. http://www.csie.ntu.edu.tw/~cjlin/libsvm (accessed: July 2010). Software available online., go:Chang01libsvm. 2.7

(Charton et al., 2010) E. Charton, A. Larcher, C. Lévy, & J.-F. Bonastre, 2010, March. Mistral: open source biometric platform. In Proc. of *Proceedings of ACM Symposium on Applied Computing*, SAC '10, New York, NY, USA, 1503–1504. Association for Computing Machinery (ACM), ISBN 978-1-60558-639-7, doi:10.1145/1774088.1774411. go:ChartonACM10. 2.7

(Chen, 2010) K. Chen, 2010, November 11. *COMP61021 Modelling and Visualization of High Dimensional Data-Background*. http://www.cs.manchester.ac.uk/pgt/COMP61021/lectures/Background.pdf (accessed: December 2010). Lecture slides., go:slidesChen. 2.4.1

(Cimarusti and Ives, 1982) D. Cimarusti & R. Ives, 1982, May. Development of an automatic identification system of spoken languages: Phase I. In Proc. of *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '82)*, Volume 7, 1661–1663. doi:10.1109/ICASSP.1982.1171492. go:CimarustiICASSP82. 1.2.1, 1.3.1, 1.3.2, 3.2.1

(Cole et al., 1989) R. Cole, J. Inouye, Y. Muthusamy, & M. Gopalakrishnan, 1989, June 1–2. Language identification with neural networks: a feasibility study. In Proc. of *Communications, Computers and Signal Processing, 1989. Conference Proceeding., IEEE Pacific Rim Conference on*, 525–529. doi:10.1109/PACRIM.1989.48417. go:ColePacRim89. 3.2.2

(Corredor-Ardoy et al., 1997) C. Corredor-Ardoy, J.-L. Gauvain, M. Adda-Decker, & L. F. Lamel, 1997, September 22–25. Language Identification with Language-Independent Acoustic Models. In (EUROSPEECH, 1997), 5–8. go:CorredorES97. 3.4.3.2

(Cortes and Vapnik, 1995) C. Cortes & V. Vapnik, 1995. Support-Vector Networks. *Machine Learning* *20*(3), 273–297. go:CortesML95. 5.1.2

(Dehak et al., 2009) N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, & P. Dumouchel, 2009, September 6–10. Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification. In (INTERSPEECH, 2009), 1559–1562. go:DehakIS09. 4.1.1

(Dehak et al., 2011) N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, & P. Ouellet, 2011, May. Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing* *19*(4), 788–798, doi:10.1109/TASL.2010.2064307. go:DehakASLP11. 4.1.1

(Dempster et al., 1977) A. P. Dempster, N. M. Laird, & D. B. Rubin, 1977. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Acoustical Society of America (JASA) 39*, 1–38. go:DempsterJASA77. 2.4.2

(Dumouchel et al., 2009) P. Dumouchel, N. Dehak, Y. Attabi, R. Dehak, & N. Boufaden, 2009, September 6–10. Cepstral and Long-Term Features for Emotion Recognition. In (INTERSPEECH, 2009), 344–347. go:DumouchelIS09. 1.4.3

(EUROSPEECH, 1993) International Speech Communication Association, 1993, September 22–25. *Proceedings of Third European Conference on Speech Communication and Technology (EUROSPEECH '93)*. http://www.isca-speech.org/archive/eurospeech_1993. go:ES93. 9.3

(EUROSPEECH, 1995) International Speech Communication Association, 1995, September 18–21. *Proceedings of Fourth European Conference on Speech Communication and Technology (EUROSPEECH '95)*. http://www.isca-speech.org/archive/eurospeech_1995. go:ES95. 9.3

(EUROSPEECH, 1997) Kokkinakis, G., N. Fakotakis, & E. Dermatas (Eds.), 1997, September 22–25. *Proceedings of EUROSPEECH '97 - 5th European Conference on Speech Communication and Technology*. International Speech Communication Association. http://www.isca-speech.org/archive/eurospeech_1997. go:ES97. 9.3

(Ferguson, 1980) J. Ferguson, 1980. *Hidden Markov Models for Speech*. Institute of Defense Analyses. go:Ferguson80. 3.1

(Fisher, 1936) R. Fisher, 1936. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics 7*, 179–188. go:Fisher36. 3.5.4.1, 4.6.2

(Fletcher and McVeigh, 1992) J. Fletcher & A. McVeigh, 1992. Towards a model of segment and syllable duration in Australian English. In J. Pittam (Ed.), *Proceedings of fourth Australian International Conference on Speech Science and Technology (SST)*, 28–33. Australasian Speech Science and Technology Association (ASSTA). go:FletcherSST92. 3.3.2

(FoCal, 2007) N. Brümmer, (2007), June. *FoCal Multi-class*. http://sites.google.com/site/nikobrummer/focalmulticlass (accessed: March 2011). Software available online., go:webFoCal. 3.6.1, 1, 9.3

(Friedman, 1997) J. H. Friedman, 1997, January. On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality. *Data Min. Knowl. Discov. 1*(1), 55–77, doi:10.1023/A:1009778005914. go:Friedman97. 2.4.1

(Furui, 1994) S. Furui, 1994, April. An overview of speaker recognition technology. In Proc. of *Workshop on Automatic Speaker Recognition, Identification, Verification*, 1–9. go:Furui94. 3.1

(Furui, 2005) S. Furui, 2005, October 17–19. 50 years of progress in speech and speaker recognition. In G. Kokkinakis (Ed.), *Proceedings of 10th International Conferences "Speech and Computer" (SPECOM) 2005*, 1–9. ISBN 5-7452-0110-x. go:FuruiSPECOM05. 3.1

(Gauvain and Lee, 1994) J.-L. Gauvain & C.-H. Lee, 1994, April. Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Transactions on Speech and Audio Processing* 2(2), 291–298, doi:10.1109/89.279278. go:GauvainSAP94. 2.4.3, 2.4.4, 7.1.2, 9.3

(Geoffrois, 2004) E. Geoffrois, 2004, November 29–30. Identification automatique des langues : techniques, ressources et évaluations. In (MIDL, 2004), 43–44. go:GeoffroisMIDL04. 1.2.1, 1.2.2

(Glembek et al., 2009) O. Glembek, L. Burget, N. Dehak, N. Brümmer, & P. Kenny, 2009, April 19–24. Comparison of Scoring Methods used in Speaker Recognition with Joint Factor Analysis. In (ICASSP, 2009), 4057–4060, doi:10.1109/ICASSP.2009.4960519. go:GlembekICASSP09. 4.3.3

(Greenberg and Martin, 2009) C. Greenberg & A. Martin, 2009, June 24–25. *2009 NIST Language Recognition Evaluation - Evaluation Overview*, In (NIST LRE, 2009). http://www.itl.nist.gov/iad/mig/tests/lre/2009/lre09_eval_results/NIST_LRE09_workshop-presentation_website.pdf (accessed: July 2010). Slides presented at NIST LRE 2009 Workshop., go:slidesLRE09res. 1, 2.6.1.2

(Gutierrez-Osuna,) R. Gutierrez-Osuna. *Introduction to Pattern Recognition – Lecture 5: Dimensionality reduction (PCA)*. http://courses.cs.tamu.edu/rgutier/cs790_w02/l5.pdf . go:slidesGutierrez. 2.4.1, 4.6.1

(Hazen and Zue, 1993) T. J. Hazen & V. W. Zue, 1993, September 22–25. Automatic language Identification Using a Segment-Based Approach. In (EUROSPEECH, 1993), 1303–1306. go:HazenES93. 1.2.1, 3.4.3.2

(Hazen and Zue, 1994) T. J. Hazen & V. W. Zue, 1994, April 19–22. Recent Improvements In An Approach To Segment-Based Automatic Language Identification. In (ICASSP, 1994), 1883–1886. go:HazenICASSP94. 3.4.3.2

(Hazen and Zue, 1997) T. J. Hazen & V. W. Zue, 1997. Segment-Based Automatic Language Identification. *Journal of the Acoustical Society of America (JASA)* 18(4), 2323–2331. go:HaZu97. 1.2.1, 1.3.3, 3.3.1.2, 3.4.3.2

(Hermansky and Morgan, 1994) H. Hermansky & N. Morgan, 1994, October. RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing* 2(4), 578–589, doi:10.1109/89.326616. go:HermanskySAP94. 3.3.1.1, 3.3.1.2, 9.3

(Hermansky et al., 1991) H. Hermansky, N. Morgan, A. Bayya, & P. Kohn, 1991, November 4–6. The challenge of inverse-E: the RASTA-PLP method. In Proc. of *Signals, Systems and Computers, 1991. 1991 Conference Record of the Twenty-Fifth*

Asilomar Conference on, Volume 2, 800–804. ISBN 0-8186-2470-1, ISSN 1058-6393, doi:10.1109/ACSSC.1991.186557. go:Hermansky91. 3.3.1.1, 3.3.1.2, 9.3

(Hermansky et al., 1992) H. Hermansky, N. Morgan, A. Bayya, & P. Kohn, 1992, March 23–26. RASTA-PLP speech analysis technique. In Proc. of *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-92)*, Volume 1, 121–124. IEEE Computer Society, ISBN 0-7803-0532-9, ISSN 1520-6149, doi:10.1109/ICASSP.1992.225957. go:HermanskyICASSP92. 3.3.1.1, 3.3.1.2, 9.3

(Hieronymous and Kadambe, 1996) J. L. Hieronymous & S. Kadambe, 1996, October 3–6. Spoken Language Identification Using Large Vocabulary Speech Recognition. In Proc. of *Proceedings of the 4th International Conference on Spoken Language Processing*. go:HieronymousICSLP96. 1.2.3, 1.3.3

(House and Neuburg, 1977) A. S. House & E. P. Neuburg, 1977. Toward automatic identification of the language of an utterance. I. Preliminary methodological considerations. *Journal of the Acoustical Society of America (JASA) 62*(3), 708–713, doi:10.1121/1.381582. go:House77. 1.2.1, 3.2.1, 3.4.3.1

(Hsu et al., 2003) C.-W. Hsu, C.-C. Chang, & C.-J. Lin, 2003. LIBSVM - A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University. `http://www.csie.ntu.edu.tw/~cjlin/papers.html` (accessed: November 2010). go:Hsu03libsvm. 2.7, 5.3.1

(Hubeika et al., 2008) V. Hubeika, L. Burget, P. Matějka, & P. Schwarz, 2008, September 22–26. Discriminative Training and Channel Compensation for Acoustic Language Recognition. In (INTERSPEECH, 2008), 301–304. go:HubeikaIS08. 1.4.3

(Hynek, 1990) H. Hynek, 1990, April. Perceptual Linear Predictive (PLP) Analysis of Speech. *Journal of the Acoustical Society of America (JASA) 87*(4), 1738–1752, doi:10.1121/1.399423. go:HermanskyJASA90. 3.3.1.2

(ICASSP, 1993) IEEE Signal Processing Society, 1993, April 27–30. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-93)*. ISBN 0-7803-0946-4, ISSN 1520-6149. go:ICASSP93. 9.3

(ICASSP, 1994) IEEE Signal Processing Society, 1994, April 19–22. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-94)*. ISBN 0-7803-1775-0. go:ICASSP94. 9.3

(ICASSP, 1995) IEEE Signal Processing Society, 1995, May 9–12. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, Detroit, MI, USA. IEEE Computer Society, ISSN 1520-6149. go:ICASSP95. 9.3

(ICASSP, 2005) IEEE Signal Processing Society, 2005, March 18–23. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, Philadelphia, Pa, USA. `https://www.securecms.com/ICASSP2005`. go:ICASSP05. 9.3

(ICASSP, 2006) IEEE Signal Processing Society, 2006, May 14–19. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2006)*, Toulouse, France. http://www.icassp2006.org/. go:ICASSP06. 9.3

(ICASSP, 2007) IEEE Signal Processing Society, 2007, April 15–20. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2007)*. IEEE Computer Society, ISSN 1520-6149. go:ICASSP07. 9.3

(ICASSP, 2009) IEEE Signal Processing Society, 2009, April 19–24. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2009)*, Washington, DC, USA. IEEE Computer Society, ISSN 1520-6149. http://www.icassp09.com/GeneralInformation.asp. go:ICASSP09. 9.3

(ICPhS, 2003) M.J.Solé, D. R. & J. Romero (Eds.), 2003, August 3–9. *Proceedings of the 15th International Congress of Phonetic Science (ICPhS)*, Barcelona, Spain. Causal Productions Pty Ltd. go:ICPhS03. 9.3

(ICSLP, 1992) International Speech Communication Association, 1992, October 13–16. *Proceedings of the Second International Conference on Spoken Language Processing (ICSLP'92)*. http://www.isca-speech.org/archive/icslp_1992. go:ICSLP92. 9.3

(IFLY NIST LRE, 2009) National Institute of Standards and Technology (NIST) (Ed.), 2009, June 24–25. *IFLY (iFlyTekSpeech Lab, EEIS University of Science and Technology of China) system*, In (NIST LRE, 2009). http://www.itl.nist.gov/iad/mig/tests/lre/2009 (accessed: July 2010). NIST LRE 2009 participant system description., go:lre09ifly. 6.5

(INTERSPEECH, 2007) International Speech Communication Association, 2007, August 27–31. *Proceedings of INTERSPEECH 2007 - 8th Annual Conference of the International Speech Communication Association*. ISSN 1990-9772. http://www.isca-speech.org/archive/interspeech_2007. go:IS07. 9.3

(INTERSPEECH, 2008) International Speech Communication Association, 2008, September 22–26. *Proceedings of INTERSPEECH 2008 - 9th Annual Conference of the International Speech Communication Association*. ISSN 1990-9772. http://www.isca-speech.org/archive/interspeech_2008. go:IS08. 9.3

(INTERSPEECH, 2009) International Speech Communication Association, 2009, September 6–10. *Proceedings of INTERSPEECH 2009 - 10th Annual Conference of the International Speech Communication Association*. ISSN 1990-9772. http://www.isca-speech.org/archive/interspeech_2009. go:IS09. 9.3

(INTERSPEECH, 2010) International Speech Communication Association, 2010, September 26–30. *Proceedings of INTERSPEECH 2010 - 11th Annual Conference of the International Speech Communication Association*. http://www.isca-speech.org/archive/interspeech_2010. http://www.interspeech2010.org/. go:IS10. 9.3

(Itahashi and Du, 1995) S. Itahashi & L. Du, 1995, September 18–21. Language identification based on speech fundamental frequency. In (EUROSPEECH, 1995), 1359–1362. go:ItahashiES95. 1.3.3, 3.3.1.2

(Itakura, 1975) F. Itakura, 1975, February. Minimum prediction residual principle applied to speech recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on* *23*(1), 67–72, doi:10.1109/TASSP.1975.1162641. go:Itakura75. 3.1

(Jancik et al., 2010) Z. Jancik, O. Plchot, N. Brummer, L. Burget, O. Glembek, V. Hubeika, M. Karafiát, P. Matějka, T. Mikolov, A. Strasheim, & J. Cernocky, 2010, June 28–July 1. Data selection and calibration issues in automatic language recognition - investigation with BUT-AGNITIO NIST LRE 2009 system. In (ODYSSEY, 2010), 215–221. go:JancikODY10. 8.1

(Juang and Rabiner, 2005) B. H. Juang & L. R. Rabiner, 2005. *Automatic Speech Recognition - A Brief History of the Technology Development*, 24. Elsevier. go:Juang05. 3.1

(Kadambe and Hieronymus, 1995) S. Kadambe & J. Hieronymus, 1995, May 9–12. Language identification with phonological and lexical models. In (ICASSP, 1995), 3507–3510, doi:10.1109/ICASSP.1995.479742. go:KadambeICASSP95. 1.3.3, 3.4.4

(Kamen and Su, 1999) E. W. Kamen & J. Su, 1999. *Introduction to Optimal Estimation*. Advanced textbooks in control and signal processing. London; New York: Springer, ISBN 185233133X. go:Kamen99. 2.4.4, 9.3

(Kenny et al., 2005a) P. Kenny, G. Boulianne, & P. Dumouchel, 2005, May. Eigenvoice modeling with sparse training data. *IEEE Transactions on Speech and Audio Processing* *13*(3), 345–354, doi:10.1109/TSA.2004.840940. go:KennySAP05details. 4, 4.1, 4.2, 4.2.2, 4.2.4

(Kenny et al., 2005b) P. Kenny, G. Boulianne, P. Ouellet, & P. Dumouchel, 2005, March 18–23. Factor Analysis Simplified. In (ICASSP, 2005), 637–640, doi:10.1109/ICASSP.2005.1415194. go:KennyICASSP05fa. 1.4.3, 4, 4

(Kenny and Dumouchel, 2004) P. Kenny & P. Dumouchel, 2004, June. Experiments in speaker verification using factor analysis likelihood ratios. In (ODYSSEY, 2004), 219–226. go:KennyOdy04. 4, 4.1, 4.1, 4.1.1

(Kenny et al., 2008) P. Kenny, P. Ouellet, N. Dehak, V. Gupta, & P. Dumouchel, 2008, July. A Study of Interspeaker Variability in Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing* *16*(5), 980–988, doi:10.1109/TASL.2008.925147. go:KennyASLP08. 1.4.3, 1.4.3, 4, 4

(Kinnunen et al., 2006) T. Kinnunen, C. Wei, E. Koh, L. Wang, H. Li, & E. S. Chng, 2006, December. Temporal discrete cosine transform: Towards longer term temporal features for speaker verification. In Proc. of *Proceedings of 5th International Symposium on Chinese Spoken Language Processing (ISCSLP'2006), LNAI 4274*, 547–558. go:KinnunenISCSLP06. 3.3.1.1, 8.4

(Kockmann et al., 2009) M. Kockmann, L. Burget, & J. Černocký, 2009, September 6–10. Brno University of Technology System for Interspeech 2009 Emotion Challenge. In (INTERSPEECH, 2009), 348–351. go:KockmannIS09. 1.4.3

(Lamel and Gauvain, 1993a) L. F. Lamel & J.-L. Gauvain, 1993, April 27–30. Cross-Lingual Experiments with Phone Recognition. In (ICASSP, 1993), 507–510, doi:10.1109/ICASSP.1993.319353. go:LamelICASSP93. 3.4.3.1

(Lamel and Gauvain, 1993b) L. F. Lamel & J.-L. Gauvain, 1993, September 22–25. Identifying Non-Linguistic Speech Features. In (EUROSPEECH, 1993), 23–30. go:LamelES93. 3.4.3.1

(Lamel and Gauvain, 1994) L. F. Lamel & J.-L. Gauvain, 1994, April 19–22. Language Identification Using Phone-based Acoustic Likelihoods. In (ICASSP, 1994), I–293–I–296. go:LamelGauvain94. 1.2.1, 1.2.3, 3.4.3.2

(Lander et al., 1995) T. L. Lander, R. Cole, B. Oshika, & M. Noel, 1995, September 18–21. The OGI 22 Language Telephone Speech Corpus. In (EUROSPEECH, 1995), 817–820. go:Lander95. 1.2.1

(Laplace, 1986) P.-S. Laplace, 1774/1986. Memoir on the Probability of the Causes of Events. *Statistical Science 1*(3), 364–378, doi:10.1214/ss/1177013621. go:Laplace1774. 2.1

(Lawrence and Barker, 2009) N. D. Lawrence & J. Barker, 2009, September 6–10. *Dealing with High Dimensional Data with Dimensionality Reduction, Interspeech Tutorial*, In (INTERSPEECH, 2009). `ftp://ftp.dcs.shef.ac.uk/home/neil/interspeech09.pdf` (accessed: December 2010). Presentation slides., go:slidesLawrenceIS09. 2.4.1

(Leonard and Doddington, 1974) R. G. Leonard & G. R. Doddington, 1974, August. *Automatic Language Identification* (Final Report, RADC- TR–74–200/TI-347650 ed.). Dallas, TX: Texas Instruments Inc Dallas. Final rept. Aug 72–Feb 74 ; 785397/1GI., go:Leonard74. 1.2.1, 3.2.1

(Li and Edwards, 1980) K. Li & T. Edwards, 1980, April 9–11. Statistical models for automatic language identification. In Proc. of *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '80)*, Volume 5, 884–887. doi:10.1109/ICASSP.1980.1170832. go:LiICASSP80. 3.2.1, 3.4.3.1

(Li, 1994) K.-P. Li, 1994, April 19–22. Automatic language identification using syllabic spectral features. In (ICASSP, 1994), I/297–I/300, doi:10.1109/ICASSP.1994.389372. go:LiICASSP94. 1.1

(Louradour, 2007) J. Louradour, 2007, January 25. *Nouyaux de séquences pour la vérification du locuteur par Machines à Vecteurs de Support*. Ph. D. thesis, IRIT, Université de Toulouse III – Paul Sabatier. go:LouradourPhD07. 5.2.2

(Martin et al., 1997) A. Martin, G. Doddington, T. Kamm, M. Ordowski, & M. Przybocki, 1997, September 22–25. The DET Curve in Assessment of Detection Task Performance. In (EUROSPEECH, 1997), 1895–1898. go:Martin97. 2.6.2

(Matrouf et al., 2007) D. Matrouf, N. Scheffer, B. Fauve, & J.-F. Bonastre, 2007, August 27–31. A Straightforward and Efficient Implementation of the Factor Analysis Model for Speaker Verification. In (INTERSPEECH, 2007), 1242–1245. go:MatroufIS07straight. 1.4.3, 4, 4, 4.4.2

(Matrouf et al., 2011) D. Matrouf, F. Verdet, M. Rouvier, J.-F. Bonastre, & G. Linarès, 2011. Modeling Nuisance Variabilities with Factor Analysis for GMM-based Audio Pattern Classification. *Computer Speech & Language* *25*(3), 481–498, doi:10.1016/j.csl.2010.11.001. go:MatroufVerdetCSL10. 1.4.3, 1.6, 4, 5, 9.2.2

(Matějka et al., 2006) P. Matějka, L. Burget, P. Schwarz, & J. Černocký, 2006, June 28–30. Brno University of Technology System for NIST 2005 Language Recognition Evaluation. In (ODYSSEY, 2006), 57–64. go:MatejkaOdy06. 1.4.1, 2.3.1, 2.5.2.2, 24, 3.3.2, 3.4.1, 3.4.1, 3.4.1

(Matějka et al., 2005) P. Matějka, P. Schwarz, J. Černocký, & P. Chytil, 2005. Phonotactic Language Identification. In Proc. of *Proceedings of Radioelektronika 2005*, 140–143. Faculty of Electrical Engineering and Communication BUT, ISBN 80-214-2904-6. go:Matejka05phlid. 3.4.3.2

(MIDL, 2004) ENST — Télécom Paris, 2004, November 29–30. *Identification des langues et des variétés dialectales par les humains et par les machines — MIDL 2004 : Modélisation pour l'identification des langues*, Paris, France. ISSN 1242-5125. `http://www.limsi.fr/MIDL/actes/`. go:MIDL04. 9.3

(Milner and Vaseghi, 1995) B. P. Milner & S. V. Vaseghi, 1995, September 18–21. An analysis of cepstral-time matrices for noise and channel robust speech recognition. In (EUROSPEECH, 1995), 519–522. go:MilnerES95. 3.3.1.2

(MISTRAL, 2009) A. Laboratoire Informatique d'Avignon (LIA), (2009), February 13. *The MISTRAL project, open source platform for biometrics authentification*. `http://mistral.univ-avignon.fr` (accessed: October 2010). Software available online., go:webMistral. 2.7

(Moschitti, 2010a) A. Moschitti, 2010, August. Kernel engineering for fast and easy design of natural language applications. In Proc. of *Coling 2010: Kernel Engineering for Fast and Easy Design of Natural Language Applications–Tutorial notes*, Beijing, China, 1–91. Coling 2010 Organizing Committee. go:MoschittiCOLE10. 5.2

(Moschitti, 2010b) A. Moschitti, 2010, September 26–30. *Kernel Engineering for Fast and Easy Design of Natural Language Applications, Interspeech Tutorial*, In (INTERSPEECH, 2010). `http://www.interspeech2010.org/TutorialsT-S3-R1.html` (accessed: March 2011). Presentation slides., go:slidesMoschittiIS10. 5.2

(Muthusamy and Cole, 1992) Y. Muthusamy & R. A. Cole, 1992, October 13–16. Automatic segmentation and identification of ten languages using telephone speech. In (ICSLP, 1992). go:MuthusamyICSLP92b. 1.2.1, 3.2.2, 3.4.3.1

(Muthusamy et al., 1992) Y. Muthusamy, R. A. Cole, & B. T. Oshika, 1992, October 13–16. The OGI Multi-Language Telephone Speech Corpus. In (ICSLP, 1992), 895–898. go:MuthusamyICSLP92a. 1.2.1

(Muthusamy et al., 1993) Y. K. Muthusamy, K. M. Berkling, T. Arai, R. A. Cole, & E. Barnard, 1993, September 22–25. A Comparison of Approaches to Automatic Language Identification Using Telephone Speech. In (EUROSPEECH, 1993), 1307–1310. go:MuthusamyES93. 3.4.3.1

(Muthusamy et al., 1994) Y. K. Muthusamy, N. Jain, & R. A. Cole, 1994, April 19–22. Perceptual benchmarks for automatic language identification. In(ICASSP, 1994), 333–336, doi:10.1109/ICASSP.1994.389288. go:MuthusamyICASSP94. 1.3.2

(NIST LRE, 2005) National Institute of Standards and Technology (NIST) (Ed.), 2005, October 6. *The 2005 NIST Language Recognition Evaluation, Evaluation Plan*. http://www.itl.nist.gov/iad/mig/tests/lre/2005 (accessed: July 2010). Workshop: (NIST LRE Workshop, 2005)., go:webNIST05lre. 1, 2.6.1.2, 2.6.3.1

(NIST LRE, 2009) National Institute of Standards and Technology (NIST) (Ed.), 2009. *The 2009 NIST Language Recognition Evaluation (LRE09), Evaluation Plan*. http://www.itl.nist.gov/iad/mig/tests/lre/2009 (accessed: July 2010). Results: (NIST LRE Results, 2009); Workshop: (NIST LRE Workshop, 2009)., go:webNIST09lre. 1.2.1, 1.4.2, 1, 2.6.1.2, 2.6.2.2, 2.6.3.2, 6, 9.3

(NIST LRE Results, 2009) National Institute of Standards and Technology (NIST) (Ed.), 2009, June 24–25. *The 2009 NIST language recognition evaluation results*, In (NIST LRE, 2009). http://www.itl.nist.gov/iad/mig/tests/lre/2009/lre09_eval_results (accessed: July 2010). Slides presenting evaluation results., go:webNIST09lreResults. 9.3

(NIST LRE Workshop, 2005) NIST LRE Workshop, 2005, December 6–7. The 2005 NIST Language Recognition Evaluation, Workshop. http://www.itl.nist.gov/iad/mig/tests/lre/2005 (accessed: July 2010). go:webNIST05lreWorkshop. 9.3

(NIST LRE Workshop, 2009) NIST LRE Workshop, 2009, June 24–25. The 2009 NIST Language Recognition Evaluation (LRE09), Workshop. http://www.itl.nist.gov/iad/mig/tests/lre/2009 (accessed: July 2010). go:webNIST09lreWorkshop. 9.3

(ODYSSEY, 2004) Ortega-García, J., J. González-Rodréguez, F. Bimbot, J.-F. Bonastre, J. Campbell, I. Magrin-Chagnolleau, J. Mason, R. Peres, & D. Reynolds (Eds.), 2004, June. *Proceedings of ODYSSEY 2004 - The Speaker and Language Recognition Workshop*. International Speech Communication Association. http://www.isca-speech.org/archive_open/odyssey_04. go:ODY04. 9.3

(ODYSSEY, 2006) International Speech Communication Association, 2006, June 28–30. *Proceedings of IEEE Odyssey 2006 - The Speaker and Language Recognition Workshop*. go:ODY06. 9.3

(ODYSSEY, 2010) International Speech Communication Association, 2010, June 28–July 1. *Proceedings of Odyssey 2010 - The Speaker and Language Recognition Workshop*. http://www.isca-speech.org/archive_open/odyssey_2010. go:ODY10. 9.3

(Pellegrino et al., 1999a) F. Pellegrino, J. Farinas, & R. André-Obrecht, 1999, September 5–9. Comparison of two phonetic approaches to language identification. In Proc. of *Proceedings of Sixth European Conference on Speech Communication and Technology (EUROSPEECH'99)*, 399–402. International Speech Communication Association. go:PellegrinoES99. 1.3.3, 3.4.3.2

(Pellegrino et al., 1999b) F. Pellegrino, J. Farinas, & R. André-Obrecht, 1999, September 13–14. Vowel System Modeling: A Complement to Phonetic Modeling in Langage Identification. In Proc. of *Multi-Lingual Interoperability in Speech Technology11, (RTO MP-28)*, 119–124. RTO/NATO 2000, ISBN 92-837-1044-4. go:PellegrinoRTO99. 1.3.3, 3.4.3.2, 3.4.3.2

(Plchot et al., 2009) O. Plchot, V. Hubeika, L. Burget, P. Schwarz, P. Matějka, & J. Cernocky, 2009, January. Acquisition of Telephone Data from Radio Broadcasts with Application to Language Recognition. Technical report, Speech@FIT. http://www.nist.gov/speech/tests/lre/2009/radio_broadcasts.pdf (accessed: November 2010). Distributed online., go:Plchot09. 2.6.1.1

(Povey and et al., 2010) D. Povey & et al., 2010, March 14–19. Subspace Gaussian Mixture Models for Speech Recognition. In Proc. of *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2010)*, 4330–4333. IEEE Signal Processing Society, ISSN 1520-6149. go:PoveyICASSP10. 1.4.3

(Rabiner, 1989) L. R. Rabiner, 1989, February. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–286, doi:10.1109/5.18626. go:RabinerSAP89hmm. 3.1

(Reynolds, 1995) D. A. Reynolds, 1995. Speaker identification and verification using gaussian mixture speaker models. *Speech Communication 17*(1–2), 91–108, doi:10.1016/0167-6393(95)00009-D. go:ReynoldsSC95. 2.4.3

(Reynolds, 1997) D. A. Reynolds, 1997, September 22–25. Comparison of background normalization methods for text-independent speaker verification. In (EUROSPEECH, 1997), 963–966. go:ReynoldsES97. 2.4.3

(Reynolds et al., 2000) D. A. Reynolds, T. F. Quatieri, & R. B. Dunn, 2000, January. Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing 10*(1), 19–41. go:ReynoldsDSPJ00. 2.4.3, 2.4.4, 4.1, 4.1

(Riek et al., 1991) L. Riek, W. Mistretta, & D. Morgan, 1991, December. *Experiements in language identification* (Technical Report SPCOT-91-002 ed.). Nashua, NH: Lockheed Sanders, Inc. go:Riek91. 1.2.1, 3.2.2

(Rose and Reynolds, 1990) R. C. Rose & D. A. Reynolds, 1990, April 3–6. Text independent speaker identification using automatic acoustic segmentation. In Proc. of *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-90)*, Volume 1, 293–296. IEEE Computer Society, ISSN 1520-6149, doi:10.1109/ICASSP.1990.115638. go:RoseICASSP90. 3.1

(Rouas and Farinas, 2004) J.-L. Rouas & J. Farinas, 2004, November 29–30. Comparaison de méthodes de caractérisation du rythme des langues. In (MIDL, 2004), 45–50. go:RoFa2004.1. 1.3.2, 3.1

(Rouas et al., 2003) J.-L. Rouas, J. Farinas, & F. Pellegrino, 2003, August 3–9. Automatic Modelling of Rhythm and Intonation for Language Identification. In (ICPhS, 2003), 567–570. go:RouasICPhS03. 1.3.3, 3.3.1.2

(Rouas et al., 2005) J.-L. Rouas, J. Farinas, F. Pellegrino, & R. André-Obrecht, 2005. Rhythmic unit extraction and modelling for automatic language identification. *Speech Communication* *47*(4), 436–456, doi:10.1016/j.specom.2005.04.012. go:RouasSC05. 1.3.2

(Roweis, 1998) S. Roweis, 1998. Em algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in Neural Information Processing Systems 10 (NIPS'97)*, Volume 10, Cambridge, MA, USA, 626–632. The MIT Press, ISBN 0-262-10076-2. (The published version has an error in equation for e^{new} on the bottom of p6 which has been fixed in these online versions. Thanks to David Ross and others for pointing this out.)., go:RoweisNIPS98. 4.6.1

(Sakai et al., 2008) M. Sakai, N. Kitaoka, & S. Nakagawa, 2008. Linear Discriminant Analysis Using a Generalized Mean of Class Covariances and Its Application to Speech Recognition. *IEICE Transactions* *91-D*(3), 478–487, doi:10.1093/ietisy/e91-d.3.478. go:SakaiIEICE08. 3.5.4.1, 4.6.2

(Savic et al., 1991) M. Savic, E. Acosta, & S. Gupta, 1991, April 14–17. An automatic language identification system. In Proc. of *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-91)*, Volume 2, Los Alamitos, CA, USA, 817–820. IEEE Computer Society, ISBN 0-7803-0003-3, ISSN 1520-6149, doi:10.1109/ICASSP.1991.150462. go:SavicICASSP91. 1.2.1, 3.2.2

(Scheffer, 2006) N. Scheffer, 2006. *Structuration de l'espace acoustique par le modèle générique pour la vérification du locuteur*. Ph. D. thesis, Université d'Avignon et des Pays de Vaucluse, Laboratoire Informatique d'Avignon, France. go:SchefferTh06. 2.3.2

(Singer et al., 2003) E. Singer, P. Torres-Carrasquillo, T. Gleason, W. Campbell, & D. Reynolds, 2003, September 1–4. Acoustic, phonetic,and discriminative

approaches to automatic language identification. In Proc. of *Proceedings of 8th European Conference on Speech Communication and Technology (EUROSPEECH 2003 - INTERSPEECH 2003)*, 1345–1348. International Speech Communication Association. go:SingerIS03. 1.2.1, 3.4.3.2, 3.5.2.1, 7.1.3

(SPro, 2004) G. Gravier, (2004), March 5. *SPro, a free speech signal processing toolkit*. http://www.irisa.fr/metiss/guig/spro (accessed: July 2010). Software available online., go:webSPro. 2.7

(Tipping and Bishop, 1999a) M. E. Tipping & C. M. Bishop, 1999. Mixtures of probabilistic principal component analysers. *Neural Computation 11*, 443–482. go:TippingNC99. 4.6.1

(Tipping and Bishop, 1999b) M. E. Tipping & C. M. Bishop, 1999. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society, Series B 61*(3), 611–622, doi:10.1111/1467-9868.00196. go:TippingJRSS99. 4.6.1

(Torres-Carrasquillo et al., 2002a) P. A. Torres-Carrasquillo, D. A. Reynolds, & J. Deller, 2002, May 13–17. Language Identification using Gaussian Mixture Model Tokenization. In Proc. of *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2002)*, Volume 1, 757–760. IEEE Signal Processing Society. go:TorresICASSP02. 3.4.3.2

(Torres-Carrasquillo et al., 2002b) P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, & J. J.R. Deller, 2002, September 16–20. Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features. In J. H. L. Hansen & B. Pellom (Eds.), *Proceedings of 7th International Conference on Spoken Language Processing (ICSLP2002 - INTERSPEECH 2002)*, 89–92. doi:10.1.1.112.4442. go:TorresICSLP02. 1.2.1, 2.3.1, 3.3.2, 3.4.3.2

(Tucker et al., 1994) R. Tucker, M. Carey, & E. Parris, 1994, April 19–22. Automatic language identification using sub-word models. In (ICASSP, 1994), I/301–I/304 vol.1, doi:10.1109/ICASSP.1994.389295. go:TuckerICASSP94. 3.4.3.2, 3.4.3.2

(Vair et al., 2006) C. Vair, D. Colibro, F. Castaldo, E. Dalmasso, & P. Laface, 2006, June 28–30. Channel Factors Compensation in Model and Feature Domain for Speaker Recognition. In (ODYSSEY, 2006), 1–6, doi:10.1109/ODYSSEY.2006.248117. go:VairOdy06. 4.1, 4.3.3, 4.4.1, 4.6

(Vaissière and Boula de Mareüil, 2004) J. Vaissière & P. Boula de Mareüil, 2004, November 29–30. Identifying a language or an accent: from segments to prosody. In (MIDL, 2004), 1–5. go:VaissiereMIDL04. 4

(van Leeuwen and Brümmer, 2006) D. A. van Leeuwen & N. Brümmer, 2006, June 28–30. Channel-dependent GMM and Multi-class Logistic Regression models for language recognition. In (ODYSSEY, 2006), 1–8, doi:10.1109/ODYSSEY.2006.248094. go:LeeuwenOdy06. 3.3.2, 3.5.6

(Velichko and Zagoruyko, 1970) V. M. Velichko & N. G. Zagoruyko, 1970. Automatic Recognition of 200 words. *International Journal of Man–Machine Studies* 2(3), 223–234, doi:10.1016/S0020-7373(70)80008-6. go:Velichko70. 3.1

(Verdet, 2005) F. Verdet, 2005, November. Automatic Language Identification System Based on Automatic Speech Segmentation Tools. Master's thesis, University of Fribourg, Switzerland, DIVA group, DIUF, Université de Fribourg, Bd de Pérolles 90, CH-1700 Fribourg, Switzerland. go:VerdetTMSc05. 1.3.3, 13, 3.4.3.2

(Verdet et al., 2005) F. Verdet, A. El Hannani, J. Hennebert, & D. Petrovska, 2005, December 6–7. UNIFRI-INT Language Recognition System Description. Technical report, NIST Language Recognition Evaluation 2005 Workshop, Washington, DC, USA. go:VerdetLRE05. 3.4.3.2

(Verdet and Hennebert, 2010) F. Verdet & J. Hennebert, 2010, April 9. Formulation de quelques idées : Le segment en tant que parcours dans l'espace acoustique discrétisé par les indices de Gaussiennes. Technical report, DIUF, Université de Fribourg et LIA, Université d'Avignon et des Pays de Vaucluse. go:Verdet10gix. 8.2.1.1

(Verdet et al., 2009a) F. Verdet, D. Matrouf, & J.-F. Bonastre, 2009, June 24–25. NIST LRE 2009 - LIA System Description. Technical report, NIST Language Recognition Evaluation 2009 Workshop, Baltimore, USA. go:VerdetLRE09. 1.6, 2.6.3

(Verdet et al., 2009b) F. Verdet, D. Matrouf, J.-F. Bonastre, & J. Hennebert, 2009, September 6–10. Factor Analysis and SVM for Language Recognition. In (INTERSPEECH, 2009), 164–167. go:VerdetIS09. 1.2.1, 1.4.1, 1.4.3, 1.6, 4, 7.1.3

(Verdet et al., 2010a) F. Verdet, D. Matrouf, J.-F. Bonastre, & J. Hennebert, 2010, September 26–30. Channel Detectors for System Fusion in the Context of NIST LRE 2009. In (INTERSPEECH, 2010), 733–736. go:VerdetIS10. 1.6, 4

(Verdet et al., 2010b) F. Verdet, D. Matrouf, J.-F. Bonastre, & J. Hennebert, 2010, June 28–July 1. Coping with Two Different Transmission Channels in Language Recognition. In (ODYSSEY, 2010), 230–237. go:VerdetOdy10. 1.4.1, 1.6, 4, 7.1

(Vogt et al., 2005) R. Vogt, B. Baker, & S. Sridharan, 2005, September 4–8. Modelling Session Variability in Text-Independent Speaker Verification. In Proc. of *Proceedings of Interspeech'2005 - Eurospeech, 9th European Conference on Speech Communication and Technology*, 3117–3120. International Speech Communication Association. go:VogtIS05msv. 1.4.3, 4

(Vogt et al., 2008) R. Vogt, B. Baker, & S. Sridharan, 2008. Explicit modeling of session variability for speaker verification. *Computer Speech & Language* 22(1), 17–38. go:VogtCSL08. 1.4.3, 4.1, 4.2

(Yan and Barnard, 1995) Y. Yan & E. Barnard, 1995, May 9–12. An approach to automatic language identification based on language-dependent phone recognition. In (ICASSP, 1995), 3511–3514, doi:10.1109/ICASSP.1995.479743. go:YanICASSP95. 3.4.3.2

(Yan et al., 1996) Y. Yan, E. Barnard, & R. A. Cole, 1996. Development of an Approach to Automatic Language Identification based on Phone Recognition. *Computer Speech & Language* *10*(1), 37–54, doi:10.1006/csla.1996.0003. go:YanBaCo96. 1.2.1, 1.3.3, 3.3.1.2, 3.4.3.2

(Zissman, 1993) M. A. Zissman, 1993, April 27–30. Automatic Language Identification using Gaussian Mixture and Hidden Markov Models. In (ICASSP, 1993), 399–402, doi:10.1109/ICASSP.1993.319323. go:Zissman93. 1.2.1, 3.2.2, 3.4.3.1, 3.4.3.2

(Zissman, 1996) M. A. Zissman, 1996, January. Comparison of Four Approaches to Automatic Language Identification of Telephone Speech. *IEEE Transactions on Speech and Audio Processing* *4*(1), 31–44, doi:10.1109/TSA.1996.481450. go:Zissman96. 1.2.1, 3, 3.4.3.2, 3.2, 3.4.3.2, 3.4.3.2, 3.5.2.1

(Zissman and Berkling, 2001) M. A. Zissman & K. M. Berkling, 2001. Automatic language identification. *Speech Communication* *35*(1–2), 115–124, doi:10.1016/S0167-6393(00)00099-6. go:ZissmanSC01. 3.4.3.2, 3.3

(Zissman and Singer, 1994) M. A. Zissman & E. Singer, 1994, April 19–22. Automatic Language Identification of Telephone Speech Messages Using Phoneme Recognition and N-Gram Modeling. In (ICASSP, 1994), 305–308, doi:10.1109/ICASSP.1994.389377. go:Zissman94. 1.2.1, 3.4.3.1, 3.4.3.2, 3.4.3.2, 3.4.3.2

Macht Eure Träume wahr!

Réalisez vos rêves !

Realisai voss sömmis !

'A fa'ariro to'outou mau moemoeā !

Make your dreams become true!

— Toni 'El Suizo' Rüttimann
builder of emergency bridges
Pontresina, Switzerland