
A Speaker Classification Framework for Non-intrusive User Modeling: Speech-based Personalization of In-Car Services

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I der Universität des Saarlandes

vorgelegt von
Michael Feld

Saarbrücken,
28. November 2011

Datum des Kolloquiums:

23. Dezember 2011

Dekan:

Prof. Dr. Holger Hermanns

Vorsitzender der Prüfungskommission:

Prof. Dr. Antonio Krüger

Gutachter:

Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster

Prof. Dr. Bernd Möbius

Akademischer Beisitzer:

Dr.-Ing. Christian Müller

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Diese Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Saarbrücken, den 28. November 2011

Michael Feld

Danksagung

Zuallererst möchte ich Herrn Professor Wahlster ganz herzlich danken für die Vergabe und Unterstützung meines Promotionsthemas. Durch die sinnvolle Verknüpfung mit spannenden praktischen Tätigkeitsfeldern am Deutschen Forschungszentrum für Künstliche Intelligenz war es mir über den gesamten Zeitraum der Arbeit möglich, auf ausreichend Motivation, Ressourcen, Erfahrung und Unterstützung zurückzugreifen. Zahlreiche persönliche Anregungen, aber auch konstruktive Kritiken, haben meine Bemühungen zusätzlich beflügelt.

Gleich im Anschluss möchte ich auch Christian Müller ganz besonders für die langjährige Begleitung meiner Forschungstätigkeiten danken. Seine Arbeiten im Bereich der Sprecherklassifikation dienten nicht nur als Inspiration für die vorliegende Arbeit, sondern seine Ratschläge und Ideen haben auch meine eigenen Methoden und Entscheidungen schon seit der Zeit vor meiner Diplomarbeit wesentlich mitgeprägt.

Des Weiteren danke ich meinen Kolleginnen und Kollegen am Lehrstuhl von Herrn Professor Wahlster und am DFKI, ganz besonders in der Automotive Gruppe, die für das hervorragende Arbeitsklima, die Unterstützung in vielen praktischen Dingen, und den regen Austausch in Bezug auf zahlreiche Forschungsthemen gesorgt haben. Darüber hinaus gilt mein Dank Etienne Barnard und Charl van Heerden für die ausgezeichnete Zusammenarbeit in der Bearbeitung der interkulturellen Forschungsfragen am Meraka Institut in Südafrika, und ohne deren Beiträge dies in der vorliegenden Form nicht möglich gewesen wäre.

Schließlich möchte ich auch meiner Familie danken, sowohl für die Geduld während der intensiven Phasen der Arbeit, als auch für die tatkräftige Unterstützung, wann immer nötig.

Einige Teile dieser Arbeit, die im Rahmen des Projektes CARMINA (Car-oriented Multimodal Interface Architecture) entstanden sind, wurden durch das Bundesministerium für Bildung und Forschung (BMBF) gefördert (Fördernummer 01IW08004).

Short Abstract

Speaker Classification, i.e. the automatic detection of certain characteristics of a person based on his or her voice, has a variety of applications in modern computer technology and artificial intelligence: As a non-intrusive source for user modeling, it can be employed for personalization of human-machine interfaces in numerous domains.

This dissertation presents a principled approach to the design of a novel Speaker Classification system for automatic age and gender recognition which meets these demands. Based on literature studies, methods and concepts dealing with the underlying pattern recognition task are developed. The final system consists of an incremental GMM-SVM supervector architecture with several optimizations. An extensive data-driven experiment series explores the parameter space and serves as evaluation of the component. Further experiments investigate the language-independence of the approach.

As an essential part of this thesis, a framework is developed that implements all tasks associated with the design and evaluation of Speaker Classification in an integrated development environment that is able to generate efficient runtime modules for multiple platforms. Applications from the automotive field and other domains demonstrate the practical benefit of the technology for personalization, e.g. by increasing local danger warning lead time for elderly drivers.

Kurzzusammenfassung

Die Sprecherklassifikation, also die automatische Erkennung bestimmter Merkmale einer Person anhand ihrer Stimme, besitzt eine Vielzahl von Anwendungsmöglichkeiten in der modernen Computertechnik und Künstlichen Intelligenz: Als nicht-intrusive Wissensquelle für die Benutzermodellierung kann sie zur Personalisierung in vielen Bereichen eingesetzt werden.

In dieser Dissertation wird ein fundierter Ansatz zum Entwurf eines neuartigen Sprecherklassifikationssystems zur automatischen Bestimmung von Alter und Geschlecht vorgestellt, welches diese Anforderungen erfüllt. Ausgehend von Literaturstudien werden Konzepte und Methoden zur Behandlung des zugrunde liegenden Mustererkennungsproblems entwickelt, welche zu einer inkrementell arbeitenden GMM-SVM-Supervector-Architektur mit diversen Optimierungen führen. Eine umfassende datengetriebene Experimentierreihe dient der Erforschung des Parameterraumes und zur Evaluierung der Komponente. Weitere Studien untersuchen die Sprachunabhängigkeit des Ansatzes.

Als wesentlicher Bestandteil der Arbeit wird ein Framework entwickelt, das alle im Zusammenhang mit Entwurf und Evaluierung von Sprecherklassifikation anfallenden Aufgaben in einer integrierten Entwicklungsumgebung implementiert, welche effiziente Laufzeitmodule für verschiedene Plattformen erzeugen kann. Anwendungen aus dem Automobilbereich und weiteren Domänen demonstrieren den praktischen Nutzen der Technologie zur Personalisierung, z.B. indem die Vorlaufzeit von lokalen Gefahrenwarnungen für ältere Fahrer erhöht wird.

Abstract

The human voice is a very efficient communication medium that we depend on each day to transfer messages. By taking a closer look, it turns out that the speech we produce is never just a message, but always contains additional, more unconsciously perceived cues about the speaker, the so-called paralinguistic information. The field of Speaker Classification, a sub-field of artificial intelligence, computational linguistics, signal processing, and others, is concerned with the extraction of this information from the speech signal. Examples of such speaker characteristics that manifest themselves in the voice are the person's age, gender, physics, emotion, personality, attention, cognitive load, or even identity.

The main goal of this dissertation is to develop a novel Speaker Classification system for automatic age and gender recognition that allows this information be used in applications for various purposes. One particular motivation is drawn from the aim to utilize the non-intrusively obtained knowledge for personalization of functions and services in the automotive context, which is a very active domain for human-machine interaction research.

One aspect of this thesis deals with the question of how current Speaker Classification systems can be improved. The methodology consequently follows a principled approach, motivating the investigation through analyses of how the speech apparatus is affected by age and gender, instead of looking only at the decision problem. An extensive study and comparison of several recent and early approaches points out where research is heading, what the most promising leads are, and where the culprits are located. Based on this survey, an approach named FRISC is developed and presented in detail. It combines the promising GMM-SVM supervector pattern recognition system with frame-based cepstral voice features (MFCCs) and applies further optimizations to source, feature, and score domain. To be a suitable choice for time-critical on-line application, the system was further designed to process incoming data and update knowledge in incremental manner and with real-time efficiency.

A further focus of this principled approach lies in the way the evaluation of the system is performed. As part of the contribution of this work, a carefully designed parameter space exploration experiment is conducted to discover the impact of each of a large number of different parameters to the overall classification performance. Parameters include MFCC extraction settings, GMM training parameters, SVM kernel function, background model impact, feature and silence filtering ratio, and more. Best practices with respect to data set generation and balancing are particularly emphasized. In spite of the complexity introduced, the procedure ensures that the complete system is always tested using a large number of samples, and only a single parameter is changed between any two experiments. It is believed that the very specific results presented in this thesis are more valuable in the long run than an unstructured or less

transparent evaluation. Nevertheless, the absolute results (51% accuracy for seven classes in the best case) also represent an improvement over earlier systems.

Given the system and the optimal parameter configuration determined experimentally, a subsequent study investigates whether the approach is independent from language and culture of the speaker. Since the voices of people in different parts of the world also have unique regional attributes, age and gender cues could be overlaid by these effects. Based on a South African speech corpus, a cross-cultural study examines how several long-term features change for speakers of different language families. Using these results, as well as a cross-language regression test, confirmation is obtained that FRISC itself is language-independent, but the actual age and gender models are not.

As an essential part of this thesis, a framework is developed that implements all tasks associated with the design and evaluation of Speaker Classification in general in an integrated development environment. The initial motivation for the development of a comprehensive framework was that none of the existing tools could provide the flexibility, performance, and ease of use to make development of efficient runtime modules for Speaker Classification feasible. Using the platform named SPEACLAP, speech technology experts and application designers alike can explore speech corpora and construct classification pipelines for arbitrary speaker properties. The concise application allows complex experiment series to be run and archived with little effort, and surpasses the memory and performance limitations of some tools using specialized data structures and storage providers. Possibly the most distinguishing feature is its ability to automatically build so-called embedded modules, which are portable runtime classification components based on classifiers trained in the development environment, and that are optimized towards resource-efficiency and can easily be integrated into other applications. The various details of this concept and the build process are explained thoroughly in the corresponding chapter.

A final aspect of this work is to take a closer look at the way user knowledge obtained from speech can assist in personalizing the human-machine interface. This includes a more general view on personalization and its challenges, such as the creation and maintenance of user models, concepts of knowledge modeling, customization, and adaptation. Focusing on the automotive domain for most part, several practical fields are identified where Speaker Classification can increase safety or comfort. For instance, elderly drivers might prefer visual presentations to be larger and warnings to be issued earlier. Women might want navigation systems to avoid specific types of parking areas at night. One demonstrator implemented as part of this work does exactly that: it adapts the in-car systems to the driver. A different application generalizes the classification approach from individual user properties to a “voiceprint”, and illustrates how speaker positions, e.g. seats in the vehicle, can be associated with identities. Two further domains studied with respect to personalization are telephone-based services and mobile devices. In case of the latter, a personalized shopping application is presented which recommends different products and features for different speakers.

Zusammenfassung

Die menschliche Stimme ist ein äußerst effizientes Kommunikationsinstrument, auf welches wir täglich zur Übermittlung von Nachrichten angewiesen sind. Bei genauerer Betrachtung stellt man fest, dass die Sprache nie auf die Nachricht beschränkt ist, sondern immer auch weitere, eher unterschwellig wahrgenommene Hinweise auf den Sprecher enthält, die sogenannte paralinguistische Information. Das Gebiet der Sprecherklassifikation, ein Teilgebiet der künstlichen Intelligenz, Computerlinguistik, Signalverarbeitung, sowie weiterer Bereiche, befasst sich mit der Extraktion dieser Information aus dem Sprachsignal. Beispiele für solche Sprechermerkmale, die sich in der Stimme manifestieren, sind das Alter, Geschlecht, Physis, Emotion, Persönlichkeit, Aufmerksamkeit, kognitive Belastung, und sogar die Identität.

Das Hauptziel dieser Dissertation besteht in der Entwicklung eines neuartigen Sprecherklassifikationssystems zur automatischen Erkennung von Alter und Geschlecht, welches diese Information für zahlreiche Anwendungen nutzbar macht. Eine besondere Motivation leitet sich aus dem Wunsch ab, dieses nicht-intrusiv erhaltene Wissen zur Personalisierung von Funktionen und Diensten im Automobilkontext zu verwenden, welcher einen besonders aktiven Forschungsbereich für Mensch-Maschine-Interaktion darstellt.

Ein Aspekt dieser Arbeit behandelt die Frage, wie sich aktuelle Sprecherklassifikationssysteme verbessern lassen. Die Vorgehensweise folgt durchgehend einem fundierten Ansatz, der die Nachforschungen durch Analysen der Auswirkung von Alter und Geschlecht auf die Sprachproduktion vorantreibt, anstatt sich auf das Entscheidungsproblem zu beschränken. Eine umfangreiche Studie mit Vergleich mehrerer aktueller und früherer Ansätze zeigt, in welche Richtung sich die Forschung bewegt, was die aussichtsreichsten Verfahren sind, und wo die Schwächen liegen. Ausgehend von dieser Übersicht wird ein Ansatz namens FRISC entwickelt und im Detail vorgestellt. Er verbindet das vielversprechende GMM-SVM-Supervector Mustererkennungsverfahren mit frame-basierten cepstralen Stimmmerkmalen (MFCCs) und optimiert zusätzlich Aspekte der Quelldaten, Merkmale und Scores. Um als praktikable Lösung für zeitkritische Anwendungen in Betracht zu kommen, wurde das System in Hinblick auf die Verarbeitung eingehender Daten und Wissensbildung in inkrementeller Weise und Echtzeit-Effizienz entworfen.

Ein weiterer Schwerpunkt dieses fundierten Ansatzes liegt in der Methodik, mit welcher die Evaluierung des Systems durchgeführt wurde. Als Teil des Beitrags dieser Arbeit wurde ein sorgfältig ausgearbeitetes Experiment zur Erforschung des Parameterraumes durchgeführt, um den Einfluss verschiedener Parameter auf die Klassifikationsleistung zu erfassen. Parameter beinhalten MFCC-Extraktionseinstellungen, SVM-Kernelfunktion, Einfluss des Background-Modells, Anteil der Merkmals- und Pausenfilterung usw. Bestmögliche Praktiken

werden in Hinblick auf die Erzeugen von Datensätzen ihrer Ausbalancierung in den Vordergrund gestellt. Trotz der involvierten Komplexität stellt das Verfahren sicher, dass stets das gesamte System mit einer großen Anzahl von Datensätzen getestet und nur ein einzelner Parameter zwischen zwei Experimenten verändert wird. Es wird davon ausgegangen, dass die in dieser Arbeit vorgestellten spezifischeren Ergebnisse langfristig von größerem Wert sind als andere weniger strukturierte und transparente Evaluierungen. Dennoch stellt auch das absolute Ergebnis (51% Genauigkeit bei sieben Klassen im besten Fall) eine Verbesserung gegenüber früheren Systemen dar.

Mit Verfügbarkeit des Systems und experimentell erhaltener Kenntnis der optimalen Parameter beschäftigt sich eine Anschlussstudie damit, ob der Ansatz unabhängig von Sprache bzw. Kultur des Sprechers ist. Da sich die Stimme von Menschen in den verschiedenen Teilen der Erde in regional einzigartigen Attributen unterscheidet, könnten Alter und Geschlecht von diesen Effekten überlagert werden. Mit Hilfe eines südafrikanischen Sprachkorpus wird in einer interkulturellen Studie untersucht, wie sich Langzeit-Merkmale zwischen Sprachfamilien unterscheiden. Anhand dieser Ergebnisse, sowie eines sprachübergreifenden Regressionstests, lässt sich die Sprachunabhängigkeit des FRISC-Ansatzes als Ganzes, aber auch die Sprachabhängigkeit der eigentlichen Alters- und Geschlechtsmodelle nachweisen.

Als wesentlicher Bestandteil der Arbeit wird ein Framework implementiert, das alle im Zusammenhang mit Entwurf und Evaluierung von Sprecherklassifikation anfallenden Aufgaben in einer integrierten Entwicklungsumgebung bündelt. Der ursprüngliche Grund für die Entwicklung des umfassenden Frameworks liegt darin, dass kein bestehendes Werkzeug allen Anforderungen an Flexibilität, Leistung und Bedienung gerecht werden konnte, um zum Entwurf effizienter Sprecherklassifikationsmodule herangezogen zu werden. Mithilfe der Plattform namens SPEACLAP können sowohl Experten aus der Sprachforschung, als auch Anwendungsentwickler Korpora studieren und Klassifikationssysteme für beliebige Sprechereigenschaften konstruieren. Die kompakte Anwendung erlaubt die mühelose Durchführung und Archivierung komplexer Experimentalreihen, wobei sie die Speicher- und Performanzbeschränkungen einiger anderer Werkzeuge durch Einsatz spezieller Datenstrukturen und Speicherverfahren umgeht. Das vielleicht herausragendste Merkmal ist die Möglichkeit, automatisch sogenannte eingebettete Module zu erzeugen, welche portable Laufzeitklassifizierer darstellen, die in der Entwicklungsumgebung trainiert und für die einfache Integration in andere Anwendungen in Bezug auf Ressourceneffizienz optimiert werden. Die zahlreichen Details dieses Konzepts und des Kompilierungsvorgangs werden im entsprechenden Kapitel ausführlich beschrieben.

Ein letzter Aspekt dieser Arbeit ist die genauere Betrachtung der Art und Weise, wie das aus Sprache erhaltene Wissen die Personalisierung der Mensch-Maschine-Schnittstelle unterstützen kann. Dies beinhaltet eine allgemeinere Betrachtung von Personalisierung und ihrer Herausforderungen, wie die Erstellung und Verwaltung von Benutzerprofilen, Konzepte der Wissensmodellierung sowie der Adaption. Mit Fokus auf der Automobildomäne über weite Teile werden viele praktische Einsatzmöglichkeiten für Sprecherklassifikation aufgezeigt, aus denen sich Steigerungen von Sicherheit oder Komfort ergeben. So bevorzugen ältere

Fahrer beispielsweise evtl. eine größere Darstellung von Inhalten und frühzeitigere Ausgabe von Warnhinweisen. Frauen wäre es möglicherweise lieber, wenn das Navigationssystem bestimmte Parkplätze nachts nicht vorschlägt. Ein Demonstrator, der als Teil der Arbeit implementiert wurde, folgt diesem Prinzip: er adaptiert die Bordsysteme in Bezug auf den Fahrer. Eine weitere Anwendung verallgemeinert den Klassifikationsansatz von individuellen Sprechereigenschaften hin zu einem “Sprachabdruck”, und veranschaulicht, wie damit die Sprecherposition, d.h. Sitze im Fahrzeug, mit der Identität in Verbindung gebracht werden können. Zwei weitere in Hinblick auf Personalisierung untersuchte Bereiche sind telefonbasierte Dienste und mobile Geräte. Im Fall von letzteren wird ein personalisierter Einkaufs-Assistent vorgestellt, welcher je nach Sprecher unterschiedliche Produktempfehlungen gibt und einzelne Produktmerkmale hervorhebt.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Research Questions	5
1.3	Outline	7
2	Theoretical Foundation of Speaker Classification	9
2.1	How is Individuality Expressed in Voice?	10
2.1.1	Human Speech Production	11
2.1.2	Impact of Individuality on Speech Production	12
2.1.3	Observations on Age and Gender Regarding Acoustic Features	14
2.1.4	Inter-cultural Aspects	16
2.2	Digital Signal Processing	17
2.3	Age and Gender as User Characteristics	20
2.3.1	General Applicability	21
2.3.2	Age and Gender in the Automotive Domain	23
2.4	Automatic Recognition of Speaker Characteristics	24
2.4.1	Embedding in Speech Science	24
2.4.2	Fundamental Aspects	27
2.4.3	Tasks Associated With Speaker Classification	29
2.4.4	Acoustic Features Used to Model Aspects of Voice	31
2.5	Pattern Classification	34
2.5.1	The Pattern Classification Task	35
2.5.2	Sensing	42
2.5.3	Segmentation	42
2.5.4	Feature Extraction	43
2.5.5	Classification	44
2.5.6	Post-Processing	60
2.5.7	Evaluation	61
2.5.8	Classification vs. Regression	74
3	Related Work in the Area of Speaker Classification	75
3.1	Early Work	76
3.2	The AGENDER Approach	77
3.2.1	Data	77

3.2.2	Features	78
3.2.3	Pre-processing and Filtering	79
3.2.4	Two-layered Classification Approach	80
3.2.5	Experimental Design and Evaluation	81
3.2.6	Conclusion	82
3.3	First Speaker Classification Workshop and Subsequent Studies	83
3.4	Interspeech 2010 Paralinguistic Challenge And Beyond	85
3.5	Summary and Comparison of Related Work on Age and Gender	91
4	The FRISC Approach	101
4.1	Concepts and Method	101
4.1.1	Task Description	103
4.1.2	Data Sets	105
4.1.3	Overview of the GMM-SVM-Supervector Concept	106
4.1.4	Frame-Based Features vs. Global Features	109
4.1.5	Silence Filtering	114
4.1.6	Feature Domain Optimization	115
4.1.7	Score-Level Optimization	118
4.2	Experiments Overview	120
4.3	Experimental Investigation of System Parameters	121
4.3.1	Experimental Set-up	122
4.3.2	Results	124
4.3.3	Discussion	125
4.4	Post-processing Level Fusion: A Hybrid System	133
4.4.1	Combining Short and Long Term Features	134
4.4.2	Combining Classification and Regression	135
4.4.3	Conclusion	137
4.5	Influences of Language and Culture	137
4.5.1	The Lwazi ASR Corpus	139
4.5.2	Feature Analysis	141
4.5.3	Regression Analysis	144
4.5.4	Results	146
4.5.5	Conclusion	147
4.6	Comparison with Synthesized Speech: Analysis-by-Synthesis	149
5	An Efficient Framework of Applied Speaker Classification	153
5.1	Scenarios and Requirements	155
5.2	Proposed Framework Architecture	160
5.3	The Integrated Development Environment (IDE)	161
5.3.1	General Features	163
5.3.2	Corpus Management	172

5.3.3	Feature Extraction and Management	174
5.3.4	Classifier Design	180
5.3.5	Classifier Evaluation	184
5.4	Embedded Modules	189
5.4.1	Basic Structure	189
5.4.2	Module Design	192
5.4.3	Compiling Modules	194
5.4.4	Module Implementation	196
5.4.5	Summary	197
5.5	Example: FRISC Experiment Set-up	198
5.5.1	Feature Extraction	199
5.5.2	Silence Filtering	201
5.5.3	Data Set Generation	202
5.5.4	GMM-UBM Training and Export	203
5.5.5	Feature Space Reduction	205
5.5.6	SVM Training	206
5.5.7	Score-Level Optimization and Evaluation	206
5.6	Evaluation	207
5.7	Summary	209
6	Achieving Personalization using Speaker Classification: Applications in the Auto-	211
	 motive Context and other Domains	
6.1	Theoretical Foundation of User Adaptation and Personalization	213
6.1.1	User-Centered Design	214
6.1.2	Personalization and Customization	216
6.1.3	User Modeling	219
6.1.4	Facets of User Identity for Applications	221
6.1.5	User Adaptation	224
6.2	Speaker Classification in the Automotive Domain	225
6.2.1	Overview and Categorization of In-car Functions	227
6.2.2	Characteristics of Personalization in the Automotive Domain	231
6.2.3	Speech in the Vehicle Context	235
6.2.4	Major Related Projects	238
6.2.5	Adaptation of an In-vehicle Information System	244
6.2.6	Multi-Speaker Positioning: Who Sits Where?	249
6.2.7	Fusion of Knowledge Sources	259
6.2.8	Integration in a Vehicular Personalization Component	261
6.3	Speaker Classification for Mobile Devices	267
6.3.1	Characteristics of the Mobile Devices Domain	268
6.3.2	A User-adaptive Mobile Shopping Assistant	270

6.4	Speaker Classification for Phone-based Services	275
6.4.1	Characteristics of the Phone-based Services Domain	275
6.4.2	Recording Caller Information	276
6.4.3	Automatic Call Distribution	276
6.4.4	Dialog Adaptation	277
7	Conclusion	279
7.1	Research Questions Revisited	280
7.2	Outlook	283
	References	285

List of Abbreviations

FRISC	FFrame-based Incremental approach to Speaker Classification
KAPcom	Knowledge management, Adaptation and Personalization COMponent
SPEACLAP	SPEAker CLAssification Platform
ACC	adaptive cruise control
ACD	automatic call distribution
ADAS	advanced driver assistance system
AI	artificial intelligence
ANN	artificial neural networks
APE	applied probability of error
API	application programming interface
APQ	amplitude perturbation quotient
ASR	automatic speech recognition
AU	application unit
CAN	controller area network
CART	classification and regression trees
CAS	collision avoidance system
CCU	communication and control unit
CDHMM	continuous densities hidden Markov model
CFS	correlation-based feature subset selection
CMVN	cepstral mean and variance normalization
CPT	conditional probability table
CSV	comma-separated values
CWS	collision warning system
DAS	driver assistance system
DBN	dynamic Bayesian network
DCF	detection cost function
DET	detection error tradeoff
DFT	discrete Fourier transform
DSP	digital signal processing
EER	equal error rate
EM	expectation-maximization
FA	false alarms
FFT	fast Fourier transform
GMM	Gaussian mixture model

GPGPU	general-purpose GPU
GPU	graphics processing unit
GUI	graphical user interface
HCI	human-computer interaction
HMI	human-machine interface
HMM	hidden Markov model
HNR	harmonics-to-noise ratio
IDE	integrated development environment
IES	infotainment/entertainment system
IT	information technology
IVIS	in-vehicle information system
JFA	joint factor analysis
kNN	k-nearest neighbor
LCT	lane-change task
LDA	linear discriminant analysis
LDW	lane departure warning
LLR	log-likelihood ratio
LP	linear prediction
LPCM	linear pulse code modulation
LTAS	long-term average spectrum
MAE	mean absolute error
MAP	maximum a posteriori
MCLR	multi-class logistic regression
MDI	multiple document interface
MFCC	Mel-frequency cepstral coefficients
ML	machine learning
MLP	multilayer perceptron network
MMI	maximum mutual information
NAP	nuisance attribute projection
NVC	nuisance variability compensation
PCA	principal component analysis
PDA	personal digital assistant
PLP	perceptual linear prediction
PSIAIF	pitch synchronous iterative adaptive inverse filtering
QP	quadratic programming
RAP	relative average perturbation
RASTA	relative spectral
RBF	radial basis function
RDF	resource definition framework
RMNL	random multinomial logit
ROC	receiver operating characteristics

SNR	signal-to-noise ratio
SPL	sound pressure level
SVM	support vector machine
SVR	support vector regression
TPP	tandem posterior probability
TRAP	temporal pattern
TTS	text-to-speech
UAR	unweighted average recall
UBM	universal background model
UIML	user interface modeling language
UML	unified modeling language
VAD	voice activity detection
VTLN	vocal tract length normalization

1 Introduction



Figure 1.1: The number of functions and services in the vehicle, including those that go beyond driving, has recently been multiplying.

1.1 Motivation

Speech is probably the most natural and well-mastered means of communication between humans. We use speech every day to deliver messages, to communicate in business and private life, to give or receive information, to make compliments or express annoyance, with a clear goal or simply to chat. But what seems like a relatively easy performance for humans is on closer examination a very complex process. Not only does the number of languages, words, and grammatical rules testify this complexity, but we can also quickly convince ourselves by studying the intricate path a sentence takes from our brain through the speech production apparatus to a vocal utterance.

Since long, computer science has tried to utilize the potential of speech for human-computer interaction. In automatic speech recognition, we have tried to convert the vocal sound into written words with the aid of a computer. With the help of linguistics and statistics, this works fairly well today. By connecting ASR with the discipline of natural language understanding, we can even interpret the spoken words and have computers react to them in an intelligent, human-like manner. This is for example exactly what happens when users talk to the *Siri* voice system on their *iPhones*. With slightly different motivations, researchers have also been able to convert a written sentence into sound that closely resembles the human voice. For some sophisticated systems, it is difficult to tell the difference between machine and human.

In recent years, computing has finally entered the remaining and most complex areas of everyday life, thus becoming truly “ubiquitous”: The home being the location where the digitization of equipment generally started in the eighties, is now followed by the places we visit on foot. Modern mobile phones, becoming more and more *smart* phones, are the key

factor in this episode. They combine the user’s data, a connection to the rest of the world, color displays, and substantial processing power in a small device. And finally, there is the place where Americans spend on average 6% (approximately 1.5 hours per day) of their time (Klepeis et al., 2001)¹, and where phones are not so convenient, which is in their car. Through computer technology, the vehicle is currently making the transition to a new level of driver support, communication, and comfort (see Figure 1.1).

In many of these areas, speech is more than just another input modality. It has a property that most other modalities cannot offer: It works “hands-free” and “eyes-free”, which means that the users can keep their eyes on the environment, which is of vital importance in the car, but also important when walking around with a mobile device. Since adding more functions – some helpful, others merely entertaining – also requires more attention from the users, it can also mean distraction from other tasks, possibly causing them to run into other people, or, in case of the vehicle, provoking accidents. Using modalities that allow a safer manipulation of the technology is a clear advantage. This increasing importance of the speech channel in future HMI applications is the main motivation of this work for having a closer look at its potential and characteristics.

At the same time, we see a trend towards personalization of end-user devices and services. The technology we interact with is no longer static, but will react completely different depending on who is using it. Today, users impose much more intelligence on their systems – they expect software to figure out their intentions without explicitly specifying them and take care of possible complications. Without an appropriate model of the user and background knowledge of her skills, abilities, and preferences, this is not possible without bothering the user. Personal computers and mobile devices have already been affected by this development in recent years. As vehicles are accumulating more advanced technological components, it is clear that they are heading in the same direction.

As of today, even though the importance of speech is increasing, its potential for personalization is greatly underused. As we all know but often do not realize, speech carries a lot more information than just a series of words with a semantic meaning. It also reveals plenty about the speaker, the so-called paralinguistic information. We can confirm this by thinking about how we imagine the person behind an unfamiliar voice on the phone, where no other modality can be consulted. We probably have an idea of the person’s gender and approximate age. We might even guess some aspects of the person’s physics, which the voice also contains. With the corresponding knowledge, we might be able to determine the country in which the person grew up. Also, we can tell if the person is happy, angry, bored, tired, or drunk – all this without actually understanding the words. And much of the way we speak with another person depends on exactly this information, i.e. we *adapt* to them.

The goal of this thesis is to teach the computer to perform the same task when it works with human speech: extract paralinguistic information so that it can serve as a basis for adaptation, or more generally personalization (which is not the only possible use). In the

¹Other studies further show that the amount is increasing, see e.g. *NHTS BRIEF – National Household Travel Survey* (2006).

“shopping scenario of tomorrow” (Wahlster et al., 2010) for instance, customers communicate naturally with anthropomorphous objects in a shop. Here, a cell phone should probably introduce itself quite differently to men and women, talking design for women and pointing out the software specifications for the male customers. Likewise, we expect that wine bottles won’t advertise themselves to children. With the amount of technology loaded into phones as well as cars, the older generations are often neglected, and may have trouble finding the basic functions. A system that pro-actively aids by providing further guidance or adapts itself simplifying the interface could be a decisive factor for many people. Particularly in the vehicle, where virtually everything can impact the safety of drivers and passengers, choosing the most appropriate settings for font sizes, speech output volume and talking speed, display times of warnings, selection of information, etc. does indeed matter. We will see several examples of personalization based on speech similar to the aforementioned situations being realized in this work.

The field of Speaker Classification deals with the technical method of extracting this paralinguistic information from speech. It has been active since shortly after 2000, although it has not received as much attention as other major speech technologies. This is partly due to the absence of influential applications prior to this time, and vice versa also due to the early stages of the technology and lack of robust implementations, making it an uneasy choice for actual products. It has recently received a push with Interspeech 2010 Paralinguistic Challenge (Schuller et al., 2010). Yet, there are overall still very little systematic studies of the acoustic features and concepts. This work attempts to reach a further milestone in Speaker Classification research by presenting a conceptual foundation and architecture that is evaluated in a principled evaluation according to a rigorous design and a focus on its suitability for upstream application. It further does what no other authors to our knowledge have attempted before, namely to study the automatic recognition of age and gender in a cross-lingual and cross-cultural context.

Most other work on Speaker Classification ends with a report on recognition performance. The step to personalization in a real-world scenario is still enormous, because results achieved with a research implementation in the lab cannot simply be transferred due to the requirements that actual application impose, such as real-time performance, scalability, and resource-adaptivity. The scenarios previously mentioned present demands at the technology that need to be considered during the whole design process. Even then, for the technology to actually find its way into homes, devices, and cars, a certain level of tool support is essential. A truly comprehensive solution would provide a convenient framework for researchers to set up and conduct experiments, while at the same time enabling application designers to configure Speaker Classification modules that suit their needs and are ready to be integrated into their existing software. Such a framework is an integral part of the author’s vision.

1.2 Research Questions

The previously presented motivation leads to the following scientific research questions that guided the work on this thesis:

1. **What are the components of a Speaker Classification approach for the estimation of speaker age, gender, and other characteristics, that is designed to deliver high classification performance, yet be sufficiently flexible to be applied to a variety of classification problems in multiple application domains?**

Speaker age and gender classification systems have not been extensively investigated until approximately half a decade ago. By applying new signal processing or pattern recognition methods or optimizing the feature set, there is still considerable room for improvement. This thesis attempts to construct a classification pipeline from a novel selection of components, thereby pushing forward research in this area. As opposed to other, often heavily engineering-focused performance tuning approaches, the bottom-up approach followed in this thesis builds on existing knowledge about the inner workings of voice aging whenever possible. Therefore, while still an important benchmark, classification accuracy is not the only success criterion. In order to be functional in the domains brought forward in the application part of the thesis, the objectives are complemented by the prerequisite of flexibility, i.e. the ability to build implementations satisfying domain-specific runtime requirements and to be extensible to other speaker properties such as emotion, height, cognitive load, sleepiness, etc.

2. **Which are the parameters that affect the performance of such a Speaker Classification system, and how can a principled approach be created to empirically determine the relative impact of each of them?**

A single good performance number, which could be the result of heavy engineering and meticulous fine-tuning, is certainly respectable, but by itself not overly beneficial to the scientific progress. Even the final architecture of a Speaker Classification system is not as insightful as the stepwise illustration of how it was obtained and configured that way. In this thesis, we want to present a principled approach, a transparent strategy by which the main parameters that affect the classification performance are investigated and iteratively improved. By closely following the relative improvements on a constant data set, we hope to gain better insights that further work can benefit from.

3. **Which aspects of the approach can ensure the incremental processing of information, i.e. a fast initial estimation and a continuously increasing expected performance with the arrival of additional speech data?**

Several Speaker Classification algorithms require complete turns before they return an estimate. This makes them unsuitable for many on-line scenarios, which depend on a steady incremental update of the current system belief right from the first data. Such applications may not even have the concept of a defined instance to be classified, but

consider a continuous voice stream as input. An example is the automotive domain, where many default settings (e.g. seat configuration, font size on screen) should be personalized as soon as possible after the user enters the car. We will point out several aspects of the design which – in conjunction with the development framework – have been optimized for this purpose.

4. Is speaker age and gender estimation or the Speaker Classification approach as a whole influenced by the speaker’s language or culture?

It is known that language and culture (or ethnical background) have an impact on the human speech production. Not only does it affect the vocabulary, which obviously differs between languages, but also the prosody or basic acoustic features. In a global society, it should be a prime concern to confirm whether an approach like the one developed would work for people living in other regions as well. Languages and language families are the most obvious boundaries for such regional differences, but there may be additional influences, e.g. ethnic or socially motivated, which are often collectively called cultural influences, and which we should consider as well. Regarding Speaker Classification, there are three possible cases: (1) The features used in our approach are not affected at all, i.e. there are no language dependencies whatsoever. (2) The feature space is affected, but in a consistent way requires only an adjustment of the model. (3) The effects override those from characteristics such as age and gender, making it necessary to find different features for the latter and possibly change the approach. Our hypothesis is that the employed Speaker Classification approach is generally language and culture independent; however, there is sufficient evidence that different training data will be required for different groups of speakers. A number of analyses are performed to confirm this hypothesis.

5. How can the approach developed for age and gender be used to recognize the identity of a user through his or her voiceprint?

Speech-based biometry is closely related to personalization, and is particularly helpful in the automotive domain to solve the problem of passenger positioning, as will be demonstrated in Section 6.2.6. Since the concepts used for age and gender estimation have their origins in speaker recognition, it is a natural question to investigate whether the same architecture can be used to detect speaker identity, i.e. to move from specific characteristics like age and gender to the more generic concept of a voiceprint. In this regard, we want to examine the feasibility in general and determine what performance can be reached with a basic system, rather than introduce new concepts to improve the performance.

6. How can an architecture or framework be designed that allows speech-based classification technology to be created, evaluated, and deployed according to the diverse requirements in desktop, server, and embedded scenarios?

The contribution of this work goes beyond the development of a specific speaker age

and gender recognition approach. To ensure its versatile use, a more general Speaker Classification framework is needed that allows the concepts to be applied to different speech-based classification problems, characteristics, data, features etc. This framework should be able to adjust a given design to the requirements of different domain and platforms, essentially “building” a resource-aware implementation based on the design specification. Additionally, the framework provides a rapid development process and a consistent approach to system evaluation and experiment design. As part of this research question, the framework requirements need to be formalized, and a number of particular issues that makes Speaker Classification training challenging (e.g. the high volume of data) have to be solved through engineering. The resulting framework should be made available to the public domain to foster wide adoption.

7. How can non-intrusively acquired speech-based user characteristics such as age and gender help to personalize the user experience in user-centered domains with different characteristics and demands?

Personalization based on non-intrusively obtained information is probably the most important application of Speaker Classification technology. Adapting the design or function of a system to the user can – among other things – increase comfort, usability, and efficiency. Resorting to the user’s voice allows personalization to be performed in situations when no other data is available. Based on the previous achievements, age and gender are the most prominently examined properties, providing many clues with respect to preferences or requirements of the user. Three particular domains are investigated: automotive HMI, phone-based services, and mobile devices. In each of these domains, concrete examples for the benefit of the technology will be given. The automotive domain is studied in more depth through the introduction of domain-specific concepts, sample applications, and a middleware for the exchange of user profile data.

1.3 Outline

The chapters of this dissertation are organized as follows.

Chapter 2 contains the theoretical foundation and backgrounds of all concepts related to Speaker Classification as a research field. These backgrounds consist of the phonetic aspects of human speech production, from which we will also draw the motivation for considering age and gender as speaker properties, digital (speech) signal processing basics, a description of the Speaker Classification task and the acoustic features, and finally an introduction to the pattern recognition methods applied.

Chapter 3 gives a comparative survey of other Speaker Classification work that deals particularly with age and gender recognition. A focus will be on the AGENDER system (Müller, 2005), which accounts for much of the theoretic and practical motivation of this thesis, and represents a notable milestone in Speaker Classification research.

The novel approach developed in this thesis is introduced in Chapter 4, starting with

the general architecture and then detailing the individual concepts. Subsequently, multiple experiments are described in the chapter, each dealing with separate aspects. The majority of experiments belongs to the main experiment series that explores numerous parameters in a single contiguous set-up. A second experiment shows how meta-level fusion impacts the overall result. Further, cross-cultural influences are analyzed in a study that is based on a multi-lingual speech corpus. Finally, a study is presented where the system is challenged by synthesized voices that are rated in parallel by the computer and human listeners.

These former three chapters deal mainly with the theoretic and methodologic aspects of age and gender recognition. The following Chapter 5 moves on to the details of the implementation, presenting a complete Speaker Classification framework (SPEACLAP) that is generically applicable and not limited to those two speaker properties. The chapter closes with an evaluation of the runtime performance of the classifiers.

Chapter 6 finally gives a survey of several applications of the technology. These applications are stemming from three different domains (automotive, mobile devices, phone-based services), but share a common goal, namely personalization based on the knowledge obtained from speech. As part of this chapter, multiple implementations of working systems created for this thesis are presented.

Chapter 7 concludes with a summary and an outlook.

2 Theoretical Foundation of Speaker Classification

2.1 How is Individuality Expressed in Voice?

There is no doubt that when we speak, we want to transfer a message: traditionally, a message to one or more other humans, or, since more recently, also a command targeted at a computer. In any case, it is the content and in effect the intention which is perceived as the essence of spoken words. It is therefore easy to overlook the other attributes that are transmitted as part of the speech, but not as part of the content. What actually happens when speaking is that the message is created as a sequence of sounds on the speaker's side, transmitted as acoustic waves over the medium air, and received, decoded and interpreted by a listener. The content is only a small part of what is encoded in the voice. The sound that makes up the speech contains much more information, as it will have a different shape for every speaker. This information is also called paralinguistic information, since it is transferred in parallel to the content. It is affected by many characteristics, such as physical properties (e.g. height or size of the head), gender, age, illnesses, drugs, language backgrounds (e.g. dialect or experience), culture, and others. The possible range of effects is as large as the individual differences in people. They can occur on multiple "levels" of the processing apparatus: It can be a general transformation affecting the overall "color" of the sound (e.g. a "higher" or more "breathy" voice), it can be an effect on the rhythm or intonation of specific sounds and syllables (e.g. an accent or stuttering from nervousness), or it can be a word-level effect (e.g. speaking under the influence of alcohol or using outdated vocabulary). Closely related to the possible impact is the place where they emerge: For instance, some result directly from physiological circumstances in the speech production system. These are usually rather constant over time, although they can still change considerably over a whole life, in the same way other biological aspects change as part of the aging process. Some others are related to the brain and nervous system, and are subject to more frequent change, even between two sentences as a reaction to stress.

Even though these characteristics are there, we often do not perceive them consciously. For example, we associate and remember these voice "features" for people we have spoken to, and use them – possibly in conjunction with visual and other impressions – to recognize a person, without being able to actually name these unique voice features. For human listeners, their ability to do this varies widely, and the details of this process, e.g. which features play what role under which circumstances, are barely understood even today (Hill, 2007). As studies indicate, even voices with strong phonetic differences can be difficult to associate with people by humans, and vice versa (see *ibid.*). Nevertheless, the same basic idea has been followed for a long time in speaker verification and speaker identification: By looking at particular, steady features of the voice, a match is attempted with a reference model, the so-called *voiceprint*. Obviously, for the deterministic machine-based solution, it can always be clearly explained which features are considered and why they lead to a specific decision result.

For more concrete properties of the speaker like the examples given earlier, the basic process is similar. When we hear a voice for the first time, we more or less unconsciously perceive the features that indicate age and gender, and compare them to the reference models we have

built up over time. This will happen in a more conscious manner when we have little other sensory information, e.g. when the voice is talking on the phone, or when there is a mismatch between our expectation and the actual voice heard. Even then, it is not easy to explain why exactly we judge a voice as “happy” in one occasion and as “sad” in another.

When we talk about reference models for speaker properties, the question arises whether they are globally valid, or whether they are only valid among a certain group of people, such as a group speaking the same language or sharing cultural aspects. It is known that intonation, rhythm, and even more obviously words are affected by culture and language. Hence, it may likewise be possible that any speaker characteristic for which these features serve as source of information could also be affected.

In this section, we look more closely at the stages of speech production to understand how individuality manifests itself in these stages. The walk-through however will not be a detailed primer on speech production, for which further literature from phonetics should be consulted.

2.1.1 Human Speech Production

For a better understanding of how individuality influences articulation, a general knowledge of the basic speech production process is needed. The human vocal organs are shown in Figure 2.1. Ladefoged (2006) divides the human speech production system into four main parts: the airstream process, the phonation process, the oro-nasal process, and the articulatory process. The *airstream process* describes how air is pushed out of the lungs and forms the “carrier” for producing sounds. The pressure built up by the lungs (or rather the indirectly connected intercostal muscles and diaphragm) provides the required energy. The vocal organs of the *phonation process* are centered around the larynx (see Figure 2.1 on the right side). When the air from the lungs is pushed through the larynx, the positioning of the vocal folds determine whether a voiced or a voiceless sound is created. A voiced sound is generated when the vocal folds are close together (adducted), blocking the air flow through the opening that is also called glottis and being set into vibration. This vibration effectively creates pulses of air that can be observed as sound waves. Apart from the voiced and voiceless states, a number of other states can be arranged by the larynx by changing the configuration of the vocal folds, their tension and degree of adduction, adjusting parameters such as pitch, loudness, breathiness, or creakiness (Dellwo, Huckvale, & Ashby, 2007). Switching between mouth and nose for the escape of air is part of the *oro-nasal process*, in which the soft palate is arranged accordingly to direct the air flow. When it is pressed against the pharynx wall, air cannot exit through the nose. If it is open and the mouth is closed, it can be used to produce nasals. The *articulatory process* is controlled mostly by parts which are found in the vocal tract. They can be thought of as giving a certain “shape” to the sound. The most prominent parts are the lip, teeth and tongue, which, through their positioning and movement, have a critical impact on what type of consonant or vowel is produced. In particular the tongue is highly mobile and allows extremely precise positioning (Ladefoged, 2006). The other parts are the alveolar ridge (behind the upper teeth), the hard palate, the soft palate, the uvula,

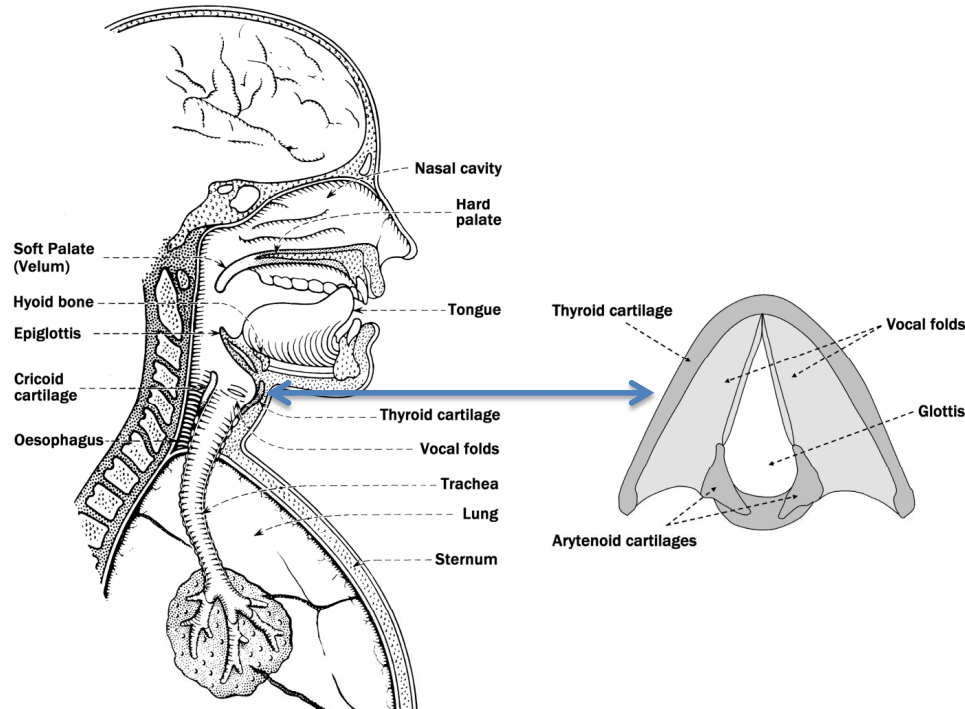


Figure 2.1: Overview of the organs of the speech production system. (Source: Dellwo et al., 2007, p. 3)

the tongue back, and the tongue root, with all ending in the pharynx.

At this point, only the articulation step, i.e. the process to form individual sounds, which can be used as the phonemes or phoneme combinations in words, has been described. To actually produce meaningful words or even sentences, several more steps are required, which run earlier and at a higher cognitive level than the basic processes described in this section. Due to their complexity, a description of these linguistic processes is not provided in this work, but is subject of other literature where it can be found in detail (see e.g. Levelt, 1991, 1999; Pechmann & Habel, 2004).

2.1.2 Impact of Individuality on Speech Production

Speech production consists of multiple stages (e.g. sound production, word forming, sentence forming), and the areas of influence of individuality described by Dellwo et al. (2007) roughly reflect these stages: the anatomical influences, control of the speech production, the sound system, and language use. Apart from the affected area of speech production, we can also take into account the origin of the effect, which can be divided into four categories: It could be caused purely by the individual's anatomy and completely out of the speaker's control (e.g. a generally high voice), it could have been acquired over time through training and practice (e.g. dialect), it may be a consciously exerted effect chosen by the speaker in a specific

Cause	Production Area	Type of Cause	Variables	Observed Effect
e.g. adjustment of the vocal folds, laryngeal changes...	Sound Production	Anatomical	Gender	Change of Pitch
	Word Forming	Experience	Culture	Breathiness
	Sentence Forming	Context (conscious)	Stress	Vocabulary
		State (unconscious)

Table 2.1: Scheme for describing and classifying effects of individuality in human speech.

context (e.g. selection of a language), or it can be an automatic context and state dependent effect (e.g. stuttering due to nervousness). Then, most effects are not completely random, but interact with other characteristics of the speaker, such as the age, gender, background, so we can also consider such influential variables. And finally, of course, we can attempt to describe the appearance of the effect, i.e. how it affects the vocal sound (such as changing its pitch), which can be done in more or less formal manner, and which is also an important step when we want to take the reverse path of concluding from an observed effect to its cause, as we do in Speaker Classification. (This description of the perceived effect is in turn subject to its own scheme, reaching from subjective perception to formal measures and features that can be determined automatically. These aspects are described in the next section.) This scheme is summarized in Table 2.1. In explaining the individual’s parameters that affect the speech production, we are closely following the article by Dellwo et al. (2007), from which this section has also inherited part of its title.

Language-related choices are one of the areas where individuality can have an impact. According to Dellwo et al. (2007), the effects reach from the choice of language, the mother tongue, and the number of secondary languages, to the choice and pronunciation of words. This includes dialect, which are linguistic variations specific to a group, such as a geographic region or social grouping. The latter are two examples of variables influencing (supporting or overriding) individuality. Another variable is the decade of birth, since the language and vocabulary changes over time. A further example is – to some extent – gender, i.e. a difference in the words chosen by women and men. Choice of words and language may also depend on the situation, most importantly the conversation partner.

The *sound system* can be divided into the segmental component and the supra-segmental component. The former deals with the mapping from words to phonetic segments and is part of the speaker’s accent, which could again depend on the speaker’s environment or which other languages are spoken. Supra-segmental parameters are “stress pattern, timing and intonation of the sequence” (Dellwo et al., 2007), which are mostly impacted by the intention of the speaker. They can also be affected by emotion or other parameters of the speaker’s state. A phenomenon witnessed among young English speakers in certain regions is a rising intonation called “uptalk” (see ibd.).

The *speech production process control* refers to the exact positioning and interplay of articulators such as tongue and teeth, but also vocal folds. There are many ways of producing vowels, some creating rather different sounds for the same phoneme, and others changing

aspects of quality. Examples are whispering, shouting, or otherwise speaking with a higher pitch than usual. Slow and fast speaking rates will have effects in this area, too, with fast speaking frequently leading to coarticulation (overlapping of phonetic segments). Consumption of drugs and alcohol are typical conditions that can temporarily affect the articulation process as well.

Intermediately responsible for observing effects of age, gender and other more “static” speaker properties are mostly the *physiological* or *anatomical* influences. The differences in speech organs for these classes result in measurable variations of the speech signal which always overlay the short-term variations that depend on the situation and state. Some of the parameters pointed out by Dellwo et al. (2007) are the length of the vocal tract, length and tension of the vocal folds, capability of the respiratory system, and form of the soft palate. Vocal tract and vocal folds have a particular impact on the default pitch of the voice and the frequencies the speaker is able to produce, and they differ clearly on average between women and men. For instance, the vocal tract length of women is about 87% of the length of that of a man. Likewise, this factor is 80% for the vocal folds (Wu & Childers, 1991). Many of the anatomical changes related to vocal aging beyond the age of 12 years have been consolidated by Müller (2005). This includes the growing and stiffening of the larynx, the gain and loss of brain and nervous system mass, the changes in flexibility of the lungs and chest, and the general involution of muscles with age. Finally, pathological conditions, such as from a cold or chronic disease, including the consequences of smoking, can also impact the physiological characteristics of the vocal organs and thus the voice. A well-known effect is the disturbance of the oro-nasal process, resulting in the typical “nasal” voice.

2.1.3 Observations on Age and Gender Regarding Acoustic Features

From knowing how individuality manifests itself in speech production to presenting methods that can measure these effects, there is still a way to go, because not all physiological variables can easily be measured. This is indeed one of the particular challenges of the age recognition task: finding appropriate measurable effects data, also known as features, that can be acquired in a non-invasive way. For example, prior to the discovery of signal-based methods, the airflow waveform at the glottis, or volume velocity, and with it several glottal features, could only be measured by pneumotachograph at the mouth or specialized microphones (Rothenberg, 1973). Nowadays, methods such as pitch synchronous iterative adaptive inverse filtering (PSIAIF) allow us to extract such features directly from the signal (Mendoza, Cataldo, Vellasco, & Silva, 2010; Pulakka, 2005). Yet, for many effects, we still rely on approximations and empirical studies to confirm relationships between features and effects, or between feature and speaker property. This is even more complicated by the facts that the effect is typically not linear, is subject to large overlapping between age classes, and usually contains traits of many other speaker properties as well.

There is a high quantity of different types of features. Reynolds et al. (2003) categorize features into five layers, from low-level to high-level cues, which are spectral, prosodic, pho-

netic, idiolectal, dialogic, and semantic features. By tendency, the lower-level features can be attributed rather to physical traits, while the upper-level features are learned. Müller and Wittig (2003) distinguish between the three levels of acoustic, prosodic, and linguistic features, from low to high abstraction layer. In some cases, the boundaries between these layers are blurred, yet in general, this work does not look at features above the acoustic level. The reason for this is twofold: For once, these features are part of a closed group of attributes, since they share many aspects and are based on a similar accessory of methods for their extraction and processing. Working with linguistic features takes the task into a completely different area of speech science. The second reason is that the lower-level features are more likely to ensure the general applicability of the approach, as the high-level features are often language-dependent.

For this work, the main foundation for the observation of aging effects in speech features is the study by Müller (2005). According to his findings, pitch is one the most suitable features for gender discrimination, particularly for the *adults* class. Voices are generally higher and their difference becomes smaller for younger people. Seniors expose a lower minimum pitch, while the average increases for men and decreases for women. Teenagers have a higher jitter and shimmer value than adults, which supposedly reflects the changes in the vocal system during puberty. The same effect can be noticed for seniors, where the two features are even higher. The harmonicity-to-noise ratio, which is low for children and teenagers, is on average higher for adults and seniors.

A comprehensive survey on age and gender related changes with respect to a number of acoustic features is provided by Schötz (2007). In addition to the chronological age, dependencies on the perceptual age are also considered in her study (see Section 2.3 for a definition of these age specifications). We learn that almost all features have a generally higher variation for seniors. This also includes otherwise less considered features such as vocal tremor (the standard deviation of pitch). Observations on the different changes of average F0 with aging for women and men is similar to those reported by Müller (2005). Although not quite an acoustic feature, the speech rate is another good indicator of age as confirmed by several studies, with a decrease of 20-25% with older age (Schötz, 2007). Other features that have shown to correlate with age and/or gender mentioned in the survey are the sound pressure level (SPL), which increases for men over 70 years; spectral features like spectral tilt, spectral emphasis, and spectral balance; and the long-term average spectrum (LTAS), which shows a tendency to rise or fall in certain frequency bands for elderly women and different frequency bands for men. Changes in vocal tract length due to age are often assumed to manifest themselves in resonance features. Therefore the formant frequencies F1 and above could allow conclusions to be drawn, although the author points out that this interrelation may depend on the vowel.

MFCC features (Mel-frequency cepstral coefficients, see also Section 2.4.4) have a strong relationship to the shape and configuration of the vocal tract. This includes aspects relevant to the sound currently being generated, to the individual speaker, and more general speaker properties such as the age and gender. The speaker-specific characteristics contained in the

MFCC can be quite prevalent, which is why techniques such as vocal tract length normalization (VTLN) are applied to remove these traces. On the other hand, MFCC have been quite popular in speaker recognition for some time. The coefficients are a more abstract feature than most others mentioned in this section, due to the numerous transformations applied, therefore the impact on MFCCs cannot usually be grasped by the human eye and ear. Instead, computers are much better at analyzing this complex type of data and drawing conclusions.

2.1.4 Inter-cultural Aspects

This thesis puts a special focus on influence factors of vocal sound that are relevant in conjunction with globalization, an important quality of IT software today. The most obvious are language-related factors, which [Schultz \(2007\)](#) enumerates as being language, accent, dialect, idiolect, and sociolect of a person. The aforementioned factors are usually more or less dependent on each other, e.g. a dialect is typically considered a sub-category of a language. In spite of this, there are also similar phenomena in dialects across languages. One reason for this is that an official “language” is a clearly defined construct, while dialects and even more accents are not. Several Chinese dialects are not promoted to languages merely for political motivations ([Schultz, 2007](#)). Language-related effects can impact speech production on multiple levels, including dialogic (choice of discourse strategy), linguistic (choice of words, grammar, vocabulary), prosodic, and acoustic. The average length of words and sentences is one effect that is different in every language. Choice of words can depend on factors like the social structure, as is the case in Japanese (*ibid.*). Referring to prosody, the consonant-vowel pattern is very characteristic for some languages: Turkish is a common example of a language with a very regular CV-pattern, while many eastern European languages are well-known for the extensive use of consonants. The fact that phoneme n-grams are commonly used for language identification ([Müller & Feld, 2006](#)) supports this observation. There are differences between vowels, too, such as the nasalized version of some vowels used in French ([Dellwo et al., 2007](#)). The airstream process normally involves the so-called egressive pulmonic airstream, which produces sounds for all languages. Special constructs used in only some languages are generated using non-pulmonic airstreams. These constructs are ejectives, implosives and clicks (see *ibid.*). On the acoustic level, tonal languages such as Mandarin Chinese show a very characteristic intonation pattern, with a specific pitch movement for each syllable. Other languages have their custom intonation patterns as well, but it is generally applied rather at sentence level to convey a particular function or emphasis (see *ibid.* and [Sorin et al., 2004](#)). Concerning statistics, [Schötz \(2007\)](#) reports on “language-related, dialectal and attitudinal differences in habitual F0, HNR and shimmer levels“. With respect to pitch, tonal languages seem to show considerably higher standard deviation in pitch, so-called F0-excursions ([Traunmüller & Eriksson, 1994](#)).

Apart from these language-related factors, which we can describe as being defined by the individual’s environment, i.e. what he or she learns to speak, there are also more general or

even orthogonal aspects. For example, women in certain parts of Asia (China, Japan...) do have a basically higher pitch than women in the western world. Hao (2002) found significant differences in vocal tract length, pharyngeal volume, other physiological measures, as well as formant frequencies between white American, African American, and Chinese speakers. Traunmüller and Eriksson (1994) confirm this by noting that the range of spoken pitch in a language is usually chosen according to physical considerations, i.e. to produce as little effort as possible. On the other hand, the social environment can have a further effect on intentional modification of the voice spectrum. Bezooijen (1995) attributes the raised voice of Japanese women to such factors. Deutsch et al. (2009) found that for speakers in different villages within China, pitch tends to differ significantly between the villages, but falls into given clusters within a village. All these effects, which are mostly of ethnical or social nature, do also have a relevancy for globalization, as they often overlay the language factors and show an even stronger effect in the attributes. Nevertheless, languages, or even more language families, and ethnical groups, are often related in terms of geographic distribution, hence the language is still an appropriate criterion for setting speaker class boundaries for comparison when needed.

While it is difficult to measure both types of effects independently, literature provides a sufficient evidence to consider either as source for local effects. In our studies, we will refer to both categories of aspects as aspects of *language and culture* or just cultural aspects. In general, these cultural aspects are mostly influenced by a combination of geographical regions, ethnical, political, and social factors. Studying these effects, one should also be aware that language and culture are always in flux, changing with generations and fashion. Younger generations for example often use language as a practical means to distinguish themselves from their parents (Schötz, 2007).

2.2 Digital Signal Processing

Physically, sound is generated by moving molecules in a medium such as air. This movement always happens in waves and occurs with a material and condition specific velocity c (usually 343 m/sec. in air). Based on shape and occurrence, we can distinguish different types of sound, such as tones or noise. In this thesis, we are interested in one particular type of sound, which is speech sound, i.e. sound generated by speaking. Speech sound has very particular characteristics that do not apply to sound in general, and many of the features and algorithms we will consider can only be applied to this type of sound.

Sound is a continuous phenomenon in nature. And since it can have arbitrary complexity, there is no way to accurately describe every possible sound digitally. However, for any type of machine-based processing of audio, we need to transfer it into some digital or binary form, e.g. a file on disk. This transformation is actually a two-step process: In the first step, the sound is recorded using a microphone, which converts it into an analog signal consisting of electric impulse waves. The amount of the original sound characteristics preserved, i.e. the quality of the signal, depends on the type and quality of microphone and conductors used

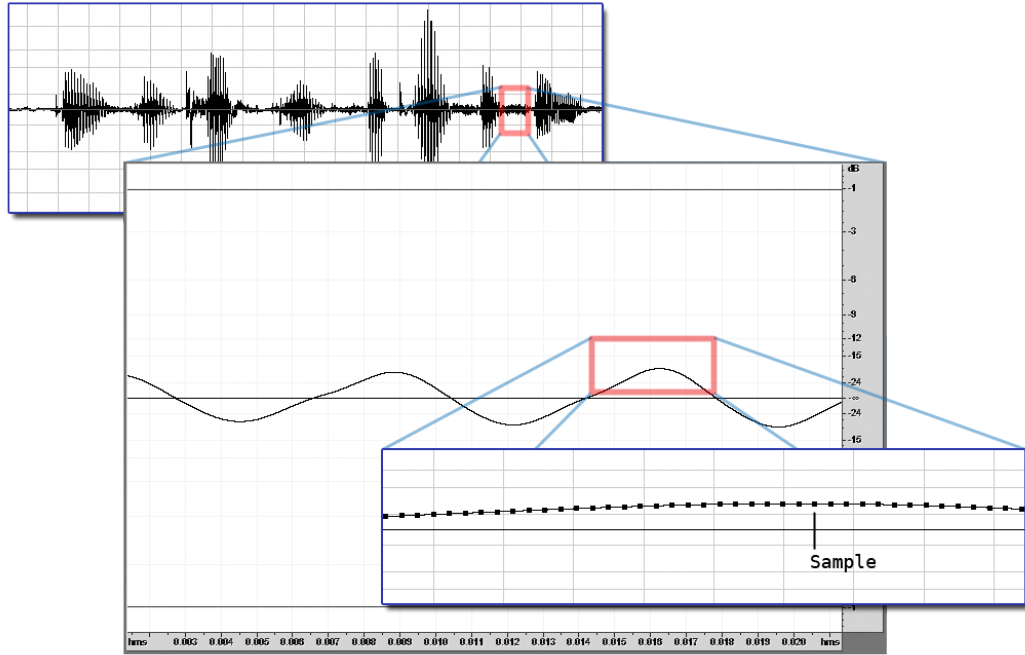


Figure 2.2: Close-up view of a raw digital audio signal, showing the individual samples as generated by pulse code modulation, and the (demodulated) continuous signal.

(Plichta, 2002). In the second step, the analog signal is converted into a digital signal by a so-called A/D converter. This process is referred to as *digitizing*. For sound waves, the by far most common procedure is to capture the analog signal at discrete points in fixed intervals and then map it to a finite set of values, which is also called *sampling* and *quantization*, respectively. (Generating a mathematical function that describes the signal would be an alternative digitizing method.) Following this, there are two main parameters describing this process: the interval length and the target range of values. In conjunction with digital sounds, the first is called the *sampling frequency* or *sampling rate* (specified in Hertz) and the second is the *bit depth* (specified in bits per sample). For completeness, a third relevant parameter is the *number of channels*, which is greater than one if multiple sound sources are processed in parallel, e.g. to produce stereo sounds. A single data point in the digital signal is called a *sample* (see Figure 2.2).

Digitizing is a lossy transformation: Both the number of samples, as well as the precision of a single sample, limit the accuracy with which the original signal can be modeled. In nature, an accurate power value can be determined between every two instants of time and has virtually infinite precision. From the digital signal, such values between two samples can only be estimated, for example by interpolation. Some types of digital codings support this interpolation by storing additional data. These considerations affect the reconstruction of the original signal, e.g. when the recorded sound is played back or processed. There are certain well-known side effects that can occur during digitization, especially if either sampling rate

Range	Description
85 - 180 Hz	average male voice F0 for speech
165 - 255 Hz	average female voice F0 for speech
75 - 500 Hz	total F0 frequency range for speech
80 - 1,100 Hz	typical F0 frequency range for singing (E2 to C6)
4,000 Hz	maximum frequency for many telephone applications
60 - 7,000 Hz	total voice frequency range for speech (incl. harmonics)
15,000 Hz	highest frequencies produced in singing
20 - 20,000 Hz	Human hearing range
44,100 Hz	Sampling rate of a typical audio CD
192,000 Hz	Maximum sampling rate of many high-definition media

Table 2.2: Frequencies and ranges relevant to the digital processing of human voice. *F0* refers to the fundamental frequency.

or bit depth are chosen too low or are set to an incompatible setting. An important one is *aliasing*, which can be the result of a sampling rate that is too low to represent a frequency from the original signal. The *Nyquist–Shannon sampling theorem* states that only frequencies below or equal to half of the sampling rate of the digital signal (the *Nyquist frequency*) can be reconstructed without distortion. Thus, to achieve perfect reconstruction of the original signal, the sampling rate would have to be chosen at least twice as high as the highest frequency in the continuous signal. For arbitrary sounds, which are not band-limited, this is not possible. However, for speech, and particularly the features we want to consider, we can make a helpful assumption about the frequency range: Since most frequencies of speech sound are below 7 kHz (including the harmonics), we can expect sound sampled at 14 kHz or higher to capture all features related to the waveform. Other important frequencies are listed in Table 2.2.

Common bit depths today are 16, 24, and 32 bits per sample. Higher values can be perceived as showing a greater acoustic dynamics. For the algorithms we will use later, 16 bits is sufficient. Other types of digitization effects include noise, DC offset, clipping, and skewing. Further, to reduce the required bandwidth, compression is sometimes applied, which can have a critical impact on the data. Voice processing applications should therefore do without compression if possible.

When interpreting or processing an audio signal, there are two major representations utilized in speech sciences: the *waveform* view and the *spectral* view. The waveform is probably the most natural representation. It consists of the raw values representing the air pressure measured as *power* or *energy*, and expressed as displacement from zero in either direction. It is also called *amplitude* because of this oscillating shape. While the value range in digital form is normalized, e.g. to the interval $[-1, 1]$ for 32 bit PCM audio, a more common measure is the logarithmic decibel (dB) scale, which reflects the human perception of sound volume. This view more clearly exposes loud and silent passages of the waveform, and intensity-based

measures can be directly extracted from this representation. On the other hand, frequencies, especially when they overlap, are difficult to detect. A much more convenient view in this case is the spectrogram, which shows the energy in various frequencies (or rather frequency bands) on the time axis. To get from the power signal in LPCM notation (i.e. the raw, absolute sample power values) to the spectrogram, a *discrete Fourier transform* (DFT) is performed. This very common method in signal processing breaks down a signal in its frequency components, resulting in a mapping from frequencies to energy, but ignoring any dynamic aspects. The most popular algorithm is the *fast Fourier transform* (FFT). To arrive at the spectrogram view, the FFT has to be performed on different parts of the signal using a sliding window. In this case, it has to be ensured that the window is sufficiently large to capture the desired frequencies. Figure 4.4 on page 111 shows a waveform and a spectrogram view of the same sound.

2.3 Age and Gender as User Characteristics

As we have seen, the information content present in a person’s voice is high. From all possible types of paralinguistic characteristics that could be extracted, this thesis focuses on two: age and gender. These two particular properties have been selected for both theoretical and practical reasons. One of the former type are the plethora of interesting applications that can be realized when we have the ability to non-intrusively detect a persons’s age or gender. A high concentration of these applications falls into the domains of automotive services, telephone-based applications, and mobile devices, hence the choice of these domains for a more in-depth look at practical examples in Chapter 6. More application-related aspects will be discussed further down in this section.

Then, there is a “quality vs. quantity” kind of argument: The number of properties was limited to be able to focus on the parameters and performance of the approach instead of broadening it to further properties. This “broadening” is actually done on the conceptual level in Chapter 5 and Chapter 6, without implementing the methods. It is believed that the concepts developed here are easily transferred to other speaker characteristics since they share many patterns, i.e. that the foremost challenge is the optimization of a single classification pipeline, and the minor challenge is the adaptation to other properties.

A further point for their consideration is that the recognition of age and gender is working well enough to be suitable for practice. It is an active field in which a substantial amount of progress has been made in recent time, including the numbers in Müller (2005). While we can think sufficiently abstract to see the potential of other methods which are not yet that accurate, the effect of personalization is easier to demonstrate when all parts are working together live. Related to this, it should also be stated that the know-how and experience present in the field of age and gender classification was supportive of the decision as well, although not decisive.

A more practical reason for choosing these speaker properties is the availability of data. The persons that are in charge of the application of Speaker Classification, but not the method,

are often initially not aware of the importance of this matter. To be able to learn to recognize paralinguistic information, a large database – a training corpus – of spoken material has to be present illustrating the property to be recognized using a variance resembling realistic conditions. For example, to learn how to estimate age, a corpus of utterances from different speakers of varying ages is required, including sufficient old speakers. To yield acceptable results, annotation is usually a requirement as well, so only a corpus where the age was specified during recording can be used. For age and gender, such corpora can be created with limited effort, and several ones are available for scientific use (see Section 4.1.1). It is considerably more difficult to create a large corpus for certain other properties such as emotion, stress, arousal, or alcohol level, because of the issue of how to reliably put the speaker into the desired condition, where full confidence in the correctness of the ground truth annotation is usually not granted. A common example is emotion, which is hard to provoke or simulate (see e.g. Schuller, Batliner, Steidl, & Seppi, 2011). And if none of the available corpora is suitable for the task, recording a new one is difficult, time-consuming, and expensive.

When we refer to the “age” of a person in this study, we always refer to the *chronological age* or *calendar age* (Schötz, 2006) of a person unless stated otherwise, i.e. the amount of time that has passed since the date of birth. This is an objective measure and is relevant for many applications. Nevertheless, there are other common meanings of age as well, which should be mentioned for completeness. The *biological age* refers to the relative position of an individual with respect to its expected life span, and it corresponds to the maturity and aging of cells, organs, and body functions (see ibd.). It is “influenced by factors like psychological, biochemical, and neurological condition” (Porat, Lange, & Zigel, 2010). The biological age is difficult to measure. The term *perceptual age* or *perceived age* (sometimes divided into *subjective age*, *personal age*, and *other-perceived age* (Barak & Schiffman, 1981)) is used to describe how old a person appears to be, i.e. how his or her age is perceived. Schötz (2007) defines it as “the mean age of a speaker as estimated by a group of listeners”. This may include perception based on the voice of a person. The *social age* is related to the progression of an individual in social structures, which depends on his or her social environment (Barak & Schiffman, 1981).

2.3.1 General Applicability

The question of what paralinguistic information a voice contains and how we can extract it would be purely academical if it were not for the applications it enables. There are numerous scenarios in which the non-intrusive acquisition of user information is beneficial. Some examples have already been given in the introduction. Here, four major application categories shall be discussed.

User Adaptation. The user interface of today’s applications is much more forged to suit the user’s need than ever before, especially with respect to constrained environments, including

cars. User adaptation, i.e. the presentation of a different interface or behavior depending on who is using the system and in what state he is, is one way of how this personalization can be achieved that does not depend on the user's explicit cooperation. In the automotive context, an example might be an increased lead time to a warning for elderly people. This type of adaptation occurs in many other areas as well. For instance, applications which run on mobile devices can adjust their services to the speaker's demands. Also, user adaptation is not limited to hardware in the user's possession. A speech-enabled public ticket terminal could automatically suggest special tariffs and bonus programs for young people. A native application domain of Speaker Classification are phone-based services, such as call centers. Here, the caller can be forwarded to an agent (*automatic call distribution*, ACD) based on gender or age, which is a type of personalization as well. In all of these scenarios, asking the user for their age/gender information is possible, but constitutes a poor option. It would introduce an extra step that requires additional explanation of a sometimes rather subtle effect, so the use of a non-intrusive classification method is crucial. The most promising strategy is likely to be a combination of both: to begin with anonymous profile containing only the estimated demographic data, and allow the user to customize the initial estimates later on (including logging in to reveal his identity). Chapter 6 deals with these aspects in much greater detail.

Semantic Labeling. For tasks in semantic computing or data mining, it might be necessary to annotate speech recordings with age, gender, and other characteristics off-line. This can be used for statistical purposes (e.g. to discover the average age of people calling in to a phone-based service), which has applications in demographic analysis and marketing. It could also be used with recordings of meetings or other arrangements with multiple speakers, which allows a better idea of the situation to be formed, e.g. we could distinguish between adults talking to a child, a man speaking with a woman, or seniors having a discussion. In the latter case, there lies an additional problem in associating voices to different people (the diarization problem), which can be solved separately or cooperatively with the determination of paralinguistic information.

Supportive Technology. Directly following from the diarization example in the last paragraph is the possibility to use knowledge about age and gender of a voice to aid other modules in their work. For instance, diarization algorithms could use the information about age and gender in a short voice segment to improve speaker separation. Similarly, speech recognition could also be improved by such background knowledge, e.g. by loading speaker class-dependent recognition profiles.

Forensics. Related to crime or security in general, identifying the age or gender of a speaker can be quite useful. When a particularly suspicious phone call is analyzed, the system can support a human expert in assessing a speaker's profile or identity, by using a completely different and thus complementary approach. System support is even more desperately needed

when hundreds or thousands of calls have to be analyzed, e.g. to filter candidates for a possible match with a particular identity or person profile.

2.3.2 Age and Gender in the Automotive Domain

People have very different driving styles, communication habits with other road users, preferences as to what locations they want to visit, where they would go shopping, and so on. Compared to a mobile phone, where personality usually affects only “soft” aspects such as menu design or maybe messaging, a car has a much stronger “presence” in the environment, resulting from its size, motion, social, and safety implications: For example, it can be used to travel with others, and of course it can also have serious effects on safety if not handled properly. Finally, it can also contain much of the other electronic equipment that would be subject for personalization today (phones, media players, navigation devices...). Even though every single person shows a different behavior and taste, both age and gender are good indicators for what they might be like for an unknown individual as long as there is no contrary evidence. A simple example of how to exploit the user characteristics this way would be a recommendation of shopping locations or sights based on the gender information, or the choice of a different voice for the (possibly rented) car’s speech synthesis. Men and women are known to prefer different cues for spatial orientation, so an advanced navigation system could point to the appropriate structures in the environment according to the gender.

Even more needed than the last examples are systems that adjust themselves to the user’s limitations and physical needs. Age and gender can both indicate such needs. Most obviously, age tells us something about the driver’s skills: An elderly driver is much more likely to suffer under bad lighting conditions, drowsiness, and also has a lower reaction time on average. Warning delays, lighting settings, volume, and choice of modality are examples of areas where adjustments can prove helpful and even reduce accidents. A young person may have better reaction, but may also have less experience and drive more risky, so certain hints given by the car, provided in the right style, can be effective. Nasoz, Lisetti, and Vasilakos (2010) claim that with respect to gender, the most common reason for accidents is also slightly different between the genders: While men drive more aggressively and more often break rules such as speed limits, women suffer from lower confidence and errors in perception or judgment. Apart from that, there are roads or stops that should be avoided by women, but also parking lots especially reserved for them. A user-adaptive navigation system would respect this in its recommendations.

As it becomes evident in these examples, it does not take much effort to realize that both the age and gender information have a lot of potential in practice. The next step will be to study how we can obtain it automatically and without disturbing the user.

2.4 Automatic Recognition of Speaker Characteristics

Speaker Classification is the subject with the goal set to find solutions to the question of *how* a machine can determine characteristics of a speaker, such as age and gender. In this context, Speaker Classification always refers to an automated process, and not the neuronal and mental processes that a human would initiate upon hearing a voice. A definition could be given as follows:

Speaker Classification deals with the automatic assignment of a segment of recorded speech data from a single speaker to a fixed set of classes, which describe certain properties of the speaker or his/her state.

This definition is a slightly refined version of the definition given in the introduction to the Speaker Classification book series by Müller (2007). Note that in this definition, the speaker's state is explicitly included, which, in addition to primary properties like gender or height, also allows emotion or the language currently spoken to be used as possible class-defining criteria. To reach its goal, Speaker Classification research deals with a larger set of problems (see Hill, 2007). This involves a study of the properties which are being examined, collection of data, extraction and analysis of possible features, building of a classification apparatus, and the experimental investigation of different system and parameter configurations. After putting the field into the scientific context, the next sections will detail on the tasks involved with Speaker Classification.

2.4.1 Embedding in Speech Science

Speaker Classification is an interdisciplinary research area. Therefore it would not be adequate to treat it as a sub-field of a single other field. Rather than that, it uses knowledge and techniques from many other areas of science. These particular areas and their connections are illustrated in Figure 2.3. For better structuring, the illustration distinguishes between those aspects of Speaker Classification that investigate the causes of differences in the human voice (left side), the core aspects dealing with the development and testing of classification systems (center), and the aspects related to applications of the technology (right side).

Among the foundations, *phonetics*, a sub-field of *linguistics*, explains the essence of speech and how it is generated. Our language system consists of phonemes, intonation, rhymes, and other constructs, and phonetics provides the link between the message and the sound that it generates. When it comes to factors that influence the speech generation, we also have to look into other areas. One of them is *cognitive psychology*, which can help us explain for instance how and why certain emotions or an increased cognitive load influence our speech. Similarly, when we are interested in the natural processes changing the speech production system during our life, pathological circumstances or other anatomical differences, the field of *phoniatics*, a sub-field of *medicine*, will provide valuable insight into the anatomical or medical reasons for certain behavior that we observe. Both have a direct and indirect (through phonetics) impact on Speaker Classification (Müller, 2005).

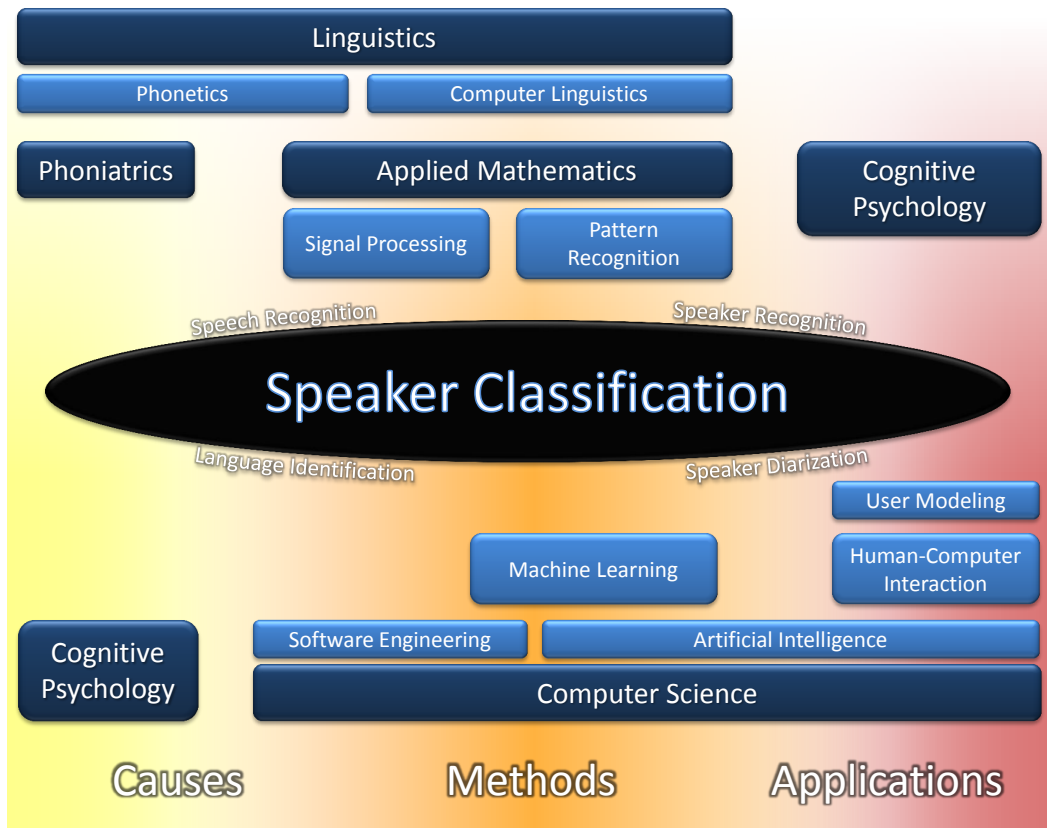


Figure 2.3: Placement of Speaker Classification in the family of sciences. Here, we do not only include the methodological aspects of Speaker Classification, but also the theoretical foundation investigating why voices differ, as well as the most common application field, which is human-computer interaction. The four areas next to Speaker Classification can be considered as sibling fields, although there are special relationships to some of them.

In terms of methods, there is a strong connection to computer science since the process itself is an automated one. Speaker Classification implies that *machine learning* and *pattern recognition* techniques are an essential part of the design. Both treat essentially the same topic, albeit the former is generally attributed rather to *artificial intelligence* as an area of *computer science* and is more generic, while the latter more to *mathematics* and *statistics*, as explained at the beginning of Section 2.5. *Computational linguistics* is also very relevant when it comes to the identification of speech-based features and their extraction, since that field has been actively developing these methods for some time. Speech recognition and speech synthesis are examples of similar technologies that are usually attributed to computational linguistics. Due to the kind of data that is being worked with, and being an interdisciplinary field itself, this in turn falls back to the basics of *signal processing*, another sub-field of

mathematics. It explains how digital signals can be filtered and processed, such as through DFT. Finally, the development of robust algorithms and frameworks, which is particularly important for this thesis, requires a considerable amount of *software engineering* knowledge. This area of computer science handles, among other things, the creation of complex systems and optimization of the efficiency of the implementation.

The application possibilities of Speaker Classification are manifold. Yet, almost all of them involve some form of *Human-Computer Interaction* (HCI), which is a further sub-field of computer science, but also has strong trails to cognitive psychology. Generally speaking, computer science gives the answers to technical questions (e.g. How can age adaptivity be integrated into an existing shopping application?), while psychology evaluates which solution should be preferred from the user's point of view (e.g. What is the user's reaction to the adaptation of this interface to his/her age?). As Speaker Classification is about particular properties of the speaker, the applications typically also perform *user modeling*, which is a special topic of HCI, and one of the types of applications it enables is *personalization*.

There are some fields and technologies that are directly related to Speaker Classification and share many aspects. Probably the closest relative is *speaker recognition*. The latter, being older and subdivided into *speaker verification* (generating a binary decision of whether an unlabeled sample is from a reference speaker) and *speaker identification* (determining by which of a set of reference speakers – if any – some unlabeled sample was produced), can even be seen as a sub-field of Speaker Classification “in which the respective class has only one member (Speaker vs Non-Speaker)” (Müller, 2007). This means that from the task point of view, the speaker identities correspond to the classes representing a collection of characteristics and features, to which an individual voice is then assigned. However, the argument could also go vice versa: From the approaches' point of view, Speaker Classification is more specific, as it will always consider features for selected user properties, e.g. emotion. This is a subset of the features which are used to describe the identity of a person, which would be the sum of all single user properties. In either case, the connections between both fields are very strong, which necessarily has an effect on the techniques which are used. These similarities are part of the reason why this work also attempts to apply the newly developed Speaker Classification concepts to a speaker identification task (see Section 6.2.6).

Very similar to speaker recognition and therefore also highly relevant is *speaker diarization*. Its goal is to distinguish the turns of different speakers, either in recorded material (e.g. from meetings) or in an on-line scenario. Providing the identity of the speaker is an optional addition; the main goal is to simply assign segments to a first speaker, second speaker, and so on. The main question it tries to answer is “Who spoke when?” (Wooters, Fung, Peskin, & Anguera, 2004; Reynolds & Torres-Carrasquillo, 2005) (or “Who speaks when?” in case of on-line diarization). Because the method is dependent on features that can separate speakers, it can benefit from the research in Speaker Classification, or even integrate Speaker Classification systems directly to aid in the segmentation process.

Language identification shall be mentioned as an example of one topic that is actually a special case of Speaker Classification, as are dialect identification, speech-based emotion

recognition, and many others known under their individual names, even though most of them exist longer. Obviously, all of these topics have a lot in common, and can all be employed for personalization.

Last, *automatic speech recognition* as one of the largest and oldest speech-based processing topics, should not be omitted in this catalog. It may have a lower relevance than the aforementioned ones, because the goal is rather different (i.e. analyzing the contents of a message and not the properties of its speaker). Nevertheless, some insight gained on speech features and methods are valuable to Speaker Classification as well. In a more indirect way, the two technologies can also benefit from each other: ASR can benefit from Speaker Classification by using (or favoring) an acoustic or even language model that matches the speaker profile, such as a *senior female* model. And vice versa, when linguistic features are considered for Speaker Classification, an ASR may actually provide their extraction, something that was already attempted by Metze et al. (2007) through the use of parallel phoneme recognizers.

Given this demarcation of Speaker Classification against other fields, the following sections introduce its most important aspects and sub-topics.

2.4.2 Fundamental Aspects

There are at least two technical aspects that all Speaker Classification approaches share: (1) They take one or more voice samples as the input. (2) For each of the speaker properties they are designed to detect, they can return the most likely class after an input sample was provided.

This merely defines some sort of a “minimal interface” for Speaker Classification technology. There can be further input a Speaker Classification system could consider, such as visual input and context knowledge, and it could also provide a more detailed output, such as likelihoods for each class. The part in between input and output is where the approaches differ. This core part resembles a decision-theoretical problem (choice under uncertainty), a very common problem handled in artificial intelligence. There are many strategies of dealing with such problems, and one of the main choices characterizing them is whether the rules for solving the problem are specified entirely by a human expert (hence called *expert knowledge*) or whether the system is at least to some part involved in the creation of the rules used in decision making. In the first case, one can also speak of an expert system, while the second is a data-driven approach that falls into the AI subfield of machine learning. The following paragraphs in conjunction with Figure 2.4 point out the differences.

Expert systems are a type of knowledge-based decision systems that can be consulted instead of a human expert. An example would be a system that can advise a pilot in operating a plane under difficult conditions, where the rules have been created manually by incorporating knowledge of experts in beforehand. A common method of implementing such a scenario are rule-based or production systems, which are rich and expressive. Examples are the rule-based languages *Jess*¹ and *Prolog*. This top-down approach has the advantage of being very trans-

¹<http://www.jessrules.com>

parent and predictable, as long as the number of factors, variables, and outputs is relatively low, and – even more important – perfectly understood by the experts. The human experts can incorporate background knowledge that would otherwise be difficult to capture for a machine, especially since it does not require actual data to be present for each special case. In other words, the system can predict solutions for conditions not observed. If the relationship among the factors or between factors and decisions is unclear, there is no advantage in having human experts. On the contrary, if decisions are based on observations only, they may even exhibit a worse performance than a machine in detecting subtle patterns.

Machine learning is the second approach to problem solving such as the Speaker Classification problem. Here, algorithms are not only used to return a decision for a concrete situation, but also to build the structures a priori bottom-up that will determine how to arrive at a decision in such a case later, which is what gives this approach its name. Observations are used to derive those structures, and they are required to be in an appropriately formalized manner. The degree to which the process is influenced by a human expert varies from selecting only basic parameters to fine-tuning the data structures. The available structures and corresponding learning methods are very diverse and their performance depends a lot on the data and the parameters of the problem.

Considering the characteristics of Speaker Classification, the bottom-up method seems as the only viable practice for the task. It is unlikely that there are any experts which could specify a working top-down solution to the problem without resorting to trial and error, as the knowledge is missing in the first place. The features we could consider as input to the system are real-numbered low-level attributes, which are sometimes hard to grasp for a human, e.g. MFCC coefficients. Also, we can hardly benefit from the expressiveness of a complex rule-based system. Finally, we are also not capable of finding the rules that will give optimal results by hand. This is due to the massive amount of data that we would have to analyze in the absence of a simple and reliable relationship between features and results. It is also due to the fact that some of the factors are highly correlated, so that the problem becomes an optimization problem of finding a global maximum over a potentially large value range and large set of input factors. On the contrary, methods from machine learning are more focused on computational solving of problems and hence less accessible to humans. Typical examples are artificial neural networks and support vector machines. More information on the relevant aspects of machine learning are given in Section 2.5. Some structures like Bayesian networks are suitable for both the machine learning and the manual approach.

There are some basic principles one has to be aware of. First, there is no perfect “solution” to the problem of Speaker Classification. The performance of a system can only be as good as the data it’s based on. And unless someone discovers a feature in the human voice which undoubtedly exposes the speaker’s attributes, which is highly unlikely according to the reasons given in 2.3 (and generally not true in biology), there will always be wrong answers from the system. In the opposite, speaker gender and even more age represent particularly difficult problems. And with increasing difficulty of a decision problem, the difference in performance of various methods also becomes more significant. The reason lies inter alia in the complexity

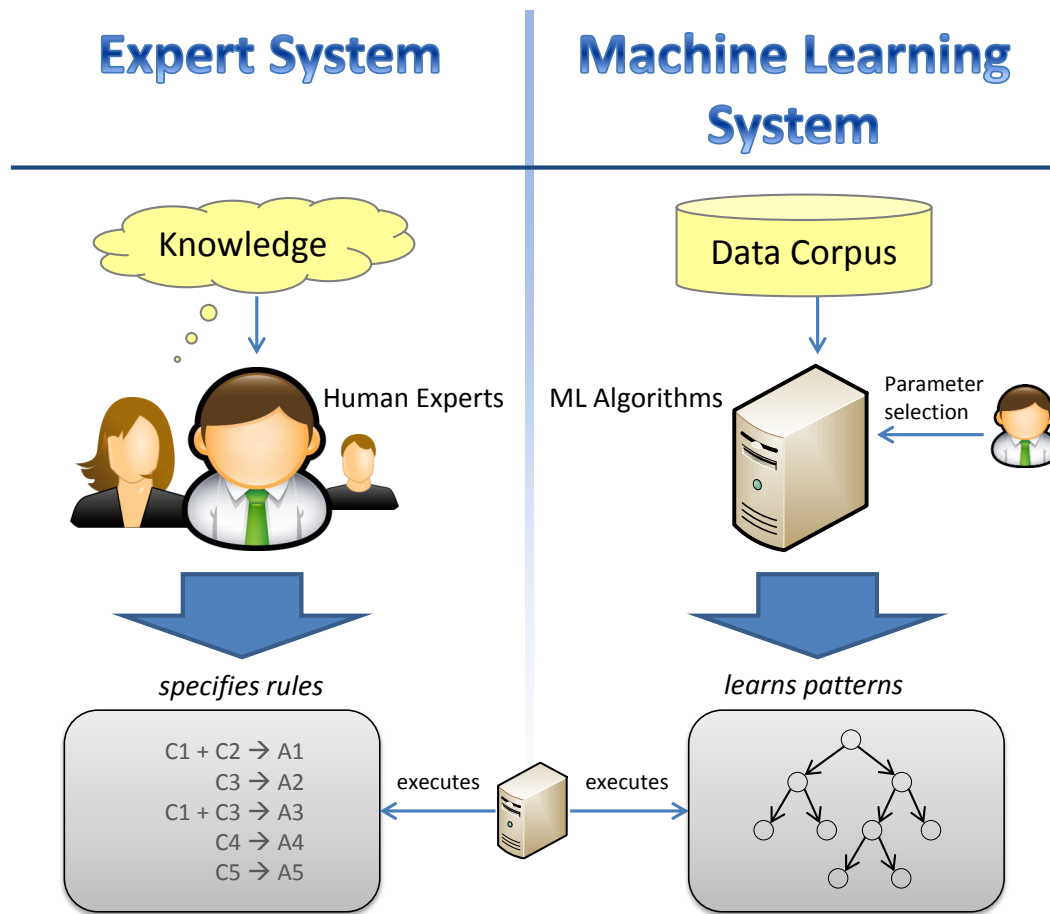


Figure 2.4: Expert systems vs. machine learning systems: The main difference lies in the way decision-supporting data structures are created.

of the decision process, the amount of knowledge involved, and the fact that some systems can deal far better with uncertainties than others.

2.4.3 Tasks Associated With Speaker Classification

Hill (2007) gives a good introduction to Speaker Classification and its challenges by connecting it to speaker recognition, a related field in which there is already a lot of experience present. The following paragraphs summarize the most important tasks that are worked on in Speaker Classification for specific classification problems.

Domain Analysis. The first task when dealing with a Speaker Classification problem is to build an understanding of the speaker property or properties which are going to be estimated, and how they should be used. If we want to classify the emotion of a speaker, we first need to think about which emotion model we want to apply, which also determines the classes that

we will define. If we want to classify intoxication, we first need to do research on what types of substances we want to detect and with what granularity. When trying to classify age, we have to think about how many age classes are required and what ages they should span. As a result of thinking about the problem domain, an initial set of classes is created. After repeated experimentation, the final class set may still look different, for example because combining certain classes that cannot be distinguished well is more rewarding. In this context, we should also get an initial idea of how critical errors are, i.e. what error rate is still acceptable. Overall, a document detailing the specifications of the task should be the outcome of the first step.

Data Acquisition. The next step comprises ensuring that the required data is available. This can turn out to be an actual blocking criterion, since the number of available corpora that are large, well-balanced, recorded in clean manner, and conscientiously annotated is limited. Recording a new corpus that adheres to these criteria is usually too time- and cost-intensive. When investigating speech corpora, it should be ensured that the desired classes are present with a sufficient number of speakers and length of material for each speaker and class. Preferably, the classes should not be biased towards a certain gender or age. If there is a bias, the results cannot necessarily be generalized. This is also true when the corpus is language-specific.

Feature Analysis. As Hill (2007) states, coming up with a good selection of features is probably the core task of Speaker Classification, as opposed to the classification of some “random” multi-dimensional vector (this would merely be part of a pattern classification task). Hence, the initial selection of features should be made manually and according to our knowledge of the speaker properties. As a simple example, consider that, for the detection of gender, the different vocal fold length for men and women results in a different pitch, so we might want to focus our first analyses on pitch-based features. There may be multiple ways to choose from how the feature can be obtained, such as through cross-correlation or auto-correlation. Plotting the feature for different samples may reveal derived measures that are worth studying, such as extrema, percentiles, or periodicity. Finally, statistical analyses, e.g. value distributions or correlation with other features, complete the picture.

Classification Approach Set-up. This is where the main engineering work is needed to put together a system that can learn and apply a model based on the features, and also fulfill any other conditions specified by the task. Like for the feature selection, choice of a method should always be guided by a hypothesis of why certain methods are expected to work better. The suitability of classification algorithms depends very much on the feature values and the general type of feature. For instance, GMMs (Gaussian Mixture Models, a type of classifier explained in detail later) work well with normally distributed frame-based features such as MFCC coefficients, while HMMs are preferred to model the “shape” that phonemes represent in case of speech recognition. Based on the feature analysis, an elaborate choice of features

to include can be made. It is still possible to add automatic feature selection in order to optimize the outcome and deal with those aspects that are not easy to see with the eye.

Experimentation and Result Analysis. This comprises the task of testing a system under different parameter configurations, comparing the results, and drawing the appropriate conclusions. These, of course, can present further reason to change the approach or the features in another iteration. Finally, it might be a good idea to consider the capabilities of human beings in the quest to improve the machine’s performance. This means that studies on how well humans perform characterizing certain speaker qualities and what features they base their decisions on can lead to greater insight and also puts the results into a meaningful relation to natural skills. In Section 4.6, we will follow up on a specific aspect of this, namely how a human characterizes the same quality in synthesized voices, which can in turn be used to confirm certain findings about the features that are believed to affect human decision regarding speaker classes.

Quite often, all of the aforementioned aspects are considered in this order for a particular problem at hand. Sometimes however, only selected tasks are covered, while the others are assumed as solved. This is the case for example in classification “challenges” (see Section 3.4), where data is provided externally and mainly existing background theory is applied to a new domain. Conversely, progress on features for Speaker Classification can also be made without evaluating the whole system.

2.4.4 Acoustic Features Used to Model Aspects of Voice

This section provides an overview of some of the most frequently used acoustic short and long term features in Speaker Classification applications, including those explored in the AGENDER and FRISC experiments.

Fundamental frequency. More commonly known under the term *pitch*², the fundamental frequency is the lowest frequency in the speech signal. It is also called *F0* in reference to the formant with index 0. Sounds with a higher F0 are also perceived as sounding “higher”. Because of vocal tract resonance, frequencies of integer multiples of F0 are additionally present in the signal, and are called harmonics. Pitch is computed for a window, of which the size depends on the smallest frequency to detect (for voice, 75 Hz is a common value). A common window function for pitch computation is the *Hanning* window. One method of obtaining the pitch value involves computing the normalized auto-correlation of the signal and dividing it through that of the window function (Boersma, 1993). The cross-correlation represents an alternate method. The result is interpolated and scanned for local maximums. The largest maximum is the best candidate for the frequency. The interpolation depth determines the

²Some authors also distinguish between pitch for auditory and fundamental frequency for acoustic properties, e.g. Ladefoged (2006, p. 23)

precision of the result. Computing pitch in this manner for a complete utterance, i.e. by applying a sliding window, the *pitch contour* of the signal is created. This curve may have gaps, as there will be no pitch in unvoiced regions. Most work related to Speaker Classification uses pitch as a global feature. The most common actual metrics are its mean, median, minimum, maximum, and standard deviation per utterance. The latter can be considered a measure of the frequency tremor. Another functional is the first order derivative or slope of the pitch.

Formants. In the vocal tract, some harmonics that are close to its resonance frequency will be amplified and represent further peaks in the spectrum of the voice. Their location is roughly predictable within a certain Hertz range. These *formants*, called $F1$, $F2$, and so on, are said to describe the quality of vocal sounds. Similar to pitch, their exact location and energy can be characteristic for individual speakers, although a strong correlation to the former can be expected (Assmann & Nearey, 2007).

Jitter. While changes in the fundamental frequency and also formants have a clearly audible impact heard as the pitch of the voice, there are also much smaller frequency variations known as jitter. Although these variations affect the pitch as well, the changes occur so rapidly that they are neither measured through the above F0 metrics, nor are they perceived as such, but rather as an “unsteadiness” or less “clean” voice. In the data, it can be measured by looking for changes of pitch in successive windows. Jitter is often specified relative to the average F0 in the signal. Apart from the simple mean, *Praat*, a free speech analysis tool commonly used by phoneticians (Boersma, 2001), also defines the several measures of jitter: *relative average perturbation* (RAP), which considers the mean difference to two adjacent windows; *five-point period perturbation quotient* (PPQ5), which is “the average absolute difference between a period and the average of it and its four closest neighbors, divided by the average period”; and the *difference of differences of periods* (DDP), which computes “the average absolute difference between consecutive differences between consecutive periods, divided by the average period” (Boersma & Weenink, 2011).

Intensity. The intensity describes how loud a signal is perceived. It measures the mean amplitude in a given region, i.e. $\frac{1}{t_2-t_1} \int_{t_1}^{t_2} x(t) dt$, where $x(t)$ returns the amplitude at point t in the continuous signal. This follows the definition used by *Praat*. Both intensity and amplitude are measured in dB. A popular derived measure is the log energy, which the *Hidden Markov Model Toolkit*³ (*HTK*) computes as $\log \sum_{t=1}^N x_t^2$, where $x = \{x_1, \dots, x_N\}$ is the discrete signal. In conjunction with the audio signal, the terms *energy* and *power* are used interchangeably with intensity unless stated otherwise. The intensity contour, which can be obtained in analogy to the pitch contour, is essentially a smoothed absolute value version of the signal. A problem with intensity is that it strongly depends on the recording situation,

³<http://htk.eng.cam.ac.uk>

such as the type of microphone and the distance from the speaker to the microphone. Even when only relative values are used, the feature is less robust than most others to noise and recording conditions, which easily outweigh the influence of speaker properties.

Shimmer. In the same way that jitter describes micro-variations in the frequency domain, the shimmer measures equally small variations of the amplitude. Since the ability of people to produce sounds with a smooth amplitude varies and depends on physiological characteristics, the feature carries speaker-specific information. As opposed to intensity, there is far less external interference present in this metric. Apart from the mean value specified either in absolute dB units or as a percentage of the average amplitude, *Praat* supports the *n-point amplitude perturbation quotient* (APQn), which is “the average absolute difference between the amplitude of a period and the average of the amplitudes of its neighbors, divided by the average amplitude” ($n = 3, 5, 11$, Boersma & Weenink, 2011).

Harmonics-to-noise ratio (HNR). Named after the signal-to-noise ratio from statistics, the HNR describes the amount of harmonic content, measured as the degree of periodicity, in a segment of speech. *Praat* calls this feature *Harmonicity*. It is expressed in dB. A low harmonics-to-noise ratio can make the voice sound hoarse (Yumoto, Gould, & Baer, 1982).

Mel-frequency cepstral coefficients (MFCCs). While the spectrum of a signal expresses how energy is allocated to the different frequencies in the signal, the cepstrum does the same for the spectrum, hence it is also called “spectrum-of-a-spectrum”. In other words, it reveals short-term periodicity that occurs in the frequency dimension of the spectrum⁴. The MFCCs (see S. B. Davis & Mermelstein, 1980 for some of the most original work on the coefficients) are a special type of cepstrum representation: They map the spectral powers to the Mel scale, which is a logarithmic scale that describes how frequencies are perceived by the human ear. Hertz can be converted into Mel by using the formula $2595 \cdot \log_{10}(1 + \frac{f}{700})$, where f is the frequency in Hz. As a consequence, large changes in the Mel-frequency domain also correspond to large perceived changes of frequency, which is a clear advantage of this representation. Further, multiplicative channel effects become additions in the cepstral domain and can easily be removed through cepstral mean subtraction (Young et al., 1999). The MFCCs are typically computed in frame-wise manner by first applying an FFT to a window of the original signal (see Section 2.2), then converting the Hertz scale to the Mel scale by applying a triangular filter band with N channels (see Figure 2.5), and then applying a linear cosine transformation on the resulting values. This transformation uses the following formula in *HTK* (see Young et al., 1999, ch. 5.6):

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^N m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right),$$

⁴In the time domain, similar features are called *temporal patterns* or *TRAPs*.

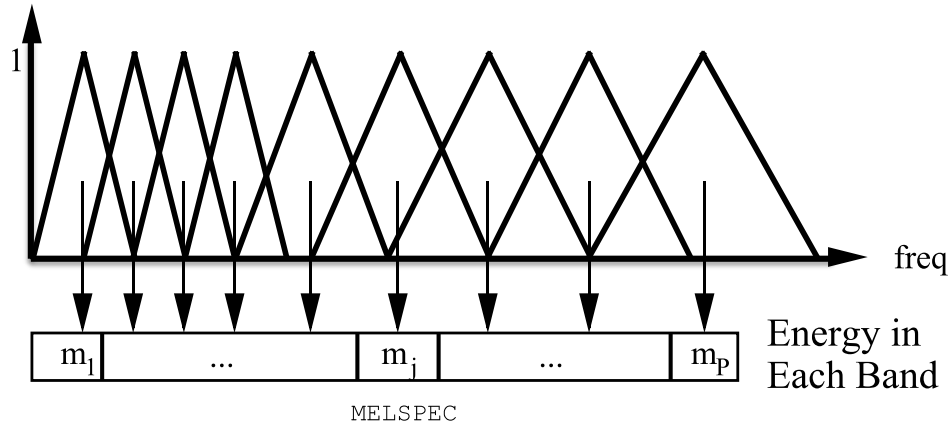


Figure 2.5: Triangular filter banks used by *HTK* to spread frequencies across the Mel scale (*binning*). (Source: Young et al., 1999, p. 65)

where N is the number of channels in the filter bank and m_j are the log filter bank amplitudes. MFCCs are used in all types of speech processing due to their high discriminability, but they are known to be particularly related to the vocal tract configuration and therefore a preferred feature in speaker recognition and related areas. Here, they often outperform the spectral features.

MFCC deltas and acceleration. The first and second order derivatives (regression coefficients) of the MFCC features, or *delta* and *acceleration* features in short, are often included to account for the change in cepstral activity over time (Kockmann, Burget, & Černocký, 2010). Some applications do even consider third and higher order derivatives. The formula to compute each coefficient in *HTK* has the following form (see Young et al., 1999, ch. 5.9):

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta (c_{t+\theta} - c_{t-\theta})}{2 \sum_{\theta=1}^{\Theta} \theta^2},$$

where d_t is the delta coefficient at time t , $c_{t-\theta}$ and $c_{t+\theta}$ are static coefficients, and Θ is the window size.

2.5 Pattern Classification

Having discussed the basic tasks of Speaker Classification and some of the options to solve them, this section provides the background knowledge for one of its most vital methods.

One of the commonly encountered tasks in computer science today deals with the automatic, i.e. computer-based recognition of structures. The topic in which solutions to these tasks are systematically developed is called *pattern recognition*, or more specifically *pattern classification* if the result represents a set of discrete “categories”. The topic has changed from a niche topic that it was approximately forty years ago to a core topic of AI (Duda, Hart, &

Stork, 2000, p. xvii). Some examples where pattern recognition problems are usually involved include speech and speaker recognition, optical character recognition, face recognition, scene analysis, gesture recognition, plan recognition, energy pattern classification, network attack detection, and many others.

Pattern recognition is one aspect of machine learning that deals with the mathematical and statistical backgrounds of the models used to solve the problem of detecting patterns in empirical data. Both machine learning and pattern recognition grew out of different sciences and motivations around forty years ago, but as of today, they can be considered simply two different views on a the same or at least a similar problem. Pattern recognition highlights the statistical aspects, while machine learning highlights the algorithmic, computer science motivated aspects (Bishop, 2007; Cornuéjols & Miclet, 2008). This is why in many situations these terms may be used interchangeably.

This section gives an introduction into pattern classification by explaining terms and presenting the most vital concepts, including several classification algorithms. For a more in-depth look, the book by Duda et al. (2000), on which several of the following descriptions are based, is highly recommended, in addition to the book by Witten and Frank (2005).

2.5.1 The Pattern Classification Task

Any pattern classification problem is centered around a question of the following form: Given a *sample* (also called *instance* in machine learning) X and a finite set of *classes* (also called *labels*) $\{1, \dots, n\}$, how can we determine to which of the classes the sample X belongs? Classification assumes that there is a clearly defined *truth*, which is the “correct” assignment of the sample to a single class, even if the relationship between both is not so clear. Normally, when the truth is a natural, objective, and non-ambiguous choice, such as the chronological age of a speaker, we also speak of *ground truth*. In some other cases, the relationship is purely based on assumptions, in which case we have to define a truth. For instance, when emotion is generated in speech, there is no way to objectively tell for certain what emotion is present in a given segment of speech, neither by the speaker, nor by the experimenter.

When the classes we can distinguish are siblings to each other, the problem is also said to be an *identification* problem (identification of the right class to which the sample belongs), while a binary (yes/no) classification problem with one class and one “rest” class is called a *detection* problem (detect if the sample is of type T or not). The reason why this distinction is important is because an identification task can also be transformed into a series of detection tasks, i.e. for each class the truth is obtained separately. As we will see in Section 2.5.7, this has a number of implications on the system design and evaluation.

To have a computer make a decision involving classification, it needs a *model* of the problem, and a system component to read and apply the model, which is the *classifier*. The model describes the knowledge of the system about a problem. The model can have an arbitrary contents, but the basic purpose is to describe a relationship between aspects of the input data, the *features* or *attributes*, and the classes. There are two basic types of models: *generative*

and *discriminative*. The first kind contains information used to sketch the properties that all samples of the class share. This knowledge could be used to *generate* samples that are members of this class, hence the name. Generative models are usually constrained to a single class and have no knowledge of other classes or feature values that are explicitly *not* an indicator of class membership. In contrast to this, a discriminative model stores knowledge about features that distinguish class samples from non-class samples (or multiple classes from each other). Only differences that are reflected in the features can be used for discrimination. These characteristics, which are illustrated in Figure 2.6, give both types of model quite diverse classification properties and their respective limitations. The system that will be proposed in Chapter 4 shows how both can be combined for increased performance. Features can be essentially anything, such as the size or shape of objects to be classified. In an actual system, the choice and value range of features will be fixed to the *feature space*. The most common value types for features are real-numbered, integer-valued, nominal (categorized), or binary (nominal with two categories). More complex features, such as the description of a gesture or the words in an utterance, need to be broken down into more basic attributes, such as distance values. This is sometimes called *filtering*. The classifier itself is limited to those features provided by the system, which are called *feature vector* due to the fact that the ordering of features usually remains identical for each sample in any given system. The algorithm that classifiers use is also known as the *classification algorithm*⁵, and there are several well-known algorithms such as kNN, Naive Bayes, and SVM.

Before we attempt to create a model, we should study the available features in sufficient detail. Determining quality metrics on the feature level after the initial selection instead of starting right away with classification is preferred, as features usually allow a much clearer and distinctive insight than models or final classification results do. It is part of the bottom-up approach that is fostered in this thesis. One basic method is a *histogram-based analysis* of the feature space. Thereby, for a single feature, one curve is created for each class in the same chart (see Figure 2.7). The amount of visual overlap between the curves gives a good impression of how well this single feature can contribute to the separation of these classes. Even if not all classes are set apart from each other, the feature may still be good at separating a single class from the rest, so it may be useful to also visually combine certain classes in a single curve. Instead of the typical bar shape histograms, the *Gaussian probability densities* of the features can also be plotted (see Figure 4.18 on page 143). Instead of the histogram “bins”, only the mean and standard deviation of the value range are plotted as a Gaussian curve with the shape

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

where μ denotes the mean and σ the standard deviation of the actual feature values. According to this formula, the curves are normalized to cover an area of 1. Besides being more concise and easier to read, they depict already an approximation of the performance of a simple GMM classifier (univariate with a single mixture). However, the view may also be too

⁵A classification algorithm is often also called *classifier* for simplicity.

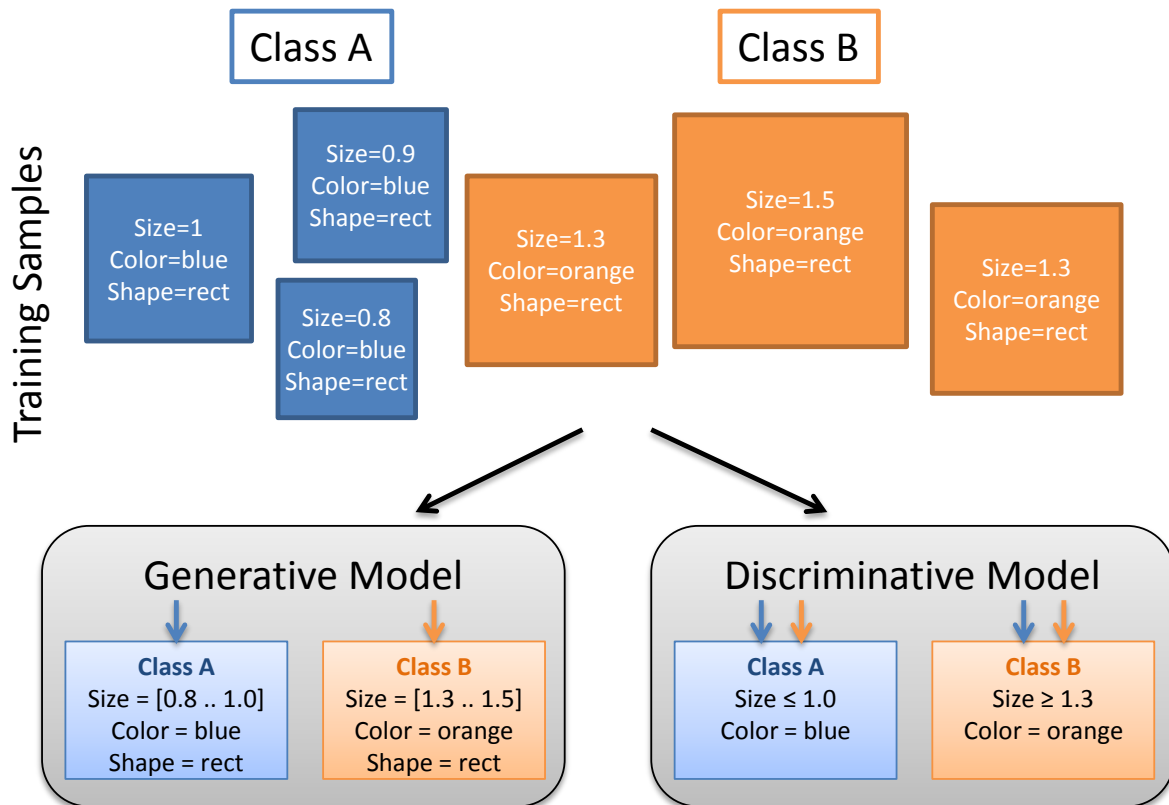


Figure 2.6: Generative versus discriminative models. In this example, a training set consists of 2 classes (A and B) with 3 instances in each class. There are 3 features (size, color, shape). The lower half displays a simplified view on the possible relations stored in a generative and a discriminative model trained on the same instances. In the case of two classes, the two discriminative models could also be merged into one.

imprecise or misleading when the feature values are not approximately normally distributed. A third feature analysis method is the correlation chart. Features that strongly correlate contain redundant information, which could make the training process slower or even decrease the classification performance. This chart plots the values of two features against each other for all samples in a two-dimensional graph (see Figure 2.8). The features must either have the same scale or be normalized first. If the samples are all arranged close to one of the diagonals, a high correlation is present.

Apart from applying a model, an important part of pattern classification is to have the model created automatically. This process is often called *training* and involves processing a number of *training samples* for which the truth is known. In addition to this *supervised training*, there is also a type of training without labels known as *unsupervised training* or *clustering*. In this case, the classes or *clusters* are not known in advance (except for rough

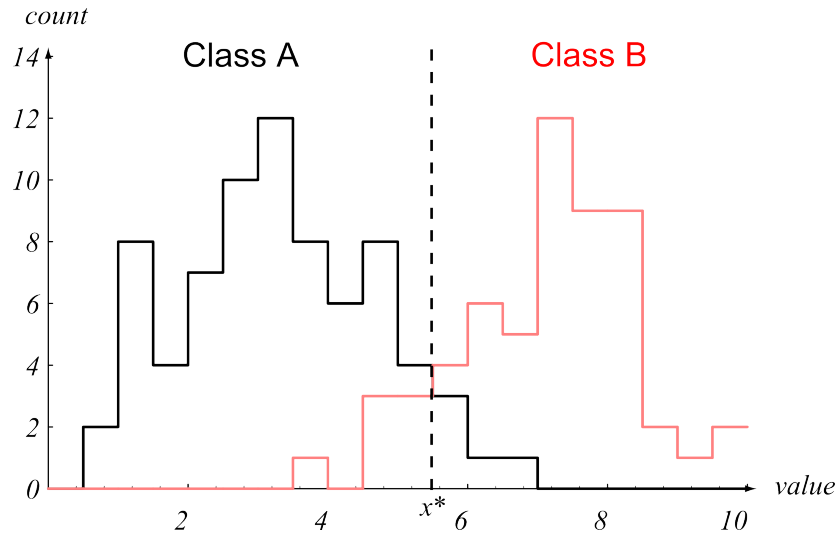


Figure 2.7: Comparison of feature values measured for two different classes in histogram form. In this example, the values are by and large separate, but there is also some overlap. (Source: Duda et al., 2000, p. 4)

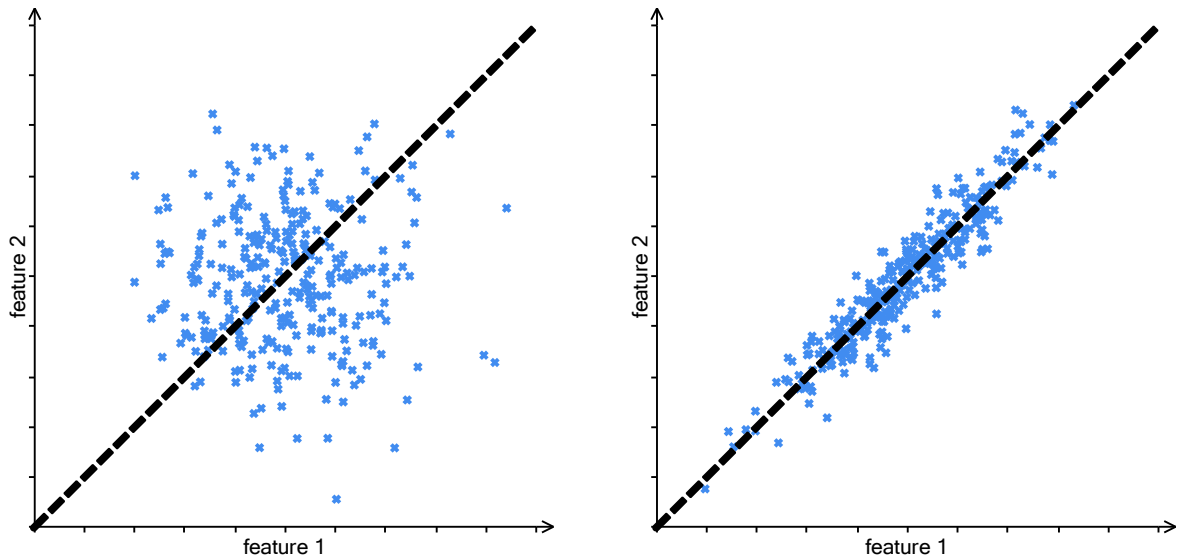


Figure 2.8: Linear correlation as observable in 2D feature plots. *Left picture:* no correlation. *Right picture:* strong correlation.

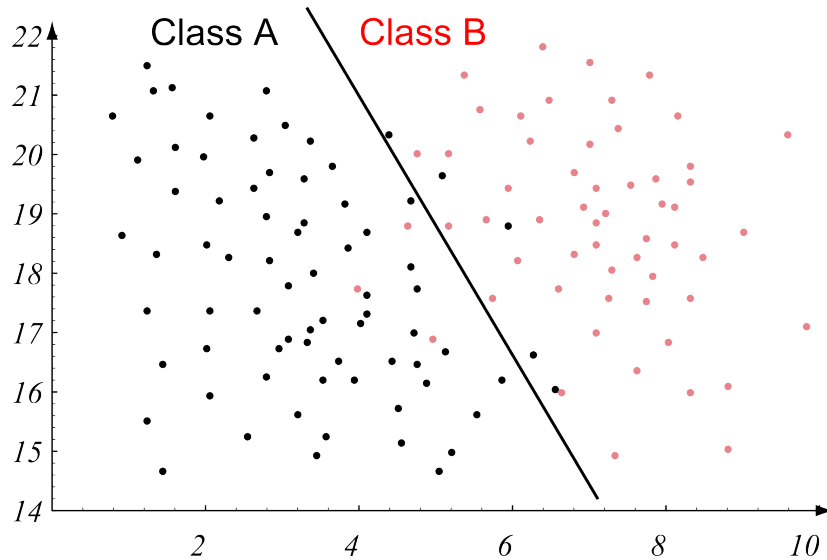


Figure 2.9: Samples of a hypothetical classification problem, plotted with respect to two feature values. The truth is indicated by color (black vs red). A black line denotes a possible decision boundary. (Source: Duda et al., 2000, p. 5)

constraints such as regarding their number), but generated in conjunction with the model. The classification algorithm is responsible for building the model based on the feature vector. Generative models are usually trained only with samples stemming from their corresponding class, while discriminative models are trained with all data.

Models often impose mathematical relations on a subset of the feature space. For example, a simple web-cam image processor could classify the time as “day” when the brightness is above a certain *decision threshold*, and as “night” otherwise – feature values below the threshold result in a different decision than values above. Normally, the decision process is more complex, because (a) there are multiple thresholds or ranges, and (b) we consider more than one feature, i.e. dimension. Therefore, we speak more generally of *decision boundaries*. If, for instance, we add a second feature, e.g. the amount of activity between two images, and create a relationship between the two features involving a single combined threshold, we get a linear 2-dimensional decision boundary that could be plotted as seen in Figure 2.9. One of the criteria that makes classification algorithms different in their ability to produce certain decision boundaries, not only in complexity, but also in general shape. Some algorithms have a very characteristic appearance when plotted in a 2D chart. For instance, some decision trees can only produce rectangular (axis-parallel) boundaries.

We can clearly see that the linear decision boundary in Figure 2.9 is suboptimal. Assuming that we use a classification algorithm that allows sufficiently complex decision boundaries, we could theoretically create an “optimal” boundary in terms of classification accuracy as shown in Figure 2.10. While this classifier could perfectly separate the training examples, it might do worse than the linear classifier under real conditions, i.e. on new examples not seen

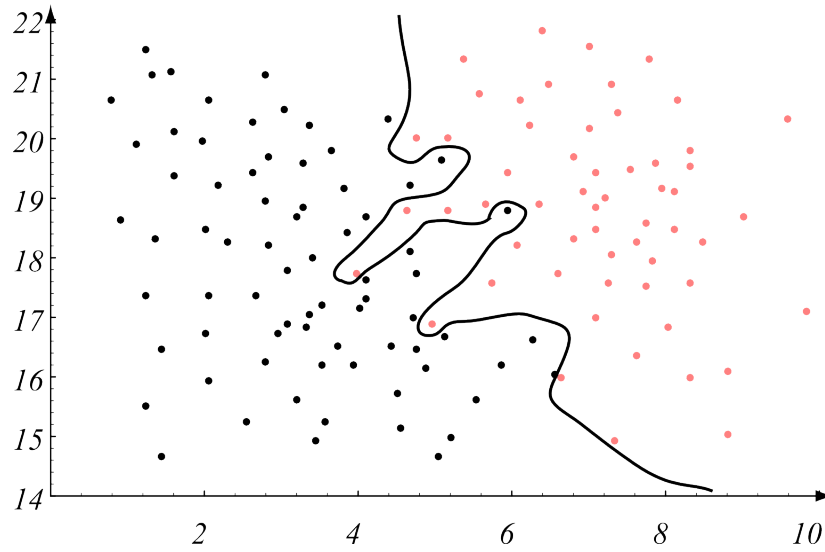


Figure 2.10: The classification problem from Figure 2.9 now uses a different, (too) complex decision boundary. (Source: Duda et al., 2000, p. 6)

during training. What this model suffers from is a problem of *over-fitting*: The boundaries are tailored to the training samples in too much detail. One can also say that it learns the actual training instances by heart, instead of learning the general properties that distinguish the classes. Duda et al. (2000, p. 7) call this quality *generalization*. A model that might reflect a better compromise between training performance and generalization is depicted in Figure 2.11.

In practice, many classification tasks cannot be solved perfectly. There are a number of reasons for this. One is related to the task itself: For some tasks, it is impossible to build a perfect classifier. If for example we want to classify an email into *spam* and *ham* (not spam) based on only the message subject, then it is obvious that a perfect accuracy can never be achieved on this data alone. But even if we have the whole message available, we can imagine that there will always be false positives. This is the classification-inherent uncertainty, which can only be mitigated by changing the input data or specification of the problem. Yet, even if a perfect feature existed in theory, which is arguably unlikely for most classification problems, we are limited to the features and extraction methods at our disposal. With most features, there is some overlap remaining between classes that we can observe in the data, even for features that appear to be good. This is often related to natural *outliers*, i.e. samples that have feature values out of the usual range: For example, when classifying gender based on voice, we might associate higher voices with women and lower voices with men. But there are also men with particularly high and women with particularly low voices, which even a good model does not cover using this feature – neither do humans. And then of course, there is uncertainty caused by a suboptimal choice of features and algorithms. A positive aspect is that we are able to quantify the collective uncertainty associated with classification: By

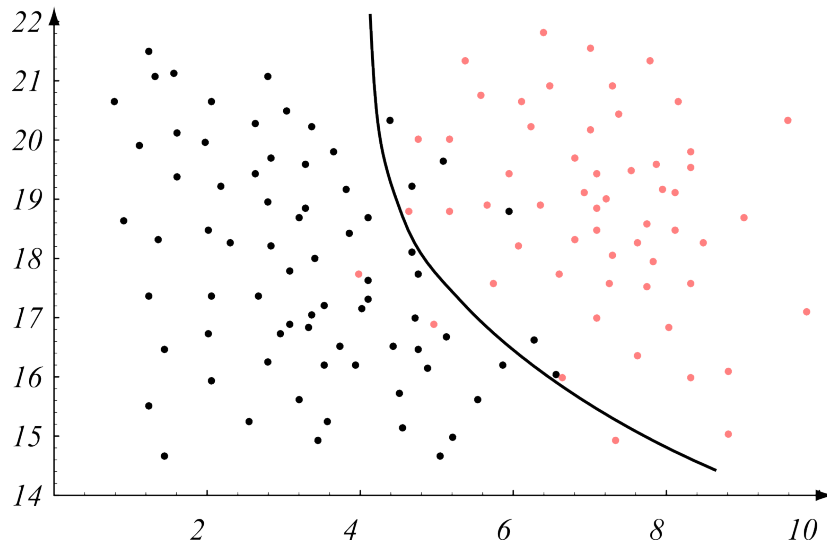


Figure 2.11: A decision boundary that represents a trade-off solution to the classification problem from Figure 2.9. (Source: Duda et al., 2000, p. 6)

measuring the classification performance in an evaluation, we can predict how accurate the model is in general. For instance, we might state that a given speaker gender classification system statistically fails for every 10th speaker.

A pattern classification system generally is expected to assign instances to classes (the actual classification), but there is more to it. Because of the uncertainty mentioned in the previous section, the system has to solve a more generalized type of *decision problem*: It must decide which result is actually better according to some external metric. A common metric is the unweighted average accuracy over all classes, which essentially means that the system should try to classify as many instances as possible correctly. But there are also cases where this “equality” of classes is not given. Duda et al. (2000) present the example of a fish processing plant separating salmon from sea bass. Obviously, due to the value of the fish, a misclassification of sea bass as salmon is less critical (from the customer’s view) than the other way round. A pattern classification system can integrate such a constraint in its decision process. In the general detection task, the corresponding metrics that are traded against each other are misses and false alarms. More generally, we speak of a *cost* measure.

According to Duda et al. (2000), the classification of an instance by a pattern recognition system can be divided into different consecutive phases or components, which are depicted in Figure 2.12: *sensing*, *segmentation*, *feature extraction*, *classification*, and *post-processing*. This arrangement is also called the *classification pipeline*. Each of these phases is described in the following. Afterwards, we give some insight into the evaluation procedure for such systems.

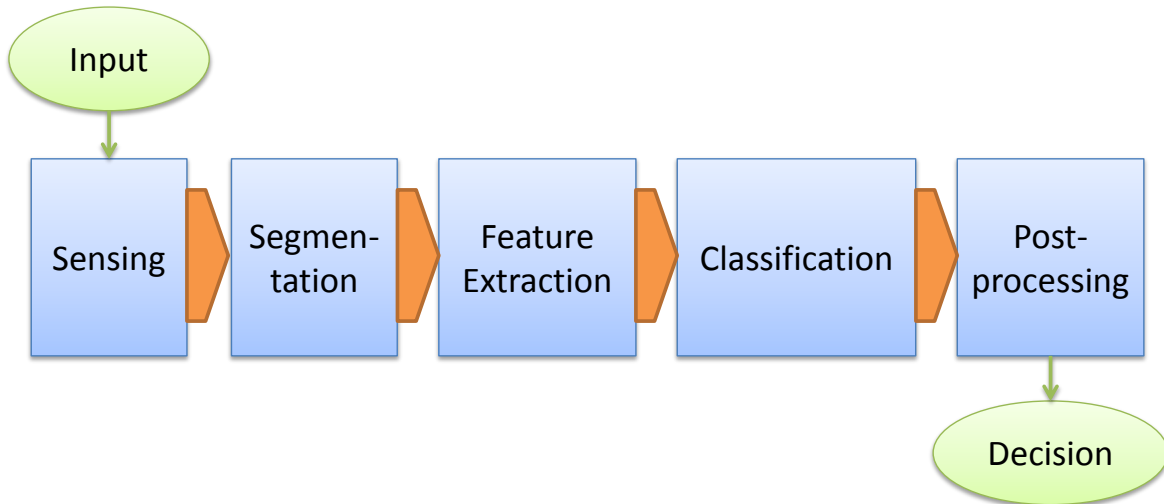


Figure 2.12: The basic components of a pattern recognition system as by Duda et al. (2000, p. 10), also called the classification pipeline. Even though the name and the arrows indicate an uni-directional data flow, it is also possible that systems utilize a feedback channel going in the reverse direction.

2.5.2 Sensing

Before data can be processed, it has to be made available to the system in some manner. This is called the *sensing* or *acquisition* step. It often involves recording from some kind of analog sensor, such as recording speech through a microphone, images through a camera, or gestures by using an acceleration sensor. Analog data is digitized in this phase. Mechanical pre-processing tasks, such as the resampling of audio data, can also be counted to the sensing phase. If the data to be classified is already available in digital form, the sensing phase is reduced to a copying of data.

2.5.3 Segmentation

In the *segmentation* phase, the raw data undergoes an initial selection and recognition procedure, which is also part of the pre-processing. There are two main aspects of segmentation, which are *filtering* and *grouping*. The goal of filtering is to remove noise that would otherwise interfere with and complicate the successive steps. Technically, this aspect of segmentation can indeed be a type of classification, usually a simple one, which distinguishes between *data* and *background* instead of the actual classes. For example, for the classification of facial expressions, the segmentation consists of a basic detection of the area of the face and removal of the background. In speech processing, a common segmentation task is the selection of segments (frames) which contain actual speech data. Grouping refers to the separation of samples, either in temporal or spatial manner. For instance, a classifier for shapes in images might get an input image with multiple objects in a single picture which it has to split up

(spatial case). In speaker recognition, we want to detect if two utterances belong to the same person (temporal case). Segmentation can also be used to detect overlapping samples.

Duda et al. (2000) note that segmentation can actually be a quite difficult problem, since classification can depend on it, but also vice versa at the same time. They also recognize one aspect of grouping that is the choice of the appropriate group size, or the problem of *subsets and supersets (mereology)*. For example, in speech recognition, it is all but clear to decide when a complete word is spoken – the sounds that make up the word “weaken” could equally be segmented into the two parts “we” and “can”. Confirming this already requires a considerable amount of feature and model knowledge.

It should be noted that many pattern classification systems do not care about segmentation and the potential issues arising from it; they rather circumvent the issue by setting the system specifications to require pre-segmented input data. For example, most Speaker Classification systems require the boundaries of an utterance to be specified externally.

2.5.4 Feature Extraction

Once the pre-processed, but still “raw” data of a single sample is available, it can be converted into something more meaningful. At the end, data should be available in the structured form of a feature vector. Feature extraction is all about transforming data from one format into another, thereby adding abstraction and itemizing information. It would be obsolete if the raw data already had this form, which may even be true for some very simple classification problems. However, this is mostly not the case. For example, an image that is stored as a matrix of color values, while technically also a feature vector, is not compatible with most classification algorithms. (In addition, the vector would have a varying size, which is also unsupported by most algorithms.) Instead, the actual feature vector might be based on particular regions of interest in the image, such as the Haar features commonly used for face detection, which are either present or not. The vector would then be a sequence of binary values, e.g. 1-0-0-0-1-1-0 (Haar features 1, 5, and 6 present). Another example is text classification, e.g. document types. While the source data is the document text, the feature vector would rather be a list of word or word combination frequencies. In Speaker Classification, various algorithms can be applied to the speech signal to extract numeric features such as pitch or noise levels.

As Duda et al. (2000) argue, the distinction between feature extraction and the subsequent classification step is not clearly defined. It is rather a practical design concept, which roughly states that feature extraction works on the raw data and classification outputs a class decision. Computing a feature may involve as much classification work as the next step, or even more. In an extreme scenario, it would be possible to put all the classification intelligence into the feature extraction and create a feature vector consisting of single integer feature “class index” that denotes the output class. The classifier would have a trivial job to do. Or conversely, we could have our black-box classifier work on the raw data and generate all intermediate feature representations internally. In these examples, it would even be difficult to decide whether the

label *feature extractor* or *classifier* was the appropriate one. Or, as Duda et al. (2000) state it, “an ideal feature extractor would yield a representation that makes the job of the classifier trivial; conversely, an omnipotent classifier would not need the help of a sophisticated feature extractor”. For the reasons mentioned earlier, it is definitely a good idea to have two distinct steps instead of a single monolithic one: It adds transparency and allows us to check the vector of actually relevant features, and possibly add further filtering steps, before turning to the next stage. Finally, feature extraction and classification algorithms usually originate from different algorithm libraries, with feature extraction often being more domain-specific, and the given architecture allows to connect them more easily.

The choice of the features best suitable for discrimination is possibly the most critical task in designing a pattern recognition system, and it is a manual task at its core. There is a strong connection between features and classifiers – some classifiers do not work well with certain features, correlation properties, large or sparse feature vectors etc. For instance, a single feature always containing the correct class index could still perform miserably when used with a classifier that cannot model the corresponding decision boundary. As a rough guideline though, several simple and intuitive feature are usually a better choice than fewer complex or encoded values.

Although a number of feature selection aspects can be automated to some extent, e.g. the restriction to a particular subset of good features or even the generation of derived features, as is the intention of *Deep Learning* (Bengio, 2009), the real breakthroughs in performance are usually achieved by a level of abstract intelligence that machines currently are not able to provide. Some of the traditional feature selection algorithms are *principal component analysis* (PCA), *factor analysis*, *nuisance attribute projection* (NAP), *hill climbing* and *random multinomial logit* (RMNL).

2.5.5 Classification

In the next step, a feature vector is processed by an algorithm that outputs the class to which the input data has the closest resemblance. In most cases, this decision is subject to uncertainty and is only accurate to a certain degree. Therefore, some classifiers return a confidence value or other type of score in addition to the class decision.

The choice of the appropriate classification algorithm is another central question of the pattern recognition system as a whole. There are actually numerous criteria which can play a role in this choice, some of which are the following:

- **Decision Boundaries:** The ability to model a certain type or “shape” of decision boundaries is one of the main parameters that influences the classification performance. If the decision problem has a different form than the classifier can model, then a satisfying performance can be reached neither on the training samples, nor on real data. This property only refers to the theoretical capability of the algorithm to model a problem, it does not indicate whether appropriate boundaries will actually be chosen.

- **Generalization Behavior:** Each classifier uses a different strategy to transfer the training samples into a model. It usually makes some assumptions about the feature ranges that have not been observed, such as by interpolating. For instance, a GMM with a single mixture assumes a normal distribution of values, hence it would work best for normally distributed values. The generalization behavior describes the ability of the algorithm to choose optimal boundaries for a given training set considering the underlying mathematical restrictions. This cannot be confirmed on the training set, because a classifier that learns the feature vectors from training by heart would achieve a flawless performance during training, but most likely not on other data.
- **Runtime/Complexity:** For many applications, the speech with which a sample is classified also plays an important role. To ensure scalability for a large number of samples or features, the computational complexity of the algorithm has to be taken into account in addition to benchmarks using actual data in a real setting. For example, the kNN algorithm has linear complexity with respect to the number of training samples.
- **Training Duration:** Although the fact, that training has to be performed only once in an off-line configuration, may seem to be a secondary aspect at first sight, it could also prove to be of critical importance under some circumstances. For a large volume of data, a single training phase can easily take days or weeks and therefore become an exclusion criterion for certain algorithms. In addition, especially for modern CPUs, the ability to take advantage of parallel processing can make one algorithm the preferred choice over another.
- **Incremental Training:** Some scenarios require the ability of a model to be extended dynamically, i.e. to perform on-line training. Not all algorithms can fulfill this requirement.
- **Model Size (Memory):** The data structure used to store the models can occupy a certain amount of space in working memory and on disk, which depends on the algorithm and possibly on the number of features and training samples. For applications running pattern recognition on embedded devices with limited resources, this is a decisive factor. Typically, the size can be traded against performance for many models, such as by introducing additional pruning for decision tree algorithms.
- **Feature Compatibility:** As outlined earlier, features are not generally compatible with all algorithms. For instance, most SVMs kernels work only well with real-numbered features.
- **Treatment of missing values:** In some cases, one or more features cannot be computed, and the value is said to be *missing* in the feature vector, which is different from being zero or some other default value. For example, pitch cannot be computed for

unvoiced segments. Algorithms (or sometimes implementations) tend to treat missing values differently: some can handle them better while others will show decreased performance. A few algorithms are even optimized for sparse feature vectors.

Duda et al. (2000, p. 7) distinguishes between three types of pattern recognition: statistical, neural, and syntactical. *Statistical pattern recognition* observes what patterns a set of features produced in the past, and based on this information, attempts to make probabilistic predictions for future samples under the assumption that they may be impacted by noise. This is the most relevant type of pattern recognition for Speaker Classification. *Neural pattern recognition* originates from a science even older than machine learning, which is the understanding and artificial reconstruction of processes in the brain and nervous system. Not surprisingly, its main representative algorithm are ANNs. It can also be considered a close descendant of statistical pattern recognition (see ibd.). The third category, *syntactical* or *structural pattern recognition*, is based less on approximate relationships, but rather on exact logical rules or grammars. Methods of this kind usually work with a much smaller, discrete feature space (i.e. using only nominal features), but can therefore infer more complex relations. An example would be the prediction of system failures in trains based on various possible system states.

Section 2.5.1 introduced the difference between generative and discriminative classifiers. A further distinction that can be made for discriminative methods is between *binary classifiers* or *dichotomizers*, which support exactly two classes, and *multi-label classifiers*, which support any number of classes. SVMs are natively binary classifiers. The reason for having binary classifiers usually lies in restrictions of the algorithm that is used, as a multi-label classifier represents a superset in terms of functionality. Yet, binary classifiers are the better choice for a detection task, since they allow the independent evaluation of single targets, which is not possible with an opaque multi-label classifier (unless it gives additional insight by providing scores for each class as part of the output). The multi-label classifier, on the other hand, resembles the identification task more closely. It is always possible to “simulate” a multi-label classifier by arranging multiple binary classifiers in the appropriate way. Two popular strategies are *one-against-one* and *one-against-the-rest*. One-against-the-rest is the simpler strategy (see Figure 2.13 B): For n classes, n binary models are trained. The samples of the first class correspond to the target samples and the second class to the non-target samples, i.e. the “rest”. For an unknown sample, each model would compute a score for its target class. The multi-label result is the model that produces the highest score (or a similar metric). The same technique can be used to create a discriminative wrapper around generative (i.e. single-class) models. In the one-against-one method (see Figure 2.13 C), $\sum_{i=1}^{n-1} i$ binary classifiers are generated, each discriminating a different pair of classes. For a new training sample, a majority voting is performed on the number of “duels” a class is able to win. While both methods work well for many tasks, a “true” multi-label classifier can potentially achieve a better performance since it has more extensive knowledge about the problem, while the two aforementioned approaches treat the model as a “black box”.

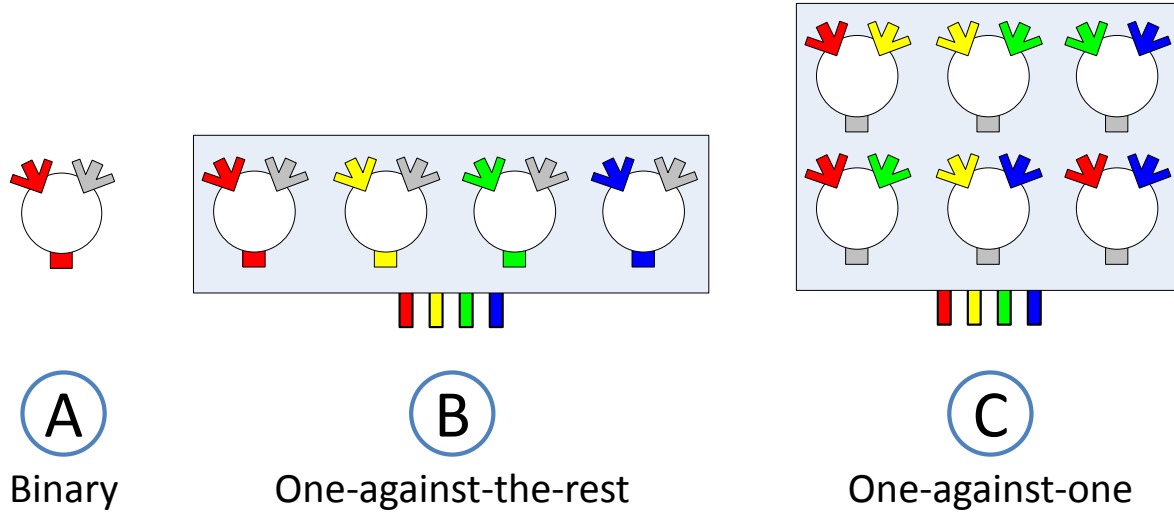


Figure 2.13: Two methods for creating multi-label classifiers from dichotomizers. Circles represent binary classifiers, the two incoming arrows are the labels they are trained with, and the box at the bottom end denote pro-class decisions. This examples uses four classes, with each color representing a different class.

The following sub-sections introduce the learning algorithms applied in the remainder of this thesis, in addition to a short overview on other popular methods at the end.

Gaussian Mixture Models

The *Gaussian Mixture Model* (GMM) is a straight application of Bayesian decision theory using a specific probability density function, the Gaussian density. It is a probabilistic method since the model is defined by the probability distribution of features, as opposed to a discrimination function. One of the earliest utilizations of these generative models for speaker recognition was done by Reynolds and Rose (1995). The overview in this section is partially based on their descriptions, as well as on Duda et al. (2000) and Reynolds, Quatieri, and Dunn (2000).

Bayesian decision theory attempts to describe a decision problem as completely as possible using a probabilistic approach. Let a given classification problem be defined by a list of C classes and F features. The probability that a sample \vec{x} (a feature vector $\{x_1, x_2, \dots, x_f\}$) belongs to class ω_c is written as

$$p(\omega_c, \vec{x}) = P(\omega_c | \vec{x}) p(\vec{x}).$$

In this equation, $P(\omega_c | \vec{x})$ is the *posterior probability*, i.e. the probability of class ω_c given that feature vector \vec{x} was observed, and $p(\vec{x}) = \sum_{j=1}^C p(\vec{x} | \omega_j) P(\omega_j)$ is the evidence factor, which scales the probabilities so that they sum up to 1 across classes. ω_c refers to a world state and

can be read as *the event that the sample is of class ω_c* . Using Bayes' theorem⁶, $P(\omega_c|\vec{x})$ can be rewritten, resulting in

$$P(\omega_c|\vec{x})p(\vec{x}) = \frac{p(\vec{x}|\omega_c)P(\omega_c)}{p(\vec{x})} = p(\vec{x}|\omega_c)P(\omega_c),$$

where $P(\omega_c)$ is the prior probability for class ω_c (with all $P(\omega_c)$ summing up to one), and $p(\vec{x}|\omega_c)$ is the *likelihood* or *class-conditional probability density function*, i.e. the probability that feature vector \vec{x} is produced by samples of class ω_c . This last function is the core aspect of the model. Before we look at how GMMs implement this density, let us see how we arrive at a classification result \hat{c} (class index). One option is to simply choose the model with the highest probability, i.e.

$$\hat{c} = c_i, i \in \{1, \dots, C\} \mid p(\omega_i, \vec{x}) = \max_c(p(\omega_c, \vec{x}))$$

This however is rather rigid in terms of performance and does not allow custom cost metrics. Therefore, another approach is to use a *likelihood ratio*, which is given for the binary classification case ($C = 2$) as

$$\mathcal{L}(\vec{x}) = \frac{p(\vec{x}|\omega_1)P(\omega_1)}{p(\vec{x}|\omega_2)P(\omega_2)}$$

and a decision can be obtained by comparing this ratio to a pre-defined decision threshold Θ that represents the cost metric:

$$\hat{c} = \begin{cases} 1 & \text{if } \mathcal{L} \geq \Theta \\ 2 & \text{if } \mathcal{L} < \Theta \end{cases}$$

As pointed out by Reynolds et al. (2000), the *log likelihood ratio* Λ is a common alternative to the likelihood ratio \mathcal{L} , and it is defined (without priors) as:

$$\Lambda(\vec{x}) = \log p(\vec{x}|\omega_1) - \log p(\vec{x}|\omega_2).$$

In case of GMMs, which are generative models ($C = 1$), either likelihood ratios are actually not defined. Reynolds et al. (2000) suggests to use an additional background speaker model that represents the rejection of the hypothesis ω_1 as class ω_2 . This model should represent the whole feature space – it can also include the knowledge from ω_1 . Such a model is called a *universal background model* (UBM).

Returning to the probability density function, the GMM employs a basic function that – in case of a single feature ($F = 1$) – has the following form known as *univariate* Gaussian probability density (see Section 2.5.1):

$$p'(x) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

⁶ $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ if $P(B) \neq 0$

If we consider multiple features, the more complex *multivariate* density is used:

$$p'(\vec{x}) = \frac{1}{(2\pi)^{F/2} |\vec{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})' \vec{\Sigma}^{-1} (\vec{x} - \vec{\mu})\right),$$

where $\vec{\mu}$ is the mean vector, $\vec{\Sigma}$ is the covariance matrix, and $\vec{\Sigma}^{-1}$ is its inverse. This equation defines a single *mixture* or *Gaussian*. Instead of a full covariance matrix, only a diagonal matrix (i.e. the variances) is often used for efficiency reasons or because they actually perform better. It is also possible to use the same covariance matrix for all Gaussians or even for all speakers (see Reynolds & Rose, 1995). Using the above function, a single normal distribution can be modeled. By combining multiple mixtures into one model using a weighted sum, this can be extended to almost arbitrary distributions. The final density function of a GMM with G mixtures has the following form:

$$p(\vec{x}|\omega_c) = \sum_{i=1}^G w_i p'_i(\vec{x}),$$

where w_i are the Gaussian weights that sum up to 1, and p'_i is the multivariate density function for the i -th Gaussian.

In order to train a GMM, i.e. to learn the mixtures, *maximum-likelihood estimation* is used to determine the optimal parameters of the likelihood function. This method is not specific to GMMs, but can be used with any classifier with a parameterized likelihood function. Let $p(\vec{x}|\theta)$ be the likelihood function for a parameter configuration θ . As suggested by Duda et al. (2000), we can write the likelihood that a parameter configuration θ produces on set of N independent training instances $\mathcal{D} = \{\vec{x}_1, \dots, \vec{x}_N\}$ as

$$p(\mathcal{D}|\theta) = \prod_{k=1}^N p(\vec{x}_k|\theta).$$

Based on this, the optimal parameter configuration $\hat{\theta}$ is the one that maximizes the log likelihood⁷, i.e.

$$\hat{\theta} = \arg \max_{\theta} (\log p(\mathcal{D}|\theta)).$$

Some types of functions can be more easily maximized than others. We can for example show (Duda et al., 2000, p. 89) that a model consisting of a single mixture and parameter vector $\theta = (\vec{\mu}, \vec{\Sigma})$ can be maximized as

$$\hat{\theta} = \left(\hat{\mu} = \frac{1}{N} \sum_{k=1}^N \vec{x}_k, \hat{\Sigma} = \sum_{k=1}^N (\vec{x}_k - \hat{\mu})(\vec{x}_k - \hat{\mu})^\top \right)$$

However, a direct maximization of multiple mixtures is not possible due to the non-linearity of the expression (see Reynolds & Rose, 1995). Instead, the iterative *expectation maximization*

⁷the logarithm guarantees monotony

(EM) algorithm (Dempster, Laird, & Rubin, 1977) can be applied to find a good approximation. A quality of the algorithm is that it has a good convergence on the optimal values. Let $\theta = \bigcup_{i=1}^G (w_i, \vec{\mu}_i, \vec{\Sigma}_i)$ be the mixture parameters (including the weights) of the GMM. Then the EM algorithm performs the following steps:

1. Determine an initial configuration $\theta = \theta_0$.
2. Compute the (log) likelihood of the current configuration on the training set, i.e. $p(\mathcal{D}|\theta)$ (*Expectation* step).
3. Estimate new parameters θ' such that $p(\mathcal{D}|\theta') \geq p(\mathcal{D}|\theta)$ (*Maximization* step).
4. Set $\theta = \theta'$, and continue with step 2 until convergence or a predefined number of iterations is reached.

For GMMs, Reynolds and Rose (1995) use the following formulas to derive the new parameter estimates for the i -th Gaussian in the maximization step:

$$\begin{aligned} w'_i &= \frac{1}{N} \sum_{k=1}^N p(i|\vec{x}_k, \theta) \\ \vec{\mu}'_i &= \frac{\sum_{k=1}^N p(i|\vec{x}_k, \theta) \vec{x}_k}{\sum_{k=1}^N p(i|\vec{x}_k, \theta)} \\ \vec{\Sigma}'_i &= \frac{\sum_{k=1}^N p(i|\vec{x}_k, \theta) [\vec{x}_k - \vec{\mu}_i][\vec{x}_k - \vec{\mu}_i]^\top}{\sum_{k=1}^N p(i|\vec{x}_k, \theta)} \end{aligned}$$

The posterior probability for a single Gaussian i can be derived from the GMM likelihood given earlier:

$$p(i|\vec{x}, \theta) = \frac{w_i p'(\vec{x}|\theta)}{p(\vec{x}|\theta)} = \frac{w_i p'(\vec{x}|\theta)}{\sum_{j=1}^G w_j p'_j(\vec{x}|\theta)}$$

Reynolds et al. (2000) suggest that five EM iterations are usually sufficient for speaker recognition. The number of Gaussians G is not learned by the algorithm and must be specified manually by consideration of the problem or through experimentation. Figure 2.14 illustrates the EM-based learning process using an example with only a single feature⁸. The upper image shows a histogram of the actual feature distribution on 1000 samples. The next image shows the normal probability density using the mean and variance of the samples, which corresponds to a GMM with a single mixture. As can be seen, the value distribution is not approximated very well with this single mixture. The following images show a GMM trained using 15 mixtures and varying numbers of EM iterations. Each of the last images plots the individual Gaussians, as well as the combined, weighted distribution. The sequence shows how the GMM slowly adapts to the actual shape, which bears great resemblance to the combined function in the last image.

⁸The sample data depicts the power consumption of a running washing machine appliance in Watts.

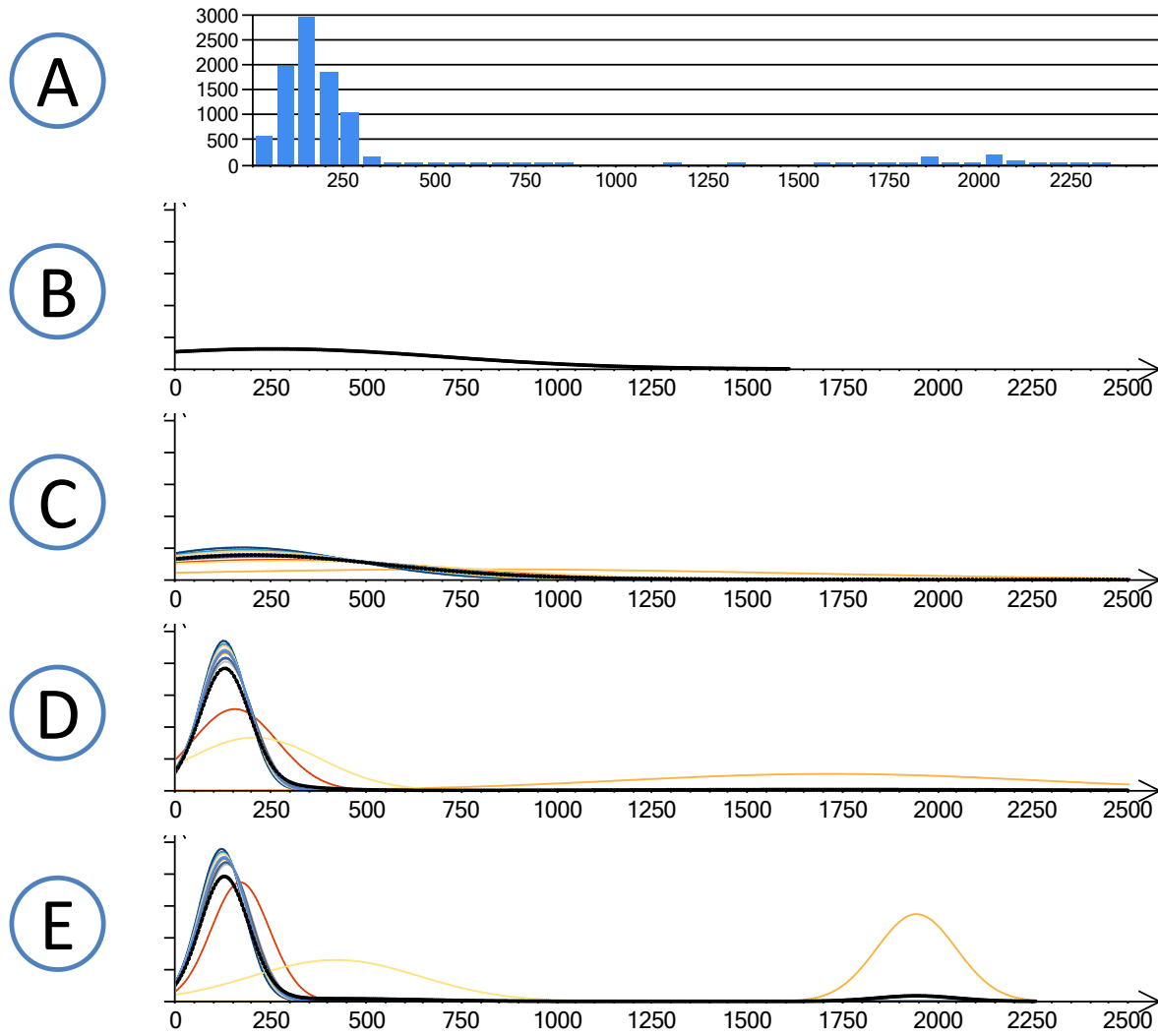


Figure 2.14: Example of expectation maximization (EM) algorithm application to sample data. **A**: Histogram of actual values (X axis = value, Y axis = number of instances). **B**: Mean and standard deviation of the samples displayed as a single univariate Gaussian density function. **C**: 15 Mixtures in a GMM trained on the data using one iteration of EM. The black curve is their weighted sum. **D**: The same GMM after 3 iterations of EM. **E**: The GMM after 7 EM iterations.

A question that remains is how to perform the initialization of the GMM, i.e. the choice of mixtures for the first EM step. Initializing all mixtures with a static configuration $(w, \vec{\mu}, \vec{\Sigma})$ is possible, but usually does not work very well. Instead, a common approach is to set $w = 1/G$, use randomly selected training samples from \mathcal{D} to compose $\vec{\mu}$, and set $\vec{\Sigma}$ to the actual covariances of all training samples. This usually provides quite good performance already, but for specific problems, more systematic methods may be preferred. One such method, which is also applied in this thesis, is the *K-means* algorithm. K-means is a *clustering* algorithm. It attempts to partition any number of instances into K clusters, which are characterized by their means, or *centroids*. It also works iteratively in a way quite similar to EM. It consists of the following steps:

1. *Initialization:* Assign the first K samples to the K -th cluster. The remaining samples are assigned to the cluster with the closest distance.
2. *Recomputation:* Compute the new centroids for each cluster based on the most recent assignments.
3. *Reassignment:* Each sample is re-assigned to the cluster with the closest centroid. A feature-compatible distance metric such as the mean Euclidean distance can be used. If the number of sample-cluster associations which have changed in this step is zero, the final configuration is reached. Otherwise, continue with step 2. Breaking is also possible when a predefined number of iterations is reached.

When K-means is used to initialize a GMM, the number of clusters K is chosen identical to the number of mixtures G . Then, the mixture weights can be set to the percentage of instances assigned to a cluster, means are identical to the centroids, and the variance can also be computed from the samples assigned to the cluster.

We have previously seen how we can arrive at a class decision by comparing (log) likelihoods directly, or by additionally subtracting the likelihood of a UBM. There is also a third common approach: Instead of evaluating the UBM directly, the class-specific models are created by adapting samples to the UBM. Reynolds et al. (2000) have used a Bayesian adaptation technique that is often referred to as *maximum a posteriori (MAP) adaptation* to accomplish this. In practice, these adapted models, which are closer to the background features space, have proven to be superior in many scenarios. They do however require a large amount of background material. To perform MAP adaptation, a UBM is first trained on all material \mathcal{D} using EM. For any given class ω_c indicated by its class index $c \in \{1, \dots, C\}$, let \mathcal{D}_c denote the training samples of this class with $N_c = |\mathcal{D}_c|$. According to Reynolds et al. (2000), the new GMM is initialized with the UBM parameters θ , and the procedure is performed in the same way as in the EM algorithm, but with a modified maximization step. Also, typically only a single iteration is performed. The maximization uses the same posterior probabilities $p(i|\vec{x}, \theta)$ as before, but with the following estimation formulas for θ_c'' :

$$\begin{aligned}
w''_{c,i} &= \left[\frac{\alpha_i^w}{N_c} \sum_{k=1}^{N_c} p(i|\vec{x}_k, \theta) + (1 - \alpha_i^w)w_i \right] \gamma \\
\vec{\mu}''_{c,i} &= \alpha_i^m \frac{\sum_{k=1}^{N_c} p(i|\vec{x}_k, \theta) \vec{x}_k}{\sum_{k=1}^{N_c} p(i|\vec{x}_k, \theta)} + (1 - \alpha_i^m) \vec{\mu}_i \\
\vec{\Sigma}''_{c,i} &= \alpha_i^v \frac{\sum_{k=1}^{N_c} p(i|\vec{x}_k, \theta) [\vec{x}_k - \vec{\mu}_i][\vec{x}_k - \vec{\mu}_i]^\top}{\sum_{k=1}^{N_c} p(i|\vec{x}_k, \theta)} + (1 - \alpha_i^v)(\vec{\Sigma}_i + \vec{\mu}_i^2) - (\vec{\mu}''_{c,i})^2
\end{aligned}$$

where γ denotes a scaling factor that simply normalizes all weights to sum up to 1. Each of these formulas estimates the corresponding parameter partially on new training data and partially on UBM data. The balance is controlled by the separate adaptation coefficients α_i^w , α_i^m , and α_i^v for weights, means, and variances, respectively. From here, we will focus only on the adaptation of the means, since only they are relevant for the following chapters of this thesis. The data-dependent coefficient is defined as

$$\alpha_i^m = \frac{\sum_{k=1}^{N_c} p(i|\vec{x}_k, \theta)}{\sum_{k=1}^{N_c} p(i|\vec{x}_k, \theta) + r},$$

where r is a static parameter called the *relevance factor*. It is often set to a default value of 16, but it can be changed to control whether the UBM or the class-specific data is emphasized. In addition, this equation expresses that the coefficient will automatically fall back to the background model if only little new training data is provided, which is particularly helpful for cases of insufficient data.

GMMs are often favored due to their good ratio of simplicity versus effectiveness. They are faster than many other methods, and can be flexibly adjusted to more complex problems by adding Gaussians. Moreover, a large number of small and independent random noise effects that occur to a set of samples form a Gaussian distribution (*Central Limit Theorem*, see Duda et al., 2000, p. 33), which can be considered as an analogy to how the variation in samples is actually produced.

Support Vector Machines

Support Vector Machines (SVMs) tackle a classification problem quite differently to GMMs. The first and most obvious difference is that the SVM is a discriminative approach that models exactly two classes instead of one. SVMs belong to the family of *linear discriminant functions*, although they are not limited to linear decision boundaries, as we will see later. Discriminant functions differ from probabilistic methods such as GMMs in that they do not make an assumption about the probabilities that produce the feature space, but rather about how the classes can be separated. Linear discriminant functions are linear either to the features or functions based on them. They often show a good trade-off between simplicity, generalization, and performance (Duda et al., 2000, p. 215).

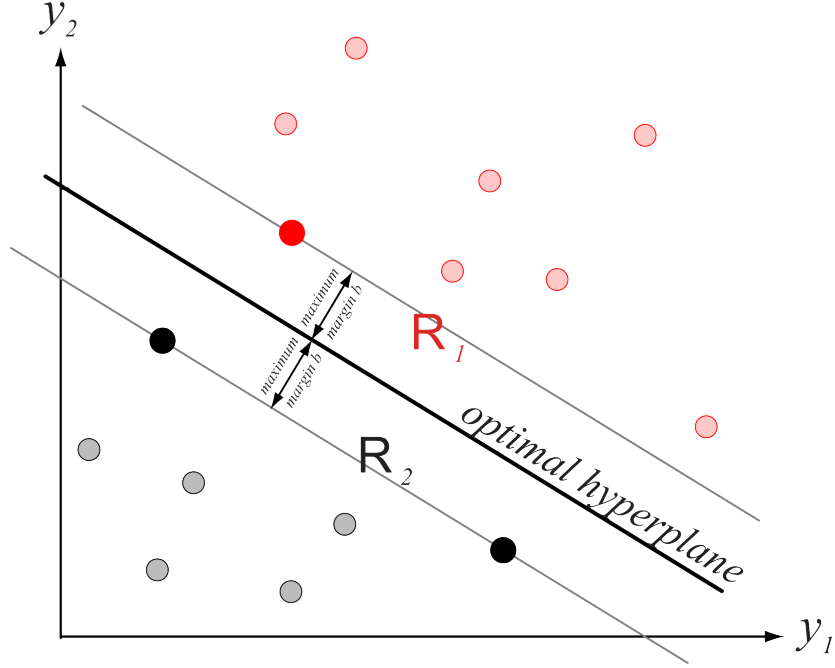


Figure 2.15: Training examples of two classes from which the optimal support vectors (full circles) are derived. The distance of the hyperplane from these samples (the margin) is equal for each. (Source: Duda et al., 2000, p. 262)

The linear discriminant function is typically written as

$$g(\vec{x}) = w_0 + \vec{w}^\top \vec{x} = w_0 + \sum_{i=1}^d w_i x_i,$$

where \vec{x} is a sample consisting of F features (like in the previous section), \vec{w} is an F -dimensional vector of weights, and w_0 is a *decision threshold*. This function is linear in \vec{x} . In contrast to this, the general discriminant function describes an expansion of \vec{x} into a different feature space of dimension \hat{d} using functions φ and is written as

$$g(\vec{x}) = \vec{a}^\top \vec{y} = \sum_{i=1}^{\hat{d}} a_i \varphi_i(\vec{x}),$$

where now \vec{a} is the \hat{d} -dimensional weights vector. Even this general function is considered linear with respect to the mapped space $\vec{y} = \varphi(\vec{x})$. Interpreted algebraically, $g(\vec{x}) = 0$ describes a plane in d or \hat{d} dimensional space with orientation \vec{w} or \vec{a} – a hyperplane – that separates two classes ω_1 and ω_2 . The plane can be said to divide the space into regions \mathcal{R}_1 and \mathcal{R}_2 , and $g(\vec{x})$ measures the distance from a sample to the plane. The closest distance from a training sample to the plane is also called *margin*.

SVMs are a special case of the generalized linear discriminant function. In contrast to other methods, which accept any hyperplane separating the feature space, the SVM attempts

to find the optimal plane in terms of margin size h . This is illustrated in Figure 2.15. The hyperplane is defined through the use of *support vectors* – training instances which represent particularly difficult patterns (i.e. resulting in a low likelihood) and which all have the same, smallest possible distance from the hyperplane. Training of a SVM is complicated by the fact that finding the support vectors requires a steady evaluation of the current model to determine this likelihood. The following distance condition is satisfied by such a hyperplane according to Duda et al. (2000):

$$\frac{z_k g(\vec{y}_k)}{\|\vec{a}\|} \geq b,$$

where $k \in \{1, \dots, N\}$ refers to a training sample, $z_k := \pm 1$ depending on the class index of \vec{x}_k , and b is a positive margin. Thus, the goal is to maximize b by finding the appropriate vector \vec{a} . By rearranging, we can derive the function L , which maximizes the margin and minimizes the training error at the same time:

$$\min L(\vec{a}, \vec{\alpha}) = \frac{1}{2} \|\vec{a}\|^2 - \sum k = 1^N \alpha_k [z_k \vec{a}^\top \vec{y}_k - 1],$$

where α_k are so-called Lagrange multipliers. Further reorganization removes the dependency of this term on \vec{a} and transforms it to what is known as the *unconstrained dual form*:

$$\max L(\vec{\alpha}) = \sum_{k=1}^N \alpha_i - \frac{1}{2} \sum_{k,j} \alpha_k \alpha_j z_k z_j \vec{y}_j^\top \vec{y}_k.$$

A common way to solve this equation is by using quadratic programming (QP).

Choosing the functions φ is an important aspect of SVM design. Many discrimination problems cannot be solved by a linear function in the original feature space. This function transforms the problem into a higher dimensional space where this may be possible. In fact for all dichotomization problems there exists such a mapping (Duda et al., 2000, p. 259). Figure 2.16 shows a simple example where the transformation from one to two dimensions allows to solve a problem in a linear way. In practice, the new feature space mostly has a considerably larger number of features. The mapping φ is usually described in terms of a single *kernel function*, which is defined as $K(\vec{x}_j, \vec{x}_k) = \vec{y}_j^\top \vec{y}_k$. This more specialized representation of the mapping is a computational optimization, as only the inner products are needed for SVM operation (as part of the above $L(\vec{\alpha})$ maximization term).

The most widely used kernel functions and their parameters are:

- Linear kernel: $K(\vec{x}_j, \vec{x}_k) = \vec{x}_j^\top \vec{x}_k$
- Polynomial kernel: $K(\vec{x}_j, \vec{x}_k) = (\vec{x}_j \cdot \vec{x}_k + a)^d$
- Radial basis function (RBF, Gaussian) kernel: $K(\vec{x}_j, \vec{x}_k) = \exp(-\gamma \|\vec{x}_j - \vec{x}_k\|^2)$

Other Machine Learning Methods

The previous sections have introduced the classification algorithms most relevant to this thesis. This section should give a brief overview of some alternate methods.

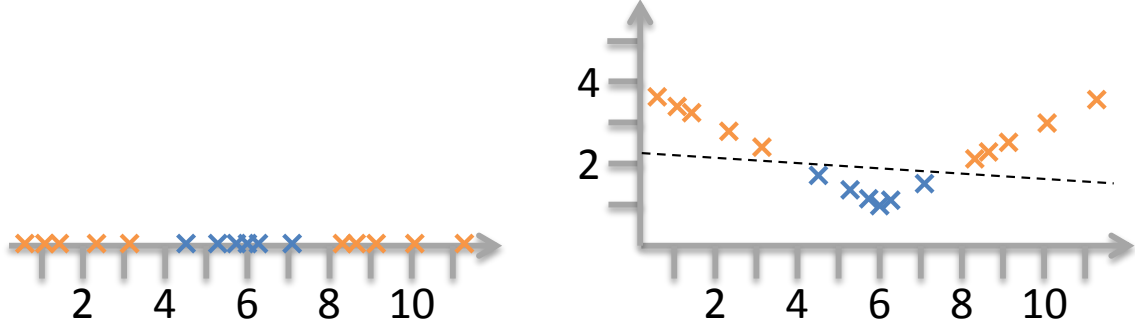


Figure 2.16: Feature space transformation as a means to simplify classification problems. In the left image (one dimension), no single threshold (which corresponds to a linear function) can separate the two classes *blue* and *orange*. In the right image, the addition of a feature $y = |\frac{4}{7}(x - 6)| + 1$ allows a simple linear separation of the classes (dashed line).

Naive Bayes. Naives Bayes is technically not a classification algorithm, but rather a general probabilistic model that includes several premises. It is based on the same Bayesian decision theory as the GMM. There are two main differences: (1) The model is not bound to a specific probability distribution. Choice of the likelihood function $p(\vec{x}|\omega_c)$ is up to the implementation, although a simple normal distribution is quite common in practice. (2) The model assumes the statistical independence of features, hence it is “naive”. For example, in face recognition, the size of eyes, nose, and mouth could be considered independent from each other – even though it may not be true. Because probability theory allows the individual properties to be multiplied in this case, it simplifies the class model to

$$p(\omega_c, \vec{x}) = \prod_{i=1}^F p(x_i|\omega_c)P(\omega_c)$$

and the likelihood for all features to

$$p(\vec{x}|\omega_c) = \frac{\prod_{i=1}^F p(x_i|\omega_c)P(\omega_c)}{p(\vec{x})},$$

where $p(\vec{x})$ is simply an evidence scaling factor. An advantage of this method is its simplicity, which makes it fast to compute and prone to overfitting. On the downside, the method is not capable of modeling complex relationships.

Multilayer Perceptron Networks. Multilayer perceptron networks (MLP), the most common type of *artificial neural networks* (ANN), can be considered an extension of the basic logic that was motivated for SVMs. As detailed in that section, the linear discriminant function used other functions φ to map the input features into a higher-dimensional space to allow nonlinear problems to be solved. While SVMs depend on a manual choice of the suitable

kernel function – an experimental exploration of possible functions would create too large a parameter space –, ANNs try to determine the degree of nonlinearity automatically (Duda et al., 2000, p. 283). This may yield better results, but can also lead to arbitrariness, as the mapping process is no longer transparent. The motivation of ANNs can be found in neurobiology: In the central nervous system, multiple neurons are connected via synapses. When the sum of the inputs (dendrites) surpasses some threshold, the output (axon) is activated and sends an electric signal. ANNs in artificial intelligence attempt to recreate this biological process in software. Technically, an MLP can be sketched as a multi-layered interconnected graph with nodes in each layer: the input layer, the output layer, and one or more hidden layers. An example is shown in Figure 2.17. We will limit the formalization to networks with three layers, since adding more layers does not add expressiveness (although it can optimize the learning process). In fact, the three-layered network can be used to model any possible decision boundary (Duda et al., 2000, p. 287). The number of nodes in the input layer corresponds to the feature vector size F , the number of nodes in the output is equal to the number of classes C , and the number of hidden units n_H is chosen according to the difficulty of the problem. Each node in the MLP computes a value called *net function*, denoted simply *net*, which is a weighted sum of all inputs into the node from the previous layer, and which is transformed by another function called *activation function* f . Formally, these functions can be specified as (see Duda et al., 2000, p. 285):

$$\text{Hidden layer: } y_j = f(\text{net}_j(\vec{x})) = f(\vec{w}_j^\top \vec{x}) = f\left(\sum_{i=0}^F x_i w_{ji}\right)$$

$$\text{Output layer: } z_c = f(\text{net}_c(\vec{x})) = f(\vec{w}_c^\top \vec{x}) = f\left(\sum_{j=0}^{n_H} y_j w_{cj}\right)$$

$$= f\left(\sum_{j=1}^{n_H} w_{cj} f\left(\sum_{i=0}^F w_{ji} x_i + w_{j0}\right) + w_{c0}\right) = g_c(\vec{x}),$$

where $j = 1, \dots, n_H$ is the index of a hidden unit and $c = 1, \dots, C$ is the index of an output node. As can be seen, ANNs are multi-label classifiers, other than SVMs. The function f is often chosen as the signum function $\text{sgn}(x) = \pm 1$ or a continuous *sigmoid* function. Note that the 0-th feature index, x_0 , is a formal shortcut to account for a decision threshold like in earlier descriptions. To train an ANN, the backpropagation algorithm can be used (see Duda et al., 2000, p. 288). Their slow training is generally considered one of the disadvantages of ANNs.

Decision Trees. Decision trees model a sequence of sub-decisions, often of binary kind (two alternates per decision). Each item in the sequence depends on the answer chosen for the previous sub-decision. At the end of the sequence, an output (i.e. class) decision is generated. The more general problem is referred to as *classification and regression trees* (CART, see Duda et al., 2000, p. 396). Decision trees can be represented as nested *if* statements with simple

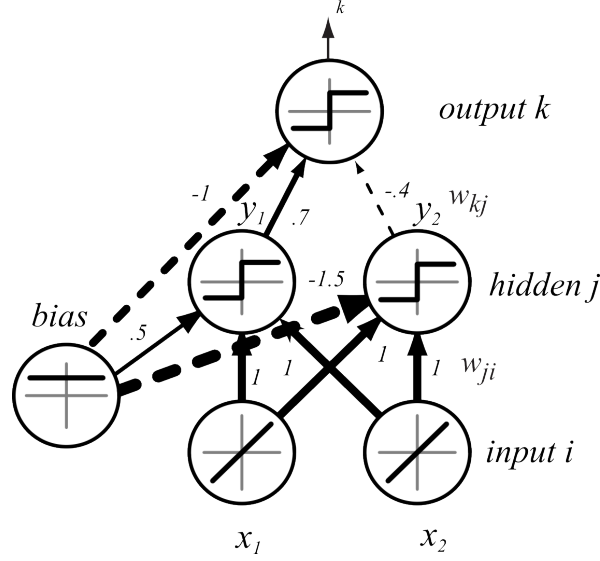


Figure 2.17: Example of a multilayer perceptron network with a single hidden layer. The *bias* node represents the static decision threshold. Solid lines indicate excitatory weights, dashed lines inhibitory, and the line width indicates the amount of the weight, which is also noted next to each arrow. (Source: Duda et al., 2000, p. 284)

comparisons. As this is a primary construct in all programming languages, their runtime performance is usually very good. They can also be visualized in tree shape as in Figure 2.18 because there is exactly one root and cycles are not allowed. The features in the tree can be real numbers, but decision trees are one of the algorithms that also work very well with nominal features. Finally, their compact memory footprint is another advantage of these models.

Nearest Neighbor. k -Nearest-Neighbor (kNN) is a straightforward classification algorithm. Like SVMs, the algorithm does not assume any probability distribution, but rather models the discrimination function (Duda et al., 2000, p. 161). Different to the former, kNN is natively able to discriminate between multiple classes. The nearest-neighbor rule is based on a distance metric d , where $d(\vec{x}_1, \vec{x}_2)$ computes the distance between two samples. For numerical features in F -dimensional space, the Euclidean distance is the most common metric used. In the simple case of $k = 1$, an unlabeled sample \vec{x} is assigned to the same class as the closest training sample $\vec{x}_i \in \mathcal{D}_c = \{\vec{x}_1, \dots, \vec{x}_N\}$ and class indices c_1, \dots, c_N :

$$\hat{c} = c_i \mid i = \min_i d(\vec{x}, \vec{x}_i), \quad i = 1, \dots, N$$

Since this may easily become inaccurate for large numbers of training samples, a majority voting is often performed on the k closest samples according to the metric. The default kNN does not perform an actual training, it merely stores the training vectors, which is very fast.

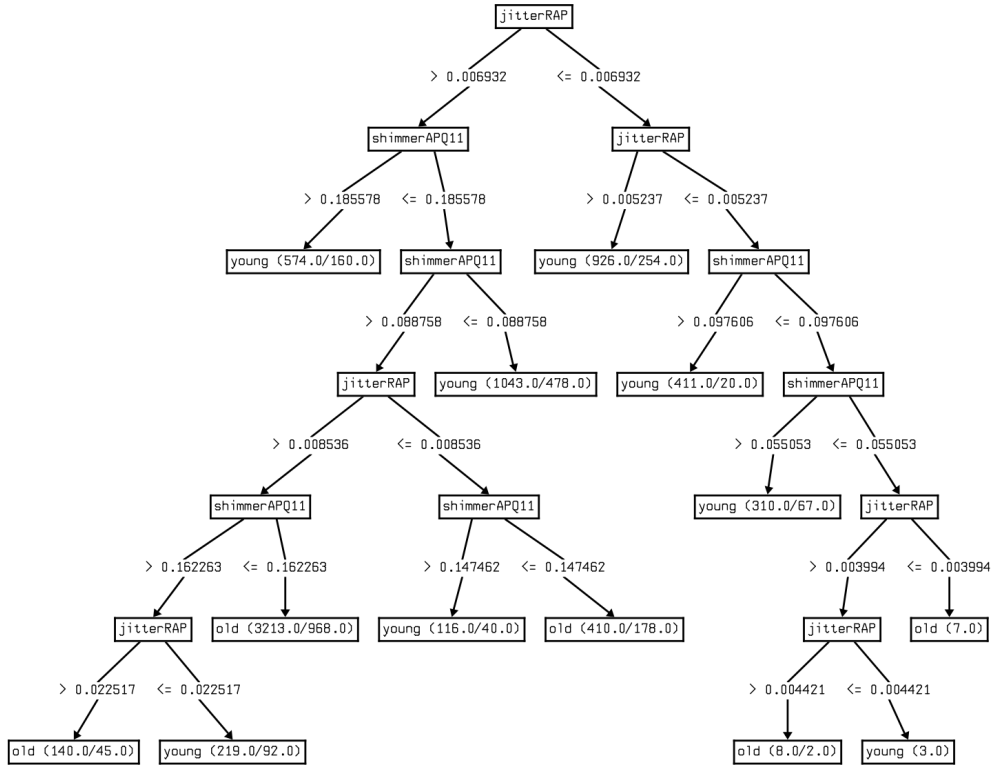


Figure 2.18: Visualization of a C4.5 decision tree modeling a simple two-class speaker age classifier based on acoustic features.

However, the runtime performance is $O(N)$ and can be low for large training sample counts. Also, model memory consumption can be high. To improve the speed, the training instances of a single class can be combined to fewer *prototype* instances. kNN may be an appropriate choice if the distance metric closely resembles the problem, e.g. a geometric classification problem. Otherwise, the algorithm is well-known rather because of its simplicity than for its classification performance.

Markov Models. All of the previously presented classification methods have no concept of time. Each sample models only a single instant. This can be a problem when a sample actually consists of a series of dependent frames, such as a gesture or a spoken word. Obviously, these can only be recognized as a whole. There are some techniques how the algorithms can still be used, such as by concatenating all samples in a single feature vector. Because the vector has to be of a fixed size, this preserves some but not all aspects of time: either the data has to be truncated or it has to be compressed. A model that does not have these drawbacks is the Markov model, which is actually a family of models. In these models, each sample is represented by a series of states which can have arbitrary and dynamic length. For many processes, *Markov chains* can be used to describe the model. If the process has some states

which cannot be observed, a *hidden Markov model* (HMM) is used instead. Like GMMs, Markov models are generative: each model describes only a single class. A Markov chain is defined by states, transition probabilities and prior probabilities, formally $\theta = (Q, A, P)$. In this model, Q is a set of states $\{q_1, \dots, q_M\}$, A is a transition matrix with $A_{ij} = P(q_i|q_j)$, and P defines the initial probabilities, i.e. $P(q_1), \dots, P(q_M)$. Assuming that each state reflects a feature vector, let $X = (\vec{x}_1, \dots, \vec{x}_t)$ denote a sequence of states with $\forall i : \vec{x}_i \in Q$. The probability of each subsequent state depends only on the previous state (or on the k last states for k -th order Markov models), i.e. $P(\vec{x}_j|\vec{x}_{j-1}, \dots, \vec{x}_1) = P(\vec{x}_j|\vec{x}_{j-1})$. Markov models are associated with a number of recurring tasks, which are the *evaluation task*, the *decoding task*, and the *learning task*. Classification is based on the evaluation task, which computes the probability that a given sequence X was produced by a model as a simple product:

$$P(X|\theta) = P(\vec{x}_1) \prod_{i=2}^t A_{\vec{x}_{i-1}, \vec{x}_i}$$

To solve an actual classification problem, multiple models have to be evaluated with the same sequence and their values have to be compared using the known techniques. Training a Markov chain θ with a given set of states Q is straightforward, since only the transitions that occur in a given set of observed training sequences X_1, \dots, X_N need to be counted and normalized to obtain A and P . Learning states is more involved. For this information and for HMMs, see e.g. Duda et al. (2000, p. 128).

2.5.6 Post-Processing

Some pattern recognition systems end as soon as the output class has been determined by the classifier, but there are also cases where further steps become necessary before the result can be reported to the system. These steps are called *post-processing*. The most common post-processing task is a kind of *fusion*.

Some systems apply multiple classifiers in parallel to the same or related input data, possibly with completely different features and algorithms. Since they may report different classes, a final decision has to be found, i.e. the individual results need to be fused (*classifier fusion*). This process can be as simple as a majority voting (reporting back the class with most hits), but this would assume that all classifiers were equally good and the outcome was independent from the input. It is often possible to get a better result if another classifier is trained using the decisions of the main classifiers as feature vector. This classifier is said to run on a different layer, or be a *meta classifier*. It is also possible to add the original features to the meta classifier or to cascade into multiple meta layers. Another case where a meta classifier can be useful is when the main classifier is a simple multi-label wrapper around binary or generative models that report scores for each class. In this case, the meta classifier could be trained on the individual scores to return a better overall class decision. All these types of classifier fusion are also called *late fusion* or *score-level fusion*, in order to distinguish them from early or feature-level fusion. Müller (2005) also calls them the *static aspect* of fusion.

A further optional post process is the *temporal fusion* of results. In several scenarios, multiple instances are known to belong to the same entity. For example, when a speaker utters multiple sentences, these may be forwarded to the system as separate samples to allow initial results to be obtained as early as possible, or to limit the maximum size of a single sample. Some applications also use a fixed frame size and process each frame as a separate sample. In both cases, the results of multiple samples have to be merged into a single result. This aspect is particularly important for incremental systems. An implementation might consist of a temporal smoothing function. Another popular method for merging multiple time slices are *dynamic Bayesian networks*, which have been used by Müller (2005) for temporal fusion of multiple utterances (therein called *dynamic aspect*). It is also possible that the cohesiveness between two samples is likely, but not for certain – e.g., the speaker could also change in the aforementioned example.

A third purpose of post-processing is the inclusion of *external knowledge*. This knowledge (also called *top-down knowledge*, *expert knowledge*, *domain knowledge*, or *contextual knowledge*) is independent from the input samples and therefore impossible for a system to learn automatically. It includes aspects such as the cost metric or cost of misclassification. For example, if we know that false positives are worse for one class than for another, we might shift the decision function slightly away from that class. A different aspect concerns the prior probability of classes. If the engineer has knowledge about the statistical frequency of classes in a given situation, she might want to configure prior probabilities to favor the class with the greater likelihood. For example, if we use age detection in a call center that provides a technical hot-line for agricultural engines, decreasing the prior probabilities of children and teenagers can probably improve the performance.

2.5.7 Evaluation

The word evaluation, standing by itself, is unfortunately quite an overloaded term in machine learning. In this work, the actual procedure involved is very similar for each context in which the term is used, which explains why they are collectively covered in this section. However, for clarification, it should be distinguished at this point between the different contexts and meanings in which the word is used in the remainder of this work:

System Evaluation A qualitative benchmarking of all components of a system in their entirety. The goal of this procedure is to get an idea of how well the system is performing.

Component Evaluation A qualitative benchmarking of the component’s performance by using an appropriate method. This can also be extended to multiple connected components.

Evaluation Data Set The global data set used to perform a so-called “one-shot” system evaluation.

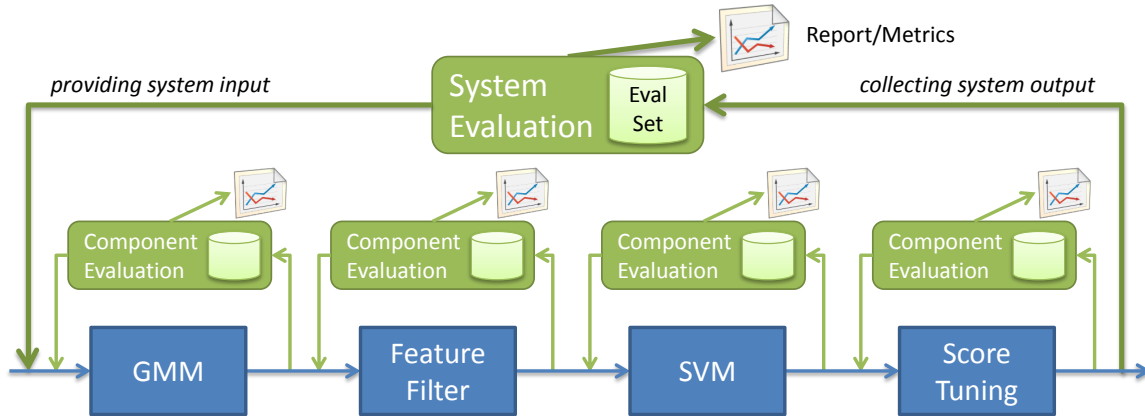


Figure 2.19: Scheme of system and component evaluation of an example system consisting of four components (blue squares). Each evaluation generates different measures and can be performed individually.

Evaluation Run A single execution of a system or component evaluation with given input data set.

Technical Evaluation Process This refers to the technical process of applying a component (usually a classifier) to some data set and then computing appropriate performance statistics on the result. This can be done as part of a system or component evaluation, but the output could also be used as input to another component.

General Procedure

A large subfield in machine learning science deals with the evaluation of systems. Ways to objectively measure the performance of systems are needed in order to rate the system, to compare the different parameter settings and design choices, and to compare it with other systems. As such, the evaluation is an integral part of the machine learning design process. The result of the evaluation process is a set of human-interpretable numbers, tables, and charts.

The general procedure for a whole system is to take a data set specifically created for the evaluation, hence called *evaluation data set* (or short *eval set*), and apply the machine learning components in a similar way as during creation of the system. This time, however, the labels are stripped from the data set and classifier training steps are replaced with classifier application steps, i.e. the missing labels are predicted, or removed (e.g. in case of feature selection). At the end, there is an additional step where the classifier predictions are compared with the true class labels.

Each system or component evaluation has two associated points that define its scope: A data input point and a results measurement point. In the classifier evaluations considered here, the data at the results measurement point corresponds to the input data with an added

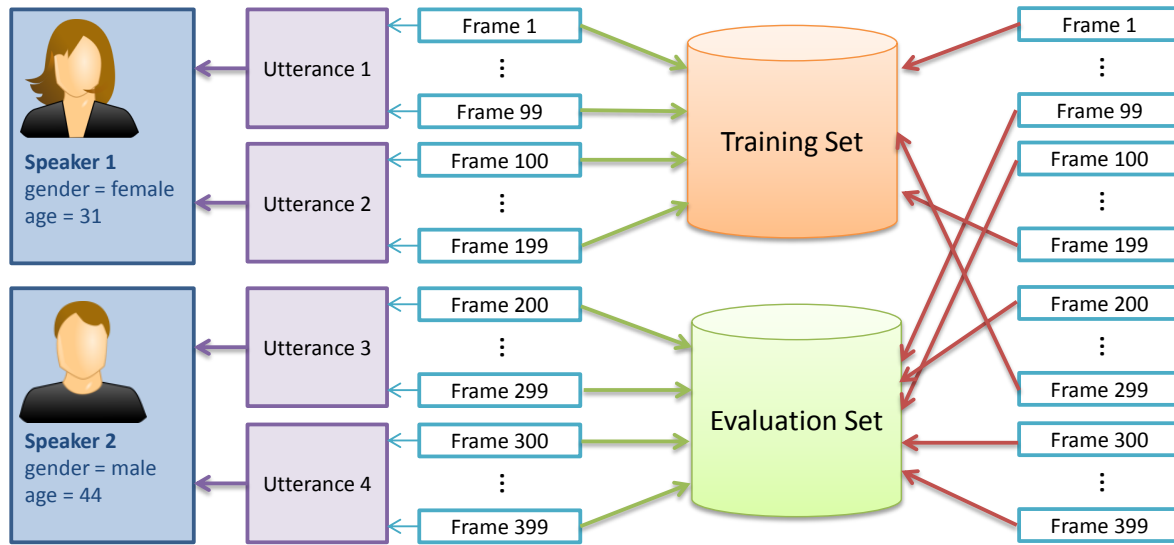


Figure 2.20: Example of a configuration that uses different units (frames) for training and evaluation than for the definition of ground truth (speakers). The data sets must preserve this association; therefore, the left (green) assignment of frames to sets is good while the right (red) assignment causes overlap.

prediction (score or label) for each instance. Figure 2.19 illustrates different components that can be evaluated within the FRISC pattern recognition system. It also illustrates the difference between the evaluation of the system and a single component. At each input point, the data set used for testing is specified. This set should be different from all other data sets used to train or optimize other components in the system with respect to the labeled source data. This has to be respected even when the actual structure of the data differs. For example, in Figure 2.20, the evaluation of the system uses audio files as input and predicts the speaker's class for each. On the other hand, the evaluation of the SVM works merely on supervectors (vectors constructed by concatenating other vectors, as will be explained later) for which scores are supplied by the classifier. Still, both audio files and supervectors can refer to the same ground truth data. When the actual evaluation is performed, this is also called an *evaluation run*.

The technical evaluation process plays an important role for the optimization of system components. It serves as a feedback channel in an automated procedure trying to find an optimal threshold or other parameter setting, e.g. the level of nuisance attribute filtering. In this case, the relevant result is not the performance on a particular parameter setting, but which parameter setting performed best, no matter what the actual result is. As the name suggests, it otherwise works just like a component evaluation. The data set for this type of evaluation is frequently called *development test data set* (or short *devtest* or *test set*). It is a perfectly legal strategy, even for the whole system. There is however one rule that is good practice in pattern recognition system design: The data set which is used for

the final performance rating of the system, i.e. the evaluation set, should not be used for optimization of any kind. “Optimization” or “tuning” in this sense has to be defined very strictly as any repeated technical evaluation process with a different system or parameter configuration. In other words, the final performance evaluation is to be done only once, and with data that has not been used before. This may seem like a stringent requirement, but it has clear advantages. It is very much for the same reason that training and evaluation sets differ: It avoids that the parameters are coined on the set that is used to measure the performance, which is a special type of overfitting introduced not by the classifier but this time by the experimenter tuning the parameters (e.g. the SVM kernel function). Some may argue that cross-validation would serve the same purpose, but this is only partially true. While it certainly improves the robustness, a true objectiveness is only guaranteed for a single evaluation run. The parameter tuning occurring *between* the runs is still subject to adaptation effects, since – even with fully randomized sets – some data used for system evaluation now was used for training and thus for determination of the current parameters at some point before. Also, a naive implementation of cross-validation as implemented by most tool suites would be suboptimal even for a single evaluation run, unless it takes into account the disjunctiveness of speakers and the length of the material during the randomization part as well (see Section 5.5.3)⁹. Nevertheless, there is a downside to the training-evaluation-set split-up as well: Having a static, random evaluation set bears the risk of using data for performance measurement that is not statistically representative. It could for example have artifacts such as multiple pathological voices. While randomization reduces this risk, the only real solution is a careful manual selection of the set by the evaluation site, which may not be feasible in practice. As a rule of thumb, in a single evaluation run, cross-validation might give a better idea of the absolute performance, while fixed sets facilitate a comparison between runs or between systems.

Since the temptation not to follow these guidelines can be considerable, often bringing up the argument that the data could be of better use to improve the system when in the tuning process (which is usually true to some extent), a widely accepted measure goes one step further: A neutral party is needed to maintain the whole available data, with no other party having access to it. The data is split in two disjunctive sets by this so-called evaluation site, and one of the sets (the training set) is sent out to the research site including full labels. It is up to the research site if it wants to split off any number of own test and evaluation sets from this data. Finally, the evaluation site releases the *unlabeled* evaluation set and collects the predictions or raw scores from the research site. It can then compute the evaluation metrics locally using the correct labels. Once a result has been submitted, no further submission is allowed. This is also called a “one-shot” evaluation, and one of the main promoters of this approach is the *National Institute of Standards and Technology* (NIST) with its annual NIST Speaker Recognition Evaluation (SRE) series. This ensures a fair and transparent process when more than one party is taking part in a comparative evaluation or challenge, such as the

⁹Even if it does, cross-validation is not practical for FRISC since the speaker grouping/speech balancing combined with equally sized, disjunctive validation sets considerably reduces the amount of available data.

aforementioned series, the NIST Language Recognition Evaluation series, or the Interspeech 2010 and 2011 Paralinguistic Challenges.

Contrary to what is discussed above, there are also components which are never technically evaluated, not even in the system evaluation. For example, in the FRISC approach, one of the core components (a GMM) is trained during the system evaluation, but is never evaluated as a model.

The remainder of this section will discuss the different metrics that are suitable to express system performance. There are many different metrics, and this section focuses on some important ones, which support the goals formulated in Section 1.2. We distinguish between two major evaluation set-ups: the *detection* task and the *identification* task. The wording already suggests that the task or goal is a criterion that makes them different, but their choice is also related to metrics. In some cases, technical constraints limit us to choosing metrics from only one of these methods. For instance, a DET curve cannot be drawn if only class decisions are available. If sufficient data is available, as in the FRISC experimental set-up, applying both methods yields complementary performance results that allow a more distinguished, task-specific view on the system.

Detection Task, Detection Cost Function, and Detection Error Tradeoff Curves

The goal of the detection task is to determine whether some sample x belongs to a class c or not. Or, as formulated by Leeuwen and Brümmer (2007) in terms of speaker verification: “Given two recordings of speech, each uttered by a single speaker, do both speech excerpts originate from the same speaker or not?”. Some tasks are native detection tasks, such as speaker verification. Other classification tasks can be converted (e.g. scaled down or merged) into a detection problem. For instance, instead of looking at multiple age classes, we might just be interested in whether the speaker is a senior or not. For certain scenarios, the answer to this question may be perfectly sufficient. And even if not, considering classes separately may give a better insight into the classification problem because the problem space is simplified and cleaner of possible class interferences. It is also much easier to fine-tune the behavior with only two classes. In the identification case, it is even difficult or impossible to see which class scored second. The possible condition that one class always has just slightly lower scores than another class is not revealed by the purely identification-based metrics. Further positive aspects noted by Leeuwen and Brümmer (2007) are a more standardized evaluation and independence from a fixed number of classes.

Every identification task can be reformulated as several detection tasks, one for every class. One might assume that an identification system can be converted easily as well, for example into a detector for class c that outputs 1 if the multi-class decision matches c and 0 otherwise. However, such a detector would neglect that in reality, a detection task might have completely different cost associated with the decision. For example, the decision in favor of one of four age classes, children, might have been a close one compared to the class of senior speakers (e.g. due to their similarity in pitch). A public terminal or voice-controlled elevator, which

could provide additional support for elderly people, might be more sensitive to this age class, signaling a match even though the four-class classifier has a higher score for a different class. Moreover, a detector that returns only 0 or 1 provides no insight into its workings and cannot be tuned. If the identification system provides scores for each of the individual classes, then we can treat the system like C different detectors that can also be optimized for the detection task. This configuration is quite common, probably more common than native binary tasks, yielding the benefits mentioned above, but still allowing to test the classifier against multiple classes. Hence, while strictly speaking the actual detection task involves only two classes, our evaluation procedure for detection tasks can consist of multiple classes.

In such configurations, the class for which the scores are reported in the same way as from a binary classifier is referred to as the *target class*, the remaining are the *non-target* classes. It can also be thought of as the “active” classifier which is currently being tested. The evaluation procedure for a detection task involving C classes is roughly as follows (see also [Martin, 2007](#); [Leeuwen & Brümmer, 2007](#)): A combination of sample and target class (detector) is called a *trial* $\tau = (x, t)$. The case when truth and target class match, i.e. $c_x = t$, is called target trial. A score $s_t(x)$ is obtained for each such trial by running the target classifier on the trial. For some metrics, an actual decision is required by the classifier. Here, it is common to interpret scores $s > 0$ as accepted samples and $s \leq 0$ as rejected samples. Yet to be more generic, we will use a class-specific decision threshold Θ_c instead of zero.

There are two basic metrics that can be computed from every detection task evaluation: *misses* and *false alarms*. A false alarm (or false positive, type I error) is encountered when the classifier accepts a negative¹⁰ sample, i.e. $s_t(x) > \Theta_{t, c_x} \neq t$ (non-target classified as target). A miss (or false negative, type II error) occurs when the classifier rejects a positive sample, i.e. $s_t(x) \leq \Theta, c_x = t$ (target classified as non-target). Based on the counts of both types of errors, the *miss probability* (or simply *miss rate*) P_{Miss} is defined as the number of misses divided by the number of target trials, while the *false alarm probability* (or *false alarm rate*) P_{FA} is the number of false alarms divided by the number of non-target trials. The false alarm rate can also be split by non-target classes, i.e. $P_{FA}(c)$. A derived metric is the *uniform error rate*, which weights both errors equally: $P_{Err} = 0.5 \cdot P_{Miss} + 0.5 \cdot P_{FA}$.

Past experience with specific applications has shown that these metrics are sometimes not sufficiently generic, for example because one error is more critical than the other. Also, the uniform error rate does not take into account that the number of target and non-target samples could be different, which is the case for any balanced evaluation set with more than two classes. Therefore, NIST researchers have created the more general *detection cost function* (DCF), which measures cost rather than probabilities. It is defined as ([Leeuwen & Brümmer, 2007](#); [Martin, 2007](#))

$$C_{Det} = C_{Miss} \cdot P_{Miss} \cdot P_{target} + C_{FA} \cdot P_{FA} \cdot (1 - P_{Target}).$$

C_{Det} is also called *expected cost of detection errors* ([Leeuwen & Brümmer, 2007](#)). As opposed to the error rates, it is not limited to a fixed range. P_{Target} is the prior probability of a

¹⁰In case of two classes in a closed world, we simply let the first class be *positive* and the second *negative*.

Detection Task Error Rates and Cost												
Target Class												
		AM	CaE	ChE	CM	FF	MF	MM	RM	SC	TS	∅
False Alarm Rate by non-target	AM		24.86%	23.91%	2.55%	41.62%	0.9%	3.35%	6%	8.65%	1.55%	12.6%
	CaE	33.82%		47.22%	8.05%	47.57%	4%	6.75%	10.56%	13.41%	4.35%	19.53%
	ChE	6.5%	47.97%		33.87%	8.45%	20.41%	26.06%	20.76%	18.21%	34.02%	24.03%
	CM	3.25%	23.01%	34.12%		4.45%	29.96%	16.61%	23.41%	26.61%	37.72%	22.13%
	FF	36.27%	40.02%	23.36%	2.45%		0.75%	2.9%	5.7%	8.7%	1.5%	13.52%
	MF	0.85%	9.2%	14.26%	22.66%	1.65%		10.16%	25.61%	26.86%	34.62%	16.21%
	MM	4.95%	14.26%	27.01%	12.06%	6.9%	12.56%		8.3%	11.26%	10.91%	12.02%
	RM	3.9%	12.76%	19.41%	21.81%	7.35%	26.86%	8.7%		58.03%	30.82%	21.07%
	SC	3.25%	11.66%	18.56%	20.36%	5.45%	36.57%	10.36%	42.17%		29.86%	19.8%
	TS	1.9%	8.65%	20.86%	27.01%	2.45%	42.22%	8.7%	32.07%	33.67%		19.73%
∅ FA Rate		10.52%	21.38%	25.41%	16.76%	13.99%	19.36%	10.4%	19.4%	22.82%	20.59%	P _{FA} = 18.06%
Miss Rate		8.75%	16.01%	17.21%	13.16%	6.95%	15.21%	12.16%	16.66%	17.36%	13.01%	P _{Miss} = 13.65%
Uniform Error		9.64%	18.69%	21.31%	14.96%	10.47%	17.28%	11.28%	18.03%	20.09%	16.8%	P _{Err} = 15.86%
Cost		10.35	20.84	24.59	16.4	13.29	18.94	10.58	19.12	22.28	19.83	C _{Det} = 17.62

Figure 2.21: Error rates for an example task. Each column represents an individual detection task, while the table as a whole corresponds to an identification task (which is however never interpreted as such by these numbers). The applied DCF uses costs of 1 and $P_{Target} = 0.1$.

target. It is usually obtained by counting target samples. In a case with C balanced classes, it should be equal to $\frac{1}{C}$. However, if the expected distribution in the target application is different, it may be more reasonable to set the parameter to that value. The constants C_{Miss} and C_{FA} can be chosen arbitrarily to express the different costs of misses and false alarms. NIST in their evaluations often chooses $C_{Miss} = 10$ and $C_{FA} = 1$, i.e. misses are considered much worse than false alarms. Figure 2.21 shows how each of these numbers can be reported on what is originally a ten-classes speaker identification task.

The presented error and cost metrics all require “hard class decisions”, i.e. a threshold to be defined, but do not depend on actual scores. In fact they discard all the additional information that scores might offer. Assuming that we are dealing with a classification algorithm that actually covers a range of scores (or at least more than two discrete values), as most algorithms introduced in Section 2.5.5 do, the choice of a different decision threshold Θ might also result in different decisions and hence different error metrics. Choosing the appropriate threshold is a type of score-level optimization and often called *calibration*, and the aforementioned metrics measure this calibration effort. When C_{Det} is chosen as the error metric, then one goal of system optimization is to find the optimal threshold Θ that minimizes C_{Det} . Since the cost function is linear, the global optimum can be found in $O(n)$ with respect to a given development test set. The minimum C_{Det} is called *minimum detection cost* and denoted by C_{Det}^{min} .

According to our error metrics, the result could be better or worse with a different threshold. It could also be better in terms of one metric and worse for the other. This is actually a kind

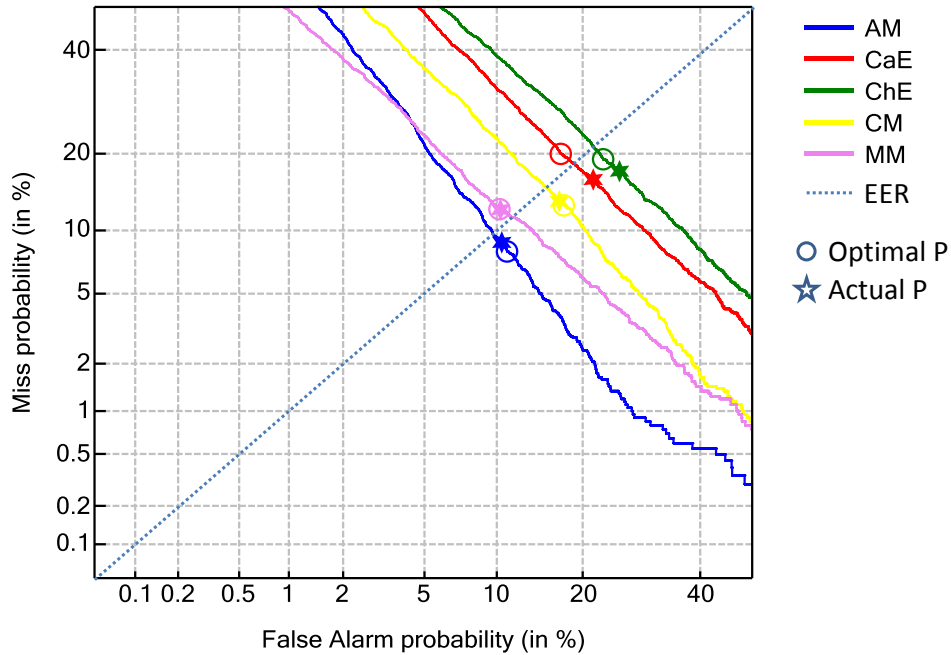


Figure 2.22: Example of a detection error trade-off curve. A five-class speaker recognition problem with each line representing the detection performance of the corresponding target speaker. The dotted line that indicates the equal error rate has been added for further illustration. Optimal P was chosen with respect to $P_{Target} = 0.5$.

of trade-off that is often investigated for scenarios where the errors have different importance. Also in the general case, studying which trade-offs a system *can* produce is very insightful. A different way to become aware of this trade-off used by [Leeuwen and Brümmer \(2007\)](#) is to plot scores for target and non-target trials and notice the overlap of both curves. It is generally true that a reduction of misses causes an increase in false alarms, and vice versa. However, the amount of the change varies. Plus, some systems are relatively better than other systems in both metrics. Essentially, we would like to have an evaluation metric where the dependency on a specific Θ is removed. Such a metric is also said to rate the discriminative aspect of a system.

A way to visualize the aforementioned aspects lies in the *detection error trade-off* (DET) curve ([Martin, Doddington, Kamm, Ordowski, & Przybocki, 1997](#)). An example of such a curve is shown in Figure 2.22. First and foremost, it plots the two metrics P_{FA} (on the x-axis) and P_{Miss} (on the y-axis) against each other. Thus, in order to draw a DET curve, we need no “hard decisions”, but merely a score where higher means better. This can be exploited to even benchmark the performance of models that produce only a likelihood, such as generative models. The curve is created by connecting the values for all possible thresholds of a single target class in the chart. Similar to the way how ROC curves are created, a single iteration

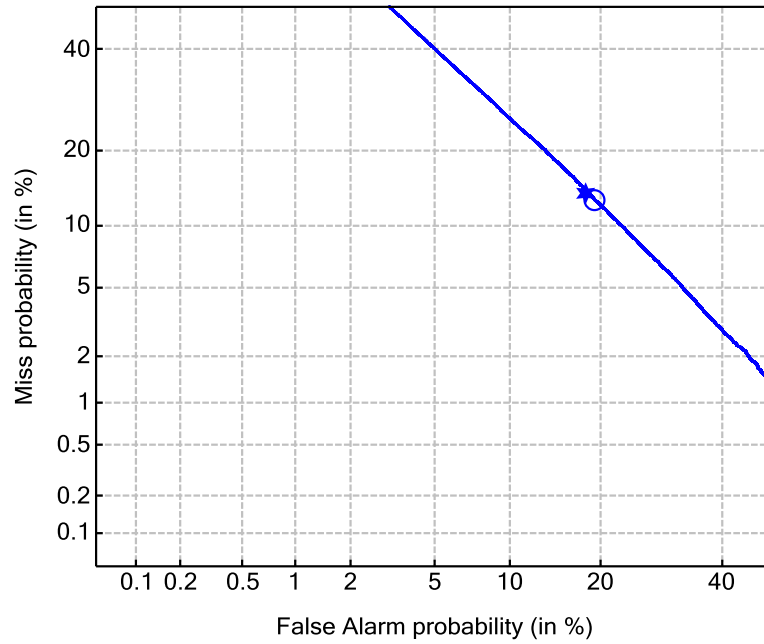


Figure 2.23: This DET curve is based on the same data as Figure 2.22, but it displays an average performance across all classes (combined DET plot).

over all trials with accumulative statistics is sufficient to draw the curve. Its “smoothness” then also depends on the number of samples used. Multiple classes can be compared by adding them to the chart as separate curves. What is special about the plot is the non-linear scaling used on the axes: It uses the *quantile function of the normal distribution* (or *probit function*), which is defined as $Q(p) = \sqrt{2} \operatorname{erf}^{-1}(2p - 1)$, with erf^{-1} denoting the inverse error function (Leeuwen & Brümmer, 2007). The function is chosen such that a normal distribution of error rates, which can be considered ideal behavior, will result in a straight line. Obviously, this is much easier to read and compare than bent lines and reduces visual overlap. Furthermore, it emphasizes small changes in the upper performance area ($> 5\%$), which appear in the lower left corner. A special type of DET plot is the combined plot as seen in Figure 2.23. This curve shows the average performance of all classes by using trials from all targets for miss and false alarm rate computation, and by further averaging the resulting numbers.

From the DET curve, we can see all possible *operating points* of the classifier (with respect to the data set). There are some special points in the chart to take note of. One is the intersection point of the curve with the diagonal on which $P_{Miss} = P_{FA}$ for points. Because of this relationship, this point is also called the *equal error rate* (EER), i.e. the point where miss rate and false alarm rate are identical. Every system has such a point, and its location can be seen as a very concise representation of the plot, which is completely independent from application parameters such as cost. Another important point is the *optimal decision threshold*, i.e. the point corresponding to C_{Det}^{min} (or minimal P_{Err} in the absence of a cost function). It is usually marked with a circle on the DET curve. Finally, the *actual decision*

threshold, i.e. the threshold that was used to produce the class decisions, can also be marked in the chart, usually by a star¹¹. One has to be aware that this point does not need to be on the actual curve if the threshold was chosen using different samples. This is true for most of the DET curves in the experiment descriptions in this thesis, because it reflects the principle of optimizing and evaluating on different sets (in this case, *test1* and *test2*).

Identification Task, Confusion Matrices, and Accuracy

When we have C classes to choose from and we want to know to which of the classes a sample x belongs, we are performing an identification task. While this is formulated as a closed-world problem, it can be converted into an open-world setting by adding an out-of-set class. To solve an identification problem, we need an algorithm that outputs at least the “winner” class index, possibly by wrapping several binary classifiers in a decision function. Some classifiers additionally provide scores for each class; however, most identification metrics are based on the minimal interface and rarely use this additional knowledge.

An identification-centered evaluation is performed in straightforward manner by obtaining for each sample x a classification result $r(x)$, which can be compared to the truth value c_x . Based on these results, the *accuracy* is clearly the most intuitive and widely used metric and is defined as the percentage of correct predictions, i.e.

$$\text{Acc} = \frac{\text{correctly classified instances}}{\text{total number of instances}}.$$

This number also gives quite a good impression to non-experts of how good the system will function in practice. For instance, an accuracy of 25% means that the system will be right only in every fourth case. To judge the achievement of the classifier, the accuracy always has to be interpreted relative to *chance level*, which is $\frac{1}{C}$ and corresponds to the average long-term accuracy of a classifier that always guesses or always reports the same class index. The further above chance level the accuracy is located, the better the achievement. This is obviously because more classes to decide between generally indicates a more difficult task.

If the measured accuracy is low, there is not much we can read from this number to identify the reason for the bad performance. Therefore, a tool named *confusion matrix* has been developed to be used in conjunction with accuracy to benchmark identification tasks. As the name suggests, the matrix identifies which classes are most frequently confused with what other classes. An example can be found in Figure 2.24. The rows of this matrix list the true (tested) classes, while the columns list the predicted classes. Hence, the cell in row *MF* and column *TS* refers to the instances of class *MF* classified as *TS*. In this case, 243 speech samples uttered by speaker *MF* (or 12.1% of *MF*’s samples) were classified wrongly as being from speaker *TS*. Typically, the matrix focuses on the relative class-wise confusions, in which case the percentages in the cells are amounts relative to the true class, and sum up to 100% in each row. The inverse diagonal (highlighted in green in the figure) contains the correctly

¹¹Sometimes it is denoted by a box, which additionally indicates the the 95% confidence intervals of P_{Miss} and P_{FA} .

Multi-label classification results (Identification Task)											
Tested	Classified as										
	AM	CaE	ChE	CM	FF	MF	MM	RM	SC	TS	Σ
	1422 (7.1%) 71.14%	95 (0.5%) 4.75%	74 (0.4%) 3.7%	14 (0.1%) 0.7%	312 (1.6%) 15.61%	9 (0%) 0.45%	12 (0.1%) 0.6%	20 (0.1%) 1%	36 (0.2%) 1.8%	5 (0%) 0.25%	1999 (10%)
	144 (0.7%) 7.2%	1110 (5.6%) 55.53%	188 (0.9%) 9.4%	40 (0.2%) 2%	378 (1.9%) 18.91%	9 (0%) 0.45%	25 (0.1%) 1.25%	34 (0.2%) 1.7%	50 (0.3%) 2.5%	21 (0.1%) 1.05%	1999 (10%)
	37 (0.2%) 1.85%	325 (1.6%) 16.26%	915 (4.6%) 45.77%	145 (0.7%) 7.25%	57 (0.3%) 2.85%	41 (0.2%) 2.05%	157 (0.8%) 7.85%	66 (0.3%) 3.3%	77 (0.4%) 3.85%	179 (0.9%) 8.95%	1999 (10%)
	23 (0.1%) 1.15%	105 (0.5%) 5.25%	118 (0.6%) 5.9%	1215 (6.1%) 60.78%	24 (0.1%) 1.2%	70 (0.4%) 3.5%	84 (0.4%) 4.2%	80 (0.4%) 4%	97 (0.5%) 4.85%	183 (0.9%) 9.15%	1999 (10%)
	143 (0.7%) 7.15%	160 (0.8%) 8%	68 (0.3%) 3.4%	3 (0%) 0.15%	1583 (7.9%) 79.19%	4 (0%) 0.2%	6 (0%) 0.3%	10 (0.1%) 0.5%	16 (0.1%) 0.8%	6 (0%) 0.3%	1999 (10%)
	5 (0%) 0.25%	42 (0.2%) 2.1%	51 (0.3%) 2.55%	140 (0.7%) 7%	7 (0%) 0.35%	1140 (5.7%) 57.03%	82 (0.4%) 4.1%	154 (0.8%) 7.7%	135 (0.7%) 6.75%	243 (1.2%) 12.16%	1999 (10%)
	24 (0.1%) 1.2%	77 (0.4%) 3.85%	99 (0.5%) 4.95%	47 (0.2%) 2.35%	52 (0.3%) 2.6%	40 (0.2%) 2%	1533 (7.7%) 76.69%	32 (0.2%) 1.6%	38 (0.2%) 1.9%	57 (0.3%) 2.85%	1999 (10%)
	18 (0.1%) 0.9%	61 (0.3%) 3.05%	70 (0.4%) 3.5%	83 (0.4%) 4.15%	51 (0.3%) 2.55%	87 (0.4%) 4.35%	42 (0.2%) 2.1%	1150 (5.8%) 57.53%	280 (1.4%) 14.01%	157 (0.8%) 7.85%	1999 (10%)
	17 (0.1%) 0.85%	61 (0.3%) 3.05%	55 (0.3%) 2.75%	87 (0.4%) 4.35%	37 (0.2%) 1.85%	146 (0.7%) 7.3%	65 (0.3%) 3.25%	242 (1.2%) 12.11%	1121 (5.6%) 56.08%	168 (0.8%) 8.4%	1999 (10%)
	7 (0%) 0.35%	31 (0.2%) 1.55%	61 (0.3%) 3.05%	100 (0.5%) 5%	17 (0.1%) 0.85%	141 (0.7%) 7.05%	42 (0.2%) 2.1%	119 (0.6%) 5.95%	129 (0.6%) 6.45%	1352 (6.8%) 67.63%	1999 (10%)
Σ	1840 (9.2%)	2067 (10.34%)	1699 (8.5%)	1874 (9.37%)	2518 (12.6%)	1687 (8.44%)	2048 (10.25%)	1907 (9.54%)	1979 (9.9%)	2371 (11.86%)	Acc = 62.7%

Figure 2.24: Confusion matrix showing the identification results for a ten-classes problem.

This exhaustive representation contains in each cell the absolute number of instances, their relative number, and their class-wise relative number.

classified instances; its average is equal to the accuracy. In the remainder of this thesis, a more condensed form of this matrix is used, in which the relative number of instances is expressed through the size of a bubble in the background of the matrix (see Figure 4.6 on page 127 for an example).

Other metrics

Precision and Recall. When dealing with a binary classification problem, four cases can be distinguished for any tested sample x : (1) the sample is positive ($c_x = 1$) and the classifier accepted the sample ($r(x) = 1$), which is called a *true positive* (TP); (2) the sample is positive ($c_x = 1$) but the classifier rejected the sample ($r(x) = 0$), which is called a *false negative* (FN, or miss, see above); (3) the sample is negative ($c_x = 0$) and the classifier correctly rejected it ($r(x) = 0$), which is called a *true negative* (TN); and (4) the sample is negative ($c_x = 0$) but the classifier accepted the sample ($r(x) = 1$), which is called a *false positive* (FP, or false alarm, see above). A special interpretation of these values are precision and recall. Recall (also named *true positive rate*, *sensitivity* or *completeness*) describes how many of the positive items were recognized, formally $\text{Recall} = \frac{TP}{TP+FN}$. On the other hand, *Precision* (also named *positive predictive value*) indicates how many of the accepted samples actually are positive, i.e. $\text{Precision} = \frac{TP}{TP+FP}$.

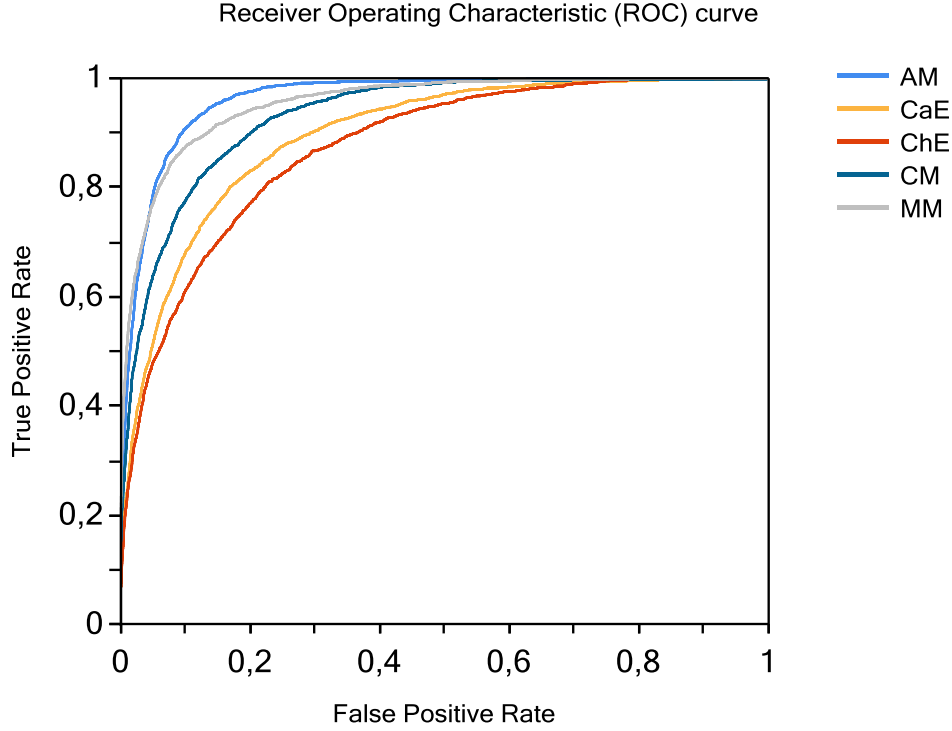


Figure 2.25: Example of a ROC curve created on the same classifiers from Figure 2.22.

F-Score. The F-score or F-measure is a combination of the precision and recall metrics commonly used in machine learning. It is defined as

$$F = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

i.e. both metrics are weighted equally.

ROC Curve. The *receiver operating characteristics* (ROC) curve (Fawcett, 2006) is a visual collection of multiple operating points of a classifier with respect to the false alarm rate (x-axis) and recall (y-axis). In this regards it is similar to DET, although it has been in use for a longer time. An example is shown in Figure 2.25. In general, the further the curve is squeezed into the upper left corner, the better the classifier.

Log-likelihood Ratio Cost Function. In the previous sections, the DET plot and DCF were introduced as a measure for detection performance. Although they both give great insight into discrimination and calibration, respectively, and the DCF offers comfortable flexibility in terms of parameters, they also both face some problems: The DET plot does not plot the full value of scores to the experimenter, while the DCF is strongly dependent on application performance. Also, there is no combined measure for discrimination and calibration. As a remedy, Brümmer (2004) introduced C_{llr} . It measures the ability of a classifier to produce

log-likelihoods that have particularly useful characteristics. One of them is that the range is calibrated on a default threshold $\Theta = 0$. As a consequence, hypothesis discrimination can occur based on the sign of the scores (positive scores indicate acceptance, negative scores rejection). Moreover, the scores are required to express a kind of confidence (i.e. scores closer to zero show less confidence). This type of decision-making is also called *soft decisions*, as opposed to the application-dependent *hard decisions* used to compute error rates. C_{llr} is calculated by integrating over all possible decision thresholds (see Leeuwen and Brümmer (2007, p. 341) for details). In analogy to C_{Det}^{min} , there is also an optimal C_{llr}^{min} that can be computed. In this case however, it is not reached by adjusting thresholds, but rather by applying a warping function to the likelihood ratios.

APE Plot. Being directly based on the C_{llr} metric, the *applied probability of error* (APE) plot (Leeuwen & Brümmer, 2007) is a depiction intended complementary to DET that displays the C_{Det} (with $C_{Miss} = C_{FA} = 1$) on the y-axis in relation to a threshold Θ on the x-axis. It is a metric for all possible operating points. An example can be seen in Figure 2.26.

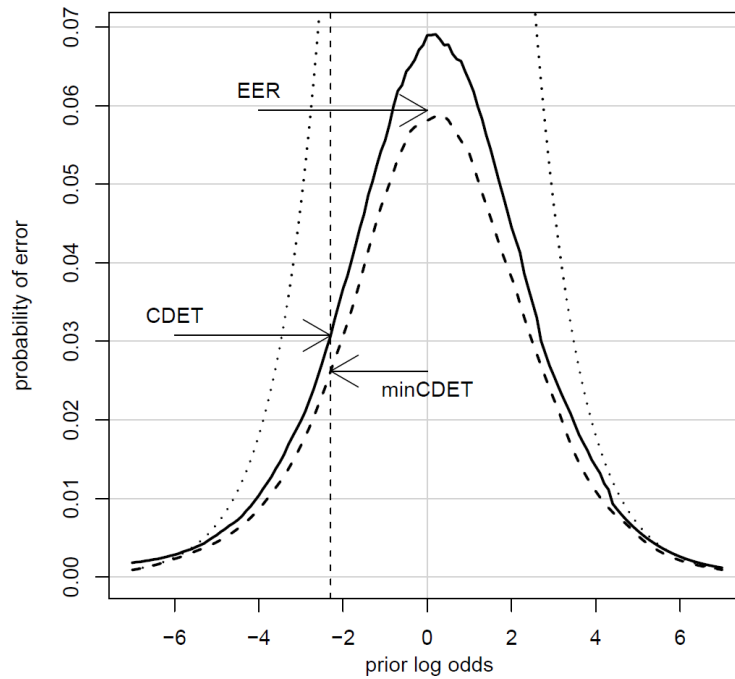


Figure 2.26: Example of an applied probability of error (APE) plot. The different curves are the probabilities based on the actual likelihoods (solid), optimal likelihoods (dashed), and based only on prior probabilities (dotted). (Source: Leeuwen & Brümmer, 2007, p. 347)

2.5.8 Classification vs. Regression

In machine learning, we can distinguish between classification and regression problems. A classification problem exists if one of a countable set of labels should be assigned to an instance in question. Examples are the detection of broken bottles in a bottle recycling plant, the assignment of categories to pictures through image processing and understanding, or the recognition of a fixed set of gestures. The term pattern classification is used to highlight the statistical aspects of the topic, similarly as with pattern recognition (vs. machine learning). A regression problem exists if we want to map an instance to a real-numbered value range. Examples are the estimation of the mass of an object from its image or scan, the prediction of sales numbers based on market observations, or the detection of a car's speed based on video. In other words, both problems differ in the domain of their outputs, which is either nominal/cardinal or real. This difference has an impact on the algorithms that are employed: Classification algorithms can usually only be applied to classification problems, and regression algorithms often work better with that type of problem.

In practice, most applications involving pattern recognition are dealing with a classification problem on the surface rather than with a regression problem. Even if we want to find out the age of person in years, it is age *classes* that represent the result. This ultimately has to do with how the information is used, and in most cases, it is used to make decisions, which is an indicator for classification. That does not mean however that regression *algorithms* may not be an appropriate way to implement the task or to represent errors, as regression results can always be mapped to classes again in simple or more complex ways. In fact, regression was also used for evaluation as part of this thesis (see Section 4.5.3).

There are other types of problems as well, but they are only marginally relevant to the work at hand. Examples are *interpolation*, where the relationship between input and output is known only for some ranges and a solution for the missing ranges should be inferred, and *density estimation*, which is rather an inverse approach to classification in that it asks for the probability of certain features in a new sample when its class label is known.

3 Related Work in the Area of Speaker Classification

The work by Müller (2005), in particular the AGENDER approach, which laid the foundation for many of the concepts in this thesis, represents one of the first attempts to deal with the automatic recognition of speaker age, together with the studies presented in Section 3.1. Previous research has been done however in other areas of Speaker Classification, such as emotion recognition. Recently, the number of publications in the field has increased, expressing a growing interest in the technology and its applications. This is at least partly due to technological advances in hardware, or more precisely, processing power: Applications and servers can work with large volumes of data, they can do sophisticated reasoning and complex computations to improve the user experience, but they still need the actual data to do so, therefore now the data acquisition methods have to keep up. And of course, the application landscape has also changed: Non-intrusive user-adaptive behavior is more common nowadays, although it is also more subtle than in its early days. As a new milestone in age and gender recognition, a special session on paralinguistic speech information extraction has been organized in conjunction with the *Interspeech 2010* conference. This special session was entitled “Interspeech 2010 Paralinguistic Challenge” (Schuller et al., 2010), and it was motivated both by the challenge in the area of language identification organized by NIST in their annual *Language Recognition Evaluation* series (see e.g. *NIST Language Recognition Evaluation*, 2011), by its Interspeech 2009 Emotion Challenge predecessor, as well as by the numerous other “challenges” popular in the machine learning community.

In this section, an overview is given on milestones in related work on Speaker Classification and state-of-the-art technology. The references found herein deal with the extraction of one or more items of paralinguistic information from speech, with a special focus on the speaker properties age and gender. For the latter, a chronological presentation has been chosen since they are often treated in combination, which is also reflected in the classes and features.

In machine learning, and even more in Speaker Classification, there are certain aspects that almost all systems share. This reflects the basic layout of a machine learning system as introduced in Section 2.5. While the building blocks of the systems are often the same, the differences lie mostly in the details. This facilitates a meaningful comparison of the different approaches further than in other areas. Making use of this circumstance, at the end of this section, a table summarizes the findings and attempts to compare the methods according to some basic criteria. The same basic scheme is applied throughout this section to analyze and introduce the contributions individually. For some systems, concrete performance data

is presented. When comparing actual numbers, however, great care must be taken, because even small differences in the specifications, data, or other conditions that may have affected the evaluation, cannot accurately be expressed in the summaries. An exception may be the contributions to the Interspeech 2010 Paralinguistic Challenge, where equal conditions were created for all participants. For completeness, the final comparison also includes the FRISC approach that will be the main subject for the remainder of this chapter.

3.1 Early Work

One of the earliest studies on the automatic estimation of speaker age was undertaken by **Minematsu, Yamauchi, and Hirose (2003)**. This study does not use the chronological age as ground truth to measure the performance, it rather compares the automatic method to the human judgment of the age of voices and evaluates the correlation of both metrics. For either rating, the benchmark is the perceptual age of the speakers. For human listeners, this is by definition the more natural choice, and for the system, it can be assumed to be an easier task, as borderline cases or pathologies are not a disturbing factor in this case. The data originates from three Japanese-language corpora named JNAS (ages 20–60), Senior-JNAS (ages 60–90), and CHILDREN (6–12). Apart from the missing ages 13–19, all speakers were male. Like with AGENDER, this age-dependent combination of corpora can have an impact on the reliability of the results. The actual implementation of the automatic recognizer uses GMMs trained on MFCCs, their first order derivatives, and deltas of the frame-wise power. Silent frames are removed using a method not detailed. One GMM is trained for each of the 320 speakers with 16 mixtures and diagonal covariance matrices. This regression approach uses a weighted sum of all speakers to estimate the numeric age of a test sample consisting of 5 seconds of speech from one speaker, and correlates to the listener’s perception by 89%.

Some additional early progress in terms of regression approaches can be attributed to **Schötz (2004)** (later extended in **Schötz, 2006**). Based on chronological age, 724 gender-balanced speakers between 17 and 88 years from the *SweDia 2000* speech corpus produced the Swedish word *rasa* 3 to 14 times. Their study was aimed at comparing spectral and prosodic features. Prosodic features were F0, intensity (which was normalized), and duration; features in the “spectral” group were F1 – F5, jitter, shimmer, spectral balance/emphasis/tilt, and HNR. Features were tested individually and in appropriate groupings. Recognition was performed on a per-word as well as on a per-phoneme basis, using *Praat* for word segmentation. The models that were trained are CART regression trees. The best per-word result achieved was a mean absolute error of 18.9 years for female speakers with mean F0 and 18.8 for male speakers with the duration. The results generally show a good performance of F0 and other prosodic features. This last result in particular however also indicates a strong dependency of the study on the chosen word.

3.2 The AGENDER Approach

The starting point of my thesis was the AGENDER approach published by Müller (2005), a Speaker Classification system employing pattern recognition methods, and the most extensive work on the automatic recognition of speaker age and gender. AGENDER builds on a thorough study of why and how voices differ between genders and ages, and identifies several features that have been extracted on a sample corpus. These features, which are shown to expose significant mean value differences for a given set of eight classes, have been chosen closely after the hypotheses on which ages are typical points of change for the human voice. These classes and their abbreviated notation are presented in Table 4.1 on page 104. Using a two-layered pattern classification set-up with multiple classifiers, a system has been constructed which reports an estimate of the actual age and gender for a given input speech sample of any length. Several combinations of features and classifiers have been evaluated.

In spite of being a separate line of research, the work at hand takes advantage of many of the findings of AGENDER. However, it also identifies a number of shortcomings and areas where further study and experimentation would pose major opportunities for further improvement. In addition, it tries to establish new specifications for system evaluation that allow for better transparency and comparison with other systems. In the following paragraphs, the approach is presented in further detail, accompanied by various deficits.

Müller (2005) also describes a component for experimentation and a client/server framework for on-line and off-line classification using AGENDER. Facilitating the design and deployment of Speaker Classification technology did not have the same importance as in this work, where it is a major area of contribution. However, the description in this section is limited to the methodological aspects of the main classification approach. The aspects related to the framework and applications are discussed in Chapter 5.

3.2.1 Data

Müller (2005) based his research on three corpora: the German corpus *BAS* and the English corpora *TIMIT* and *Scansoft*. In total, there were 1164 speakers producing 38,202 utterances. The gender is approximately balanced in the data (52% female, 48% male); however, this is not the case with age (see Figure 3.1). Speakers younger than 10 years were not present, which means that the age class *Children* was effectively composed of ages 10 to 12. The training data was generated by obtaining a randomized selection balanced by class, length, and dialog situation. The corpora with higher recording quality were downsampled to 8 kHz and 16 bits sample size (telephone quality).

The heterogeneous corpus data used to train the AGENDER system poses a number of problems: First, the distribution of ages in the data indicates an inherent bias that cannot be prevented by balancing of data alone, since some ages are clearly underrepresented. This applies particularly to the group of adult speakers between 40 and 60 years. It is expected that the performance on real speech would be worse in this age class than for other ages. However, it is not possible to predict to what extent the evaluation results might also be

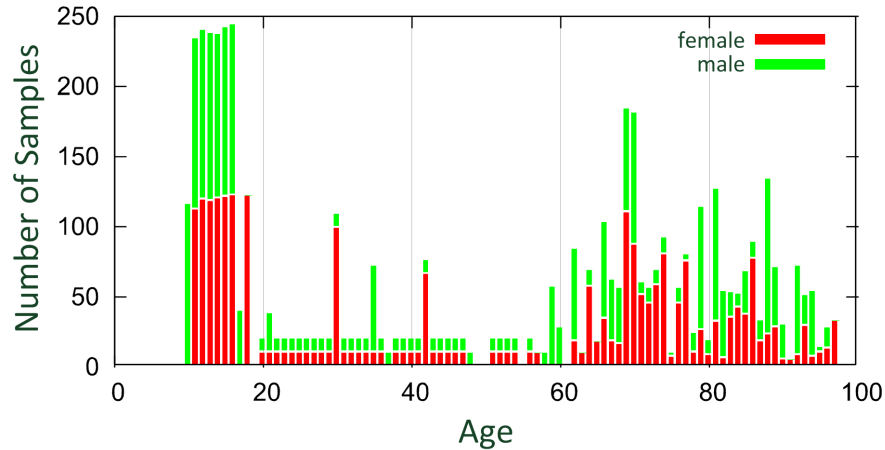


Figure 3.1: Number of speech samples for female (red) and male (green) speakers by chronological age. (Source: Müller, 2005, p. 61)

affected. Second, there are substantial differences (i.e. diverse parameters) between the corpora. Not only do the recording parameters and quality differ (this may be comparable to channel effects), but also the language and dialog situation (read vs. spontaneous utterances). This might pose a high risk of overlaying and overriding the other features, possibly making some age and gender specific effects more or less visible. Even more critical, since there exists an association between certain ages and corpora (i.e. each corpus contained only a specific age range), the machine learning algorithms might inadvertently use corpus parameters to draw conclusions about the age, which results in an over-estimation of the classification performance (a phenomenon often called *corpus effect*, and one of the reasons why a direct comparison between AGENDER and FRISC would not seem sensible). There was no experiment in the original study to confirm or reject such a relationship. In case of language, a language-independence of the method was hypothesized but not confirmed empirically. As will be shown in Section 4.5, it is not safe to make this assumption. Finally, many features do exhibit differences between language families and even languages. Last, the data sets were not disjunctive with respect to speakers, which means that the evaluation would use the same speakers accepted for training, even though with different samples.

Several of the issues described above were acknowledged by the author yet deemed acceptable for the study. Nevertheless, the data used for training and evaluation of FRISC originates from a single new corpus (*SpeechDat-II*). This corpus displays a better distribution of ages and is thus supposed to prevent adverse effects that might be caused by artifacts in the data. Additionally, we introduce new data set selection methods.

3.2.2 Features

There are three categories of features that were used in AGENDER: The acoustic features extracted using the tool *Praat* (derivatives of pitch, jitter, shimmer, and HNR), the speech

activity metrics extracted using *SRSAD* (onset latency, number and duration of pauses, articulation time), and the speech rate extracted using *ENRATE*. All of these features are global or long-term features, which means that they are computed over the full length of an utterance, implying a feature vector of a fixed size (one value per metric).

The use of only global features however completely ignores the time domain and duration information of an utterance. At first, this might not seem to be a problem, since most speaker properties do not change over the course of a single utterance. However, it allocates features computed on short and long sentences an identical weight, which can cause confusion for some types of sentences, e.g. for pitch in a short question such as “How are you?” with the voice raised at the end. Detection and handling of such cases is not possible in AGENDER, where any short-term information is blended into the long-term value prior to the actual classification.

Another issue is that a system that is exclusively based on global features depends on the external specification of utterance boundaries, and returns its belief only after the full utterance is recorded, which reduces the application possibilities in practice. A frame-based system like the one presented herein works on a continuous stream of speech frames in incremental manner, updating its beliefs in very short, fixed intervals. This makes it possible to obtain a result at any given time, which means that a component does not have to wait for the complete recording, but will always receive the best possible solution available (with respect to the expected performance).

Furthermore, it allows speaker changes to be detected by looking for changes in the classification results over a certain time window, which is useful to support multi-speaker scenarios such as diarization.

There is also a persistent controversy about what speaker characteristics are reflected in which features. By focusing on the raw and more low-level short-term cepstral features, which are expected to superset several of the features investigated by Müller (2005), we avoid dealing with certain weak or problematic features in FRISC.

Apart from the basic choice of features, their fine-tuning can also be important. In AGENDER, most features have been extracted by the corresponding tools using their default parameters or by changing them to settings that seemed reasonable. However, and as the experiments in this thesis will confirm, slight changes in these settings can have a major impact on the overall performance. Moreover, they help us to better understand the feature and its relation to the speaker class. This type of small-scale optimization can only be performed in a guided experimental set-up as described in Section 4.3.

3.2.3 Pre-processing and Filtering

Apart from sample rate conversion, the original AGENDER rather goes without preprocessing or filtering, and uses all of the raw features as input to the classifiers. This seems reasonable since the features are high-level features. For most features, pauses, noise-only segments, or short unvoiced segments, are filtered and ignored as part of the underlying extraction

algorithms used. For some features however, such as HNR or the speech rate, it is not clear what effect such “silent” parts within the speech may have and whether it would have been advantageous trying to filter them out in separate process.

3.2.4 Two-layered Classification Approach

The main part of a given AGENDER set-up follows a strict two-layered scheme: The first layer corresponds to the classification phase of a pattern recognition system while the second layer represents the post-processing step. For the first layer, six different types of classifier were compared and used interchangeably as well as in combination: a GMM, a Naives Bayes classifier, a SVM, an artificial neural network, a k-Nearest Neighbor classifier, and a C 4.5 decision tree. In the majority of scenarios, the ANN yielded the best overall recognition accuracy. Each of the first layer classifiers is based on a specific subset of the full list of global features, and is trained on the final set of 8 classes, a subset, or – in most cases – a grouping of classes. Grouping (i.e. merging) two output classes, such as female and male children, into a single class of the model (e.g. children), can reduce the complexity of the model and improve its accuracy. A concrete example from AGENDER (see Müller, 2005, p. 195) is a classifier which uses the features `pitch_quant`, `pitch_mean`, `jitt_la`, `jitt_rap`, `jitt_ddp`, `shim_l`, `shim_ldb`, `shim_apq3`, `shim_apq11`, `shim_ddp`, and `harm_mean`, and classifies three groups: $Cf+Cm+Yf+Ym+Af$, $Sf+Sm$, and Am . The classifiers are composed manually by studying the feature value distributions of individual classes.

The second layer is usually required to be present because most set-ups consist of more than one classifier on the first layer, so the different results need to be combined to an overall decision. As discussed in Section 2.5.6, it can serve both the purpose of classifier fusion and the integration of expert knowledge. Dynamic Bayesian networks (DBNs, see Brandherm & Jameson, 2004) were suggested and evaluated, where the classifiers are connected with each other and conditional probabilities can be specified as appropriate. They can further merge the results from multiple consecutive timeslices, with each timeslice representing one utterance. Figure 3.2 shows an example of such a network. One of the main findings of the investigation of voice aging by Müller (2005) was a different progression for men and women. This was modeled on the second layer by using gender-specific classifiers for age classification, and by linking the conditional probability of the age classifier to the result of a gender classifier (which would be a predecessor node in Bayesian network schematics).

This design is quite flexible, but faces a structural problem: It consists of numerous variables (number and type of classifiers, classes, features...) which are set only by the estimation of the human experimenter, and could be subjective and suboptimal. For instance, the choice of conditional probabilities for the Bayesian network is a difficult task, especially with a growing number of nodes, when it inevitably becomes very opaque. An automated experimental parameter selection might be possible, however it is not part of AGENDER and might also prove difficult due to the number of possible combinations. In addition, parameters of the classification methods, such as the number of mixtures in the GMM, were not thoroughly

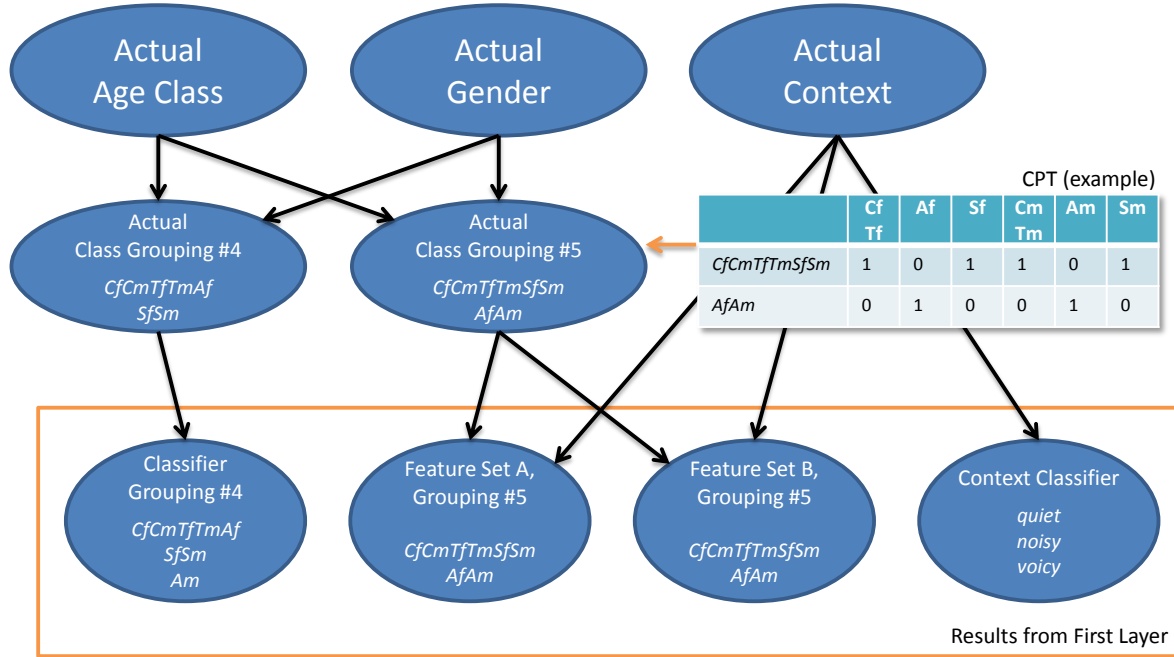


Figure 3.2: First and second layer of a classification set-up in AGENDER. The depicted network represents a single timeslice of the DBN. (Source: Müller, 2005, p. 195)

investigated. The fact that a single type of classifier (the ANN) achieved the best (or next to it) accuracy under almost all conditions, encourages the hypothesis that for similar features, concentrating on the optimization of the parameters of a single classifier might be more rewarding than experimenting with many different classifier types, but testing only few configurations or even only a single default configuration. Therefore, in FRISC, only a single combination consisting of a GMM (generative) and SVM (discriminative) classifier was explored, but using a variety of parameter settings.

3.2.5 Experimental Design and Evaluation

The majority of experiments in Müller (2005) aims at feature analysis, i.e. at building generic models of feature-class-relationships. Gaussian probability distribution analysis, mean value comparison, and correlation analysis are some of the tools applied. The performance of the Speaker Classification itself was not used as a metric in these experiments; however, it was evaluated in a separate series of experiments using a set of feature configurations, class groupings, and classification methods. The method utilized cross validation of classifiers built into the *WEKA*¹ tool (Witten & Frank, 2005).

In the work at hand, the performance evaluation of the FRISC system in the target configuration receives a considerably higher priority as in Müller (2005), of which the experimental

¹ Waikato Environment for Knowledge Analysis

design is not considered sufficient. In particular, for a better comparison, the same target class configuration is used for all experiments in the main experiment series. Further, cross validation is replaced with fixed data sets and a “one-shot” evaluation using strict research/evaluation site separation (see Section 2.5.7). This theoretically allows results not only to be compared within the series, but also with different approaches developed by other researchers.

3.2.6 Conclusion

The AGENDER approach can be considered the “spiritual predecessor” of this work in that it lays the groundwork of modern Speaker Classification in many ways. However, in terms of the actual concepts applied, this work investigates a new approach and constructs a completely novel system design from ground up using this experience. While its main goal is to make progress in the overall performance of Speaker Classification, it also particularly tries to tackle the deficiencies of AGENDER summarized below:

- Unbalanced (i.e. biased) distribution of age classes in the source corpora and particular sparseness for certain ages
- Data inconsistency due to substantial differences in recording conditions, dialog situation, and language between the three corpora
- Possibility of corpus effects due to corpus-class relationships
- Missing empirical evidence for language-independence of the method
- Sensitivity to utterance length by consideration of only long-term features
- No consideration of cepstral features
- Some features computed on non-speech regions without investigation of potential adverse effects
- No incremental processing of speech (streaming) and dependency on external segmentation
- Manual (as opposed to algorithmic) selection of parameters and many parameters remaining unexplored
- Missing systematic evaluation of different configurations, in particular for the second layer
- Parameter tuning performed on evaluation set (i.e. no “one-shot” evaluation).

3.3 First Speaker Classification Workshop and Subsequent Studies

The next milestone after AGENDER is the study performed by [Metze et al. \(2007\)](#), which describes and compares the performance of four automatic combined age and gender classification approaches, and is the first documented comparison of systems from different contributors using the same data sets and separating the research from the evaluation site. The systems participating in the study were simply named *System A*, *B*, *C*, and *D*. This evaluation is also the first published usage of the *SpeechDat-II* corpus for this purpose. Seven classes were used corresponding largely to the AGENDER age specifications as found in Table 4.1 on page 104, with only children spanning one more year. A special condition during the evaluation were the *short* and *long* evaluation sets, which consisted of particularly short and long utterances, respectively. In addition, the *VoiceClass* corpus, consisting of 660 (unbalanced) native German speakers each speaking 5 to 30 seconds to a voice server, was intended as a special out-of-domain condition. Unfortunately, due to the summarizing character of the paper, not all aspects of the individual systems were described on the level of detail used in the overview and comparison tables in this chapter.

System A is a parallel phoneme recognizer (PPR) with decision layer. For each class, a phoneme recognizer is trained, which outputs a neg-log score. For any given input, the recognizer with the lowest score is designated the winner. The phoneme recognizers were originally designed for use in ASR and language identification applications. They are based on MFCCs that are reduced to 24 features using linear discriminant analysis (LDA), which in turn feed into continuous densities hidden Markov models (CDHMMs). The age/gender class models are actually the phoneme bi-grams for the individual class. This system reached 54% precision and 55% recall on the *SpeechDat-II* evaluation set, which was the overall best mark of the four systems. However, for the short utterances set, it declined to 45% and 46%, respectively, which can be seen as a disadvantage compared to the other systems in the test.

System B computes multiple variations of the features jitter, shimmer, harmonics-to-noise-ratio, and pitch, resulting in a total of 17 features. Similar to [Müller \(2005\)](#), three “first layer” classifiers, more precisely multi-layer perceptron networks (MLP), were fused using a dynamic Bayesian network as the “second layer”: one gender classifier and two gender-specific age classifiers. For optimization of the CPTs, the cross validation set of the gender classifier had been used. This system reached 40% precision and 52% recall in the evaluation. For short utterances, the decline was less than two percentage points, which shows a certain robustness against utterance length variability.

[Ajmera \(2006\)](#) observed a relationship between the fundamental frequency of a speaker and the distance between the signal power spectrum and the spectral envelope estimated by a linear prediction (LP) analysis at the spectral peaks. Based on the presumption of a relationship between this distance metric d and the speaker age/gender, **System C** models d using a GMM. Considering that only a single feature was used effectively, some amount of class overlap is to be expected. As test data confirmed, this overlap is large between adults and children, therefore these two classes were merged, resulting in a lower maximum score

for the system. With a precision of 27% and a recall of 50%, the system is worse than the others, yet sufficiently above chance level to confirm the usefulness of the feature.

Finally, **System D** is related to the attempt by Minematsu et al. (2003) in that it classifies the raw MFCC coefficients 1–12 using a GMM. The GMM however is trained per target class, and uses 256 mixtures for age and 128 mixtures for gender. Further, the first and second order derivatives of the coefficients are added, resulting in a total of 36 frame-based features. The system uses four age classifiers and 4 times 2 gender classifiers, since gender GMMs were trained specifically for each age. In addition, a separate gender classifier distinguishes men and women using the same MFCCs, plus pitch. Fusion of the classifiers occurs in a two-step process (age and gender first, then gender-specific age) and employs majority voting on the frames. Frames were also inspected for silence (pauses, breathing...) using a “power-based criterion”, which was meant to filter out the corresponding frames. System D scored at 42% precision and 46% recall.

In addition to the system evaluation, Metze et al. (2007) also conducted a human listener experiment using the same data. 30 people associated with the organizers, i.e. some from the field of speech technologies, rated different sections of the evaluation data with headphones according to the seven-class scheme, covering about half of the data set. Explicit cues on the target class were tagged by the subjects and discarded (about 1% of the data). The manual annotation reached a precision of 55% and a recall of 69%, which is only slightly better than the result of System A.

Bocklet, Maier, Bauer, Burkhardt, and Nöth (2008) were the first to apply the GMM-SVM supervector approach to the problem of age and gender classification. In their work, they are comparing the performance of a “plain” GMM-UBM to a GMM-SVM pipeline. Like FRISC, they are using seven age/gender classes; however, the age boundaries were adopted from Müller (2005) and Metze et al. (2007). The employed corpus is also *SpeechDat-II* (a secondary corpus, *VoiceClass*, is used for comparison) with the same definition of *training* and *test* data sets as in Metze et al. (2007), which means that the results could potentially be compared to the former. However, there is no mention of a separate *evaluation* set, so the reported results have been obtained through tuning on the test set and might be lower on real live data or in a “one-shot” evaluation. Only frame-based features are examined, namely MFCCs 1–12 and their first order derivatives. A UBM with varying numbers of mixtures is built on all training data. This UBM is MAP-adapted for each speaker. While the MAP relevance factor is not inspected, different types of matrices are used in the EM algorithm, such as full and diagonal covariance matrix. The supervector is built from the stacked means of the speaker GMMs, apparently without further processing or normalization. For three system parameters, a comparison of the performance with different settings is given: the number of GMM mixtures, EM/MAP training parameters, and SVM kernel function. The results are presented in terms of precision and recall. The best combination, a GMM-SVM with 512 Gaussians, full-covariance MAP adaptation, and a linear kernel, achieved 77% precision and 74% recall. However, unlike most other studies, evaluation results were collected in terms of speakers (i.e. one instance corresponds to one speaker), which greatly simplifies the task by

increasing the amount of material per test sample to what equals a rather long turn. While this alleviates the conclusions drawn by the authors, it still shows that the GMM-UBM is considerably outperformed; it achieved only 49% respective 41% in the best configuration.

Sedaaghi (2009) presents a further empirical, heavily engineering-driven approach to optimize the performance of age and gender classification. It is based on the two corpora *English Language Speech Database for Speaker Recognition* (ELSDSR, containing 23 speakers) and *Danish Emotional Speech* (DES, containing 4 speakers). The speakers are approximately gender-balanced, but insufficient information is given regarding the age. The overall low number of speakers could lead to a potential inaccuracy. The age decision is a binary one in this work, discriminating between *young* and *old* with an age threshold set at 33 respective 45 years due to age distribution in each corpus. The strategy applied is a common one in result-oriented machine learning: A large accessory of possible features adopted from another publication is created, upon which multiple types of classifiers are trained and compared. Contrary to the bottom-up approach followed in FRISC, the major amount of work is put into the evaluation of different learning strategies. A disadvantage of that approach is that it assumes that a great deal of intelligence is already part of the classification algorithm, although practice shows that this is often not the case. The features comprise 16 formant features, 35 pitch-based features, 34 intensity-based features, and 28 spectral features, which makes a feature vector size of 113 global features. As classifiers, the author experimented with Naive Bayes, ANN, SVM, kNN, and GMM. The best gender classification was achieved with a linear SVM (96% accuracy), while age performed better using an ANN (89%).

3.4 Interspeech 2010 Paralinguistic Challenge And Beyond

This section summarizes the contributions of the Interspeech 2010 Paralinguistic Challenge and selected subsequent work. The actual challenge was organized by **Schuller et al. (2010)** and took place prior to the Interspeech event, where the results were presented. It consisted of three sub-challenges: age, gender, and affect, of which the first two could optionally be merged into a combined-classes task by the participants. The four age classes are defined using the same boundaries as in FRISC (see Table 4.1 on page 104). The gender property consists of three classes (male, female, children), since the detection of gender for children was beyond the challenge. The corpus used for the age and gender classification tasks was called “aGender”, but it must not be confused with the AGENDER approach. It is actually a refinement of *SpeechDat-II* for this particular challenge. In addition to the corpus, a set of 450 acoustic features supposedly relevant to the task of paralinguistic information extraction was provided by the organizers, consisting of MFCCs, loudness, LSP frequencies, pitch, jitter, shimmer, voicing probability, and others. The features were extracted using *openSMILE* (Eyben, Wöllmer, & Schuller, 2009), which is part of the open-source *Emotion and Affect Recognition* toolkit and was already employed in the Interspeech 2009 Emotion Challenge, although this time with a slightly extended feature set. Throughout the challenge, classification performance is reported as weighted (with respect to instances per class) and unweighted av-

erage recall, with the latter being the decisive criterion for determining the challenge winner. A “baseline” result obtained on the 450-feature set using sequential minimum optimization learned pairwise SVMs with a linear kernel trained with *WEKA* was 49% unweighted average recall (UAR) for age and 81% for gender. As the goal of the challenge was clearly defined as the optimization of this metric, application-driven criteria such as complexity, scalability, and resource-adaptivity were rarely picked up by the contributions.

Bocklet, Stemmer, Zeissler, and Nöth (2010) participated in the challenge with an extension of their earlier concepts (**Bocklet et al., 2008**). They created five different back-ends and fused the results of these into an aggregate score. Three of these back-ends are GMM-SVM supervector systems, one is based on long-term prosodic features, and a fifth uses an iterative frame-based approach. For the GMM-SVMs, three different feature sets were examined: The first set uses MFCCs as described in their previous work. The second set explores a revised variant of perceptual linear prediction (PLP) (**Hermansky, 1990**), which differs from the MFCC features mainly in that linear prediction (LP) smoothing is applied on the Mel filter bank spectrum (hence perceptual). 13 PLP coefficients are adopted. The third set is called *TempoRAI PatternS* or *TRAPS* (**Hermansky & Sharma, 1998**). It considers a longer temporal context (310ms) than the other features, in which it computes time trajectories “smoothed by a Hamming window and transformed into frequency domain”. The 558-dimensional feature vector is reduced to 24 features through linear discriminant analysis (LDA). The GMM modeling step appears to be largely identical to their original work; however, this time utterances instead of speakers are used for building the GMMs, which puts more focus on the SVM. The fourth back-end consists of a feature vector of dimensionality 219, including features based on pitch, energy, jitter, shimmer, speech pauses etc., and that was also used in previous work. Computation of the prosodic features differs, yet they are all aggregated over a complete utterance before being processed by the classifier. Finally, a so-called *Glottal Excitation System* was tested, which follows a completely different and more bottom-up approach: An existing glottis model from acoustic phonetics was evaluated in conjunction with changes of the speech production system that are indicative of age, such as “increased harshness or hoarseness, increased strain, higher incidence of voice breaks, vocal tremor and increased breathiness”. For a detailed description, we refer to the paper and the references cited therein. Of all five single models, the GMM-SVM using MFCCs performed best with 42% UAR for the 7-combined-classes problem. For the fusion, two approaches are compared: early and late fusion. Early fusion refers to fusion on the feature level, in which case a single large feature vector is generated from all models (including three supervectors), resulting in a total of 3878 features, and is classified using the SVM. For late fusion, a meta-level regressor (multi-class logistic regression, MCLR) is trained on the scores of the individual SVMs, which also serves as a means of score optimization. The late fusion outperforms early fusion and achieves 48% UAR on the test set (a result for the combined problem on the eval set was not available).

Gajšek et al. (2010) explored a plain GMM approach similar to **Minematsu et al. (2003)** for gender recognition, modeling three full-covariance GMMs with 512 mixtures and using

the Linde-Buso-Gray algorithm for initialization. Features for the GMM were the 12 lower MFCCs, their first order derivatives, and a short-term energy value. These features were normalized by cepstral mean and variance normalization (CMVN). In addition, 33% of the frames with the lowest energy are removed. The result on the eval set corresponds to 83% UAR. Unfortunately, the authors do not report what the impact of the parameters they chose (frame filtering, GMM initialization...) was, as it would otherwise be possible to compare their findings to those in the work at hand. Gajšek et al. (2010) also state that they tried a GMM-SVM approach, but the result was worse.

A further combination of different classifiers was presented by Kockmann et al. (2010). A total of six different sub-systems was tested both individually and with score-level (or decision-level) fusion. The first and best-performing single system is a GMM-UBM based on 13 MFCCs (with cepstral mean subtraction), first and second order deltas, resulting in 39 features. The authors further removed slow and very fast changes in the spectrum since they believe that this is not indicative of natural speech. They used a RelAtive SpecTrAl (RASTA) filter to accomplish this. In addition, silent frames were removed with the aid of a phoneme recognizer. Channel compensation was attempted, but did not improve any results. From a UBM trained on the whole training set, target class GMMs are MAP-adapted. Results for 64, 128, 256, and 512 mixtures are reported, with a steady increase being observed reaching 46% UAR for 512 mixtures on the 7-class problem. The authors also determined that for the separate detection of age and gender, the combined-classes model with the appropriate mappings performs better than four- respectively three-class models. The second sub-system uses the same features, but performs discriminative maximum mutual information (MMI) based training on class-specific GMMs (versus the MAP applied to the UBM in the first sub-system). The result appears not much different from the GMM-UBM. Next, a SVM operating in one-versus-one mode is trained on the challenge baseline set of 450 features, which are first rank-normalized. As an additional measure, the SVM is not only evaluated directly, but the scores of the individual models are meta-classified using a GMM for mapping to the target classes. This score domain optimization clearly improves the performance in almost all cases. Another subsystem, like in Bocklet et al. (2010), uses 16 PLP features. Again, these features perform worse than the other sub-systems by themselves. Two more sub-systems with similar results were the MFCC-JFA-Eigenvoice-SVM, which is similar to a plain GMM but also incorporates feature space optimization through joint factor analysis (JFA) (Kenny, Ouellet, Dehak, Gupta, & Dumouchel, 2008), and the MFCC-JFA-Anchor-SVM, which introduces anchor models covering prototypic data and adapt the rest similar to the idea of MAP adaptation. Fusion of the systems is executed only on the score level through MCLR. Since the authors both tune and evaluate their system on the same development set, a clean data set separation is not preserved and overestimation has to be expected. Despite this flaw, the contribution of Kockmann et al. (2010) scored best for the age task in the challenge with 52% UAR.

Similar in general procedure is the study performed by Li, Jung, and Han (2010). They developed five different sub-systems, which they combine on the score level. These

sub-systems are a GMM-UBM, three GMM-SVM supervector systems, and a SVM based on long-term acoustic features. The frame-based feature vector used in the first four sub-systems is almost identical to the previous study, consisting of 13 MFCCs including first and second order derivatives, and applying CMVN. The UBM consists of 256 Gaussians in each condition and was MAP-adapted with a relevance factor of 12 to each target class (for the GMM-UBM) or training utterance (for the GMM-SVMs). It is not stated whether those parameters were chosen experimentally. Unlike most other work, [Li et al. \(2010\)](#) use separate sets for UBM training and MAP adaptation (approximately 2.5:1 ratio). For the GMM-SVM systems, the 9984² stacked means were used and normalized prior to the SVM training. To reduce the computational complexity, the discriminative part employs a two-step approach, starting with 21 one-against-one classifiers, followed by seven one-against-the-rest classifiers. A variant of the GMM-SVM uses maximum likelihood linear regression (MLLR), which is also common in speaker recognition, instead of MAP adaptation. Like the latter, it adapts a general “background” model to individual speakers. Since the MLLR matrix itself is used to build the feature vector, this method produces a smaller supervector space, which is further compacted by a LDA. As the method is also more efficient, it might show a better runtime performance. For the third GMM-SVM variant, the authors consider the tandem posterior probability (TPP) supervector. Since the UBM additionally models phoneme-related properties, this vector works like a histogram of class characteristics. Details on this method are given in the paper. The fifth system consists of a SVM trained on the long-term challenge baseline features without further modifications. Fusion occurs by applying a linear weight function to the scores. Here, the weights correspond to the inverse entropy, which however violates the strict tuning set separation criterion in the way it was performed. Concerning single systems, GMM-UBM and GMM-SVM perform similar on seven classes (43% UAR). However, all frame-based models were outperformed by the long-term features (45%). The fusion improved the individual results considerably (53% on the development set).

One contribution to the contest, submitted by [Lingenfelser, Wagner, Vogt, Kim, and André \(2010\)](#), deals almost exclusively with the fusion aspect. First, a classifier ensemble consisting of $n + 1$ classifiers (n corresponds to the number of classes) is built based on the 450 baseline features according to certain feature selection and diversity criteria. The method used for feature selection is the correlation-based feature subset selection (CFS) provided by *WEKA*. Ensembles are known to be helpful in situations with relatively few training instances compared to the number of features. Two classification methods were tested: Naive Bayes and SVMs. The fusion methods, which all work at decision level, are mean rule and cascading specialists (CS). Several variations of these methods are compared. Details are beyond this consideration and can be found in the paper. Most results were reported on the Naive Bayes classifier, which was preferred due to computational complexity aspects. The final UAR on the test set was 37%. The relatively low number even after score optimization can be

²The SVM contains one more dummy dimension of 1 values for more efficient training.

attributed to the choice of the classification algorithm. However, using the SVM did not improve the results. The authors suspect that the method used for feature selection might not be compatible with the SVM.

Meinedo and Trancoso (2010, 2011) present another meta-system engineered from multiple sub-systems. In addition to the *aGender* corpus, three other corpora were used for training: *CMU Kids*, *PF STAR children*, and *Broadcast News (BN ALERT)*. Separate strategies have been followed for age and gender, however, all of the sub-systems train 7 classes and combine the output probabilities accordingly to meet the age or gender challenge specifications. The first two sub-systems for age estimation use the challenge feature set, while one applies a linear kernel SVM, and the other an ANN (more precisely a multilayer perceptron feed-forward architecture with two hidden layers). There is another front-end taking as input 12 PLP coefficients, deltas, energy, and frame-wise pitch, which no other system has used in this combination before. The idea of this design was to pick up the speaking rate of the speaker. Different from most other frame-based systems seen before, seven contiguous frames of features were directly fed into the ANN. In fact, and surprisingly if we consider the other studies, it performs better than the other sub-systems. However, there is another fourth front-end following the more common GMM-UBM approach: A total of 1024 mixtures is built from 28 MFCC-based features, and one GMM is MAP-adapted for each of the seven target classes. Finally, the scores of the four systems are combined using linear logistic regression. For gender, a similar architecture is used, however with slightly different parameters. The challenge results were 49% UAR for age and 84% for gender, the latter being the best result in this category.

Nguyen, Le, Tran, Huang, and Sharma (2010) show how the performance of a traditional SVM can be improved by adding fuzzy membership functions to each of the classes, which is a more advanced way of compensating for noise and outliers than using the C parameter of the SVM. The main challenge however lies in the computation of the fuzzy memberships, for which the authors have previously proposed a solution that promotes the data points in overlapping regions. Three remaining SVM parameters, C and γ , and the kernel function (linear and RBF kernels were compared) were determined experimentally. All 450 challenge baseline features were used after being normalized, and one SVM was trained for each of 7 target classes on the training set. Regarding fuzzy class memberships, the authors found that setting the same weight for all vectors, but a different value for the highly-confused *Children* class, improved the performance by one percent absolute over the baseline to 45% UAR. Learning all feature vector weights following the idea mentioned earlier resulted in approximately the same gain.

A purely GMM-SVM-focused approach to age recognition is explored by **Porat et al. (2010)** as part of the challenge. The authors' hypothesis states that the high-dimensional feature vector usually involved with the GMM-SVM concept poses a potential problem to this approach, and refer to several feature reduction strategies that may be applied. However, in their paper, they investigate a different method in order to avoid the dimension reduction step altogether: Instead of using MAP to create utterance-specific models from the background

model, they calculate a supervector consisting of Gaussian weights. These weights express the average relative importance of each UBM Gaussian for a particular utterance. Since there is one weight for each Gaussian, the supervector is smaller than the stacked means vector of a traditional GMM-SVM; its dimension corresponds to the number of mixtures, which is 512 in the reported study. As features, 12 normalized MFCC coefficients and their deltas were used from only the frames containing speech. The SVM was trained with linear, RBF, and polynomial kernels. In this 4-classes age identification task, the UAR was improved from 51% to 59% percent by using the weights supervector and a cubic SVM. However, these results were obtained using the whole speaker material, which represents very idealized conditions. This seems to be the main reason for the decrease to 43% UAR on the challenge test set, suggesting that the method might not be equally suitable for shorter utterances.

Although not as numerous, there are other recent publications on automatic age and gender estimation apart from the Interspeech 2010 Paralinguistic Challenge. In the work presented by [Mendoza et al. \(2010\)](#), which follows a more theoretically motivated line, the impact of glottal features on three age groups is examined: 15–30, 31–60, and 61–90 years. These features are related to the air pulses produced by the vocal folds and glottis, and are obtained using a method called pitch synchronous iterative adaptive inverse filtering (PSIAIF, see [Pulakka, 2005](#)). They include closing and opening phase of the glottis (Ko , Ka), opening and closing quotient (OQ , CIQ), amplitude quotient (AQ), and others. More well-known features like F_0 , jitter, shimmer, and HNR are also considered. Although the features are individually analyzed with respect to their impact on the age classes using common methods, the authors use automatic feature selection involving a generic algorithm combined with an ANN to come up with two final sets of features, one for women and one for men. For the training and evaluation, data from a Portuguese corpus is used, about which unfortunately not much is documented. 60 female and 60 male speakers are selected to form three data sets (*training*, *test*, *eval*). The employed classifier, a MLP, achieves 83% accuracy for women and 92% for men. However, due to the low number of samples, these results are not overly indicative.

Most of the systems that have been presented to this point use classification into discrete classes for age. Considering the applications, this appears sound. Nevertheless, [Wada, Shinozaki, and Furui \(2010\)](#) argue that the age transition is naturally continuous, and so the performance should become better if regression is used, which is also a reasonable argument. Consequently, their work compares discrete and continuous estimators, in this case SVM and SVR (support vector regression). Comparing these poses a challenge for the definition of performance specifications. The authors opt to go with a seamless target space, therefore an artificial mapping from SVM classes to absolute ages is needed. This is accomplished by using ten overlapping age classes centered on the nominal label, with a window size of 15 years. A second novel aspect in this paper is MLLR based adaptation, which is compared to MAP based adaptation. While [Li et al. \(2010\)](#) also applied MLLR at roughly the same time, the study by [Wada et al. \(2010\)](#) is the first direct comparison. 300 speakers from the *Corpus of Spontaneous Japanese* (with a male/female ratio of 2:1) form

the training and evaluation data, with only a single sample per speaker that is composed of 10 concatenated utterances of 3 seconds each. Cross validation is chosen instead of fixed data sets, but a special algorithm is used to guarantee disjunct speakers. As features, the authors take advantage of 12 MFCCs, their first and second order derivatives, and the log energy. Following the experiment, the SVR always outperforms the corresponding SVM, which is not a real surprise as the error metric is more natural to this type of task. Furthermore, the MLLR variant has a slightly lower mean absolute error (MAE) (9.4 years) compared to MAP (9.9) when using classification, but a higher MAE (8.4) than MAP (7.3) for regression. As an alternative to adapting a GMM, initializing from a HMM is also attempted, which performs slightly better. A plain GMM used for comparison produces the worst estimation (10.9 years MAE). Overall, it remains questionable whether a study like the aforementioned is really suitable to attest a general advantage of regression over classification.

3.5 Summary and Comparison of Related Work on Age and Gender

This section provides a summary of all systems presented in the previous sections. The first part puts the focus on a fact comparison of the different architectures, while the second part attempts to compare the systems in a qualitative way, particularly with respect to this thesis. Table 3.1 A and B use a fixed scheme to relate the work to each other to the extent possible. In cases where multiple sub-systems or parameter configurations were evaluated, the description was usually limited to the best performing one to avoid overloading the table. A question mark denotes items that could not be determined from the quoted publications, while a dash signals that a feature is not available.

The criteria used in this analysis scheme are the following:

Classes: Number and type of target classes used. *A* stands for age classes, *G* for gender classes, and *A/G* for combined age/gender classes.

Age Boundaries: Age boundary scheme that was applied. See Table 4.1 on page 104 for the AGENDER and FRISC age class definitions. $x - y - z$ denotes ranges $x - y$ and $y - z$.

Corpora: The corpora which were used for training, evaluation, or both.

Frame Filtering: Any filters applied to the raw, segmented audio signal to remove less useful frames (usually those without speech).

Score Optimization: Methods for optimizing the system's decision on a meta level, i.e. based on classifier scores.

Evaluation Method: The evaluation strategy applied. This is usually either cross validation or the definition of fixed sets. *One-shot* refers to fixed sets including a special *eval* set that is available only in unlabeled form to the research site.

Frame Features: Lists the frame-based features employed, such as MFCCs. *Deltas* refers to the first order derivatives and *acc* (acceleration) to those of second order.

Frame Step/Win: Indicates the step width (interval) in which frames were extracted, and the size of the window that was applied.

Frame-based Cls.: Denotes the classifier applied to frame-based features. The most common methods are GMM and the GMM-UBM and GMM-SVM combinations.

Global Features: Any global features that are used in the approach (in some cases given as an excerpt).

Global Feat. Cls.: Denotes the classifier applied to global features.

Fusion: Any fusion methods that are applied at the post-processing stage, e.g. to fuse the results of multiple classifiers.

Feature Vector Size: The total number of features. If both global and frame-based features are used, the number refers to the frame-based features.

Feature Reduction: Any automated approaches applied to reduce (optimize) the number of features.

UBM: If a universal background model is used, this column describes the data set on which it is based as well as additional training parameters mentioned.

UBM Mixtures: The number of mixtures in the UBM (and instance GMMs).

Instance GMM: Describes how the instance GMMs are derived from the UBM (e.g. through MAP adaptation), and how much data is used for each instance.

Supervectors: Contains the type of supervector used, if any.

Supervector Size: The dimension of the supervector, which typically depends on the *Feature Vector Size*, *UBM Mixtures*, and *Supervectors* columns.

SVM Composition: Indicates how a multi-class decision is derived from the binary SVM classifiers. The two common approaches are sets of one-versus-one SVMs or one-versus-the-rest SVMs.

SVM Normalization: The normalization method applied to the SVM input features.

SVM Kernel: The best performing SVM kernel function.

Table 3.2 compares the systems described in the previous sections in a binary manner, highlighting areas where the approach taken in this thesis has potential advantages and outperforms the other systems. The criteria for this comparison are defined as follows:

Bottom-up Argument: Assesses whether the design of the approach, in particular the choice of features, was motivated by actual empirical evidence on the aging of the human voice. Following the argument of Hill (2007, p. 35), it is our belief that a comprehensive understanding the underlying processes affecting the features is vital if the results are to be re-used in further work and serve as a basis for continuing scientific progress. Bottom-up studies thoroughly build up their hypotheses before validating or optimizing on the data. Opposed to this, many approaches using the top-down method accept existing sets of features commonly used in other speech-related tasks and focus on the pattern recognition algorithms instead, possibly attempting to explain the observations later. This criterion does not simply consider the outcome, but rather the argument followed in the presentation of the system.

Experimental Parameter Space Exploration: Assesses the presence of a well-founded experimental procedure that provides valuable insight into the meaning and impact of individual parameters. For a system description, this requires that (1) a substantial amount of parameters is tested with different settings, (2) a clear order or optimization strategy is identifiable, (3) conditions are identical for all evaluation runs except for a single parameter being changed, and (4) the presentation is complete, i.e. all results of the various settings are documented.

Best Practices Evaluation: Confirms whether the evaluation process described strictly follows the scientific practice for an objective and comparable system evaluation. This includes the “one-shot” evaluation principle (research and evaluation site separation) and using no grouping larger than utterances as test samples for the majority of experiments. Those systems submitted as part of a public challenge such as the Interspeech 2010 Paralinguistic Challenge (Schuller et al., 2010) usually fulfill this requirement by design, while others usually do not.

Identification and Detection Task: This criterion expresses whether the system evaluation considers both the identification as well as the detection task characteristics. Both types of tasks have different applications in practice, and favoring one may reduce the performance in the other. Similar to the reason for studying ROC and DET curves, comparing both types of tasks gives a more complete idea of the general performance of the method than looking at only a single result.

Clean and Transparent Data Selection: Past experience and preliminary studies show that the selection of training data plays an essential role for meaningful results. This is even more important when the strict evaluation criterion is not satisfied. For a system to meet this requirement, it needs to report on balancing of classes, speaker disjunctiveness of data sets, and balancing of material per speaker.

Frame-based Features: Indicates whether the suggested solution evaluates or uses frame-based features. Frame-based features contain short-term temporal information that

Criterion		Minematsu et al. (2003)									
Classes	A	8x A/G	7x A/G	7x A/G	7x A/G	7x A/G	7x A/G	7x A/G	7x A/G	2x A, 2x G	4x A, 3x G
Age Boundaries	continuous	AGENDER	AGENDER	AGENDER	AGENDER	AGENDER	AGENDER	AGENDER	AGENDER	0-33.99 / 0-45.99	FRISC
Corpora	JNAS, CHILDREN BAS	Timit, Scansoft, VoiceClass	SpeechDat-II, VoiceClass	SpeechDat-II, VoiceClass	SpeechDat-II, VoiceClass	SpeechDat-II, VoiceClass	SpeechDat-II, VoiceClass	SpeechDat-II, VoiceClass	SpeechDat-II, VoiceClass	ELSDSR, DES	agender
Frame Filtering	silence	n/a	-	n/a	-	silence (power-based)	-	n/a	n/a	-	-
Score Optimization	-	-	?	-	?	?	?	train + test	-	-	MCLR
Evaluation Method	cross validation	cross validation	one-shot	one-shot	one-shot	one-shot	one-shot	cross validation	one-shot	one-shot	-
Features	12x MFCC + Delta + Power Delta	MFCCs (24 features)	-	LP-Power distance	12x MFCC + Deltas + Acc	12x MFCC + Deltas	-	-	12x MFCC + Deltas, TRAPS	-	-
Frame Step/Min	10 / 25 ms	n/a	?	n/a	?	?	10 / 16 ms	n/a	10 / 16 ms	n/a	10 / 16 ms
Frame-based Cls.	GMM	n/a	HMM	n/a	GMM	GMM	GMM-SVM	n/a	GMM-SVM	n/a	GMM-SVM
Global Features	-	F0, jitter, shimmer, H2N...	-	F0, jitter, shimmer, H2N	-	Pitch	-	FO-FN, spectral, intensity	FO, jitter, shimmer, MFCCs...	-	FO, energy, jitter, shimmer, pauses...
Global Feat. Cls.	n/a	various	n/a	ANN (MLP)	n/a	GMM	n/a	various	SVM	-	SVM
Fusion	-	DBN	PPR	DBN	-	Majority Voting	-	-	-	-	features, MCLR
Feature Vector Size	25	various	24	17	1	36/37	24	113	450	24	24
Feature Reduction	-	-	LDA	-	-	-	-	-	-	-	LDA
GMM-UBM-SVM	-	-	-	-	-	-	-	-	-	-	-
UBM	n/a	n/a	n/a	n/a	n/a	n/a	n/a	10x EM on training set	n/a	n/a	training set
UBM Mixtures	n/a	n/a	n/a	n/a	n/a	n/a	n/a	512	n/a	n/a	64
Instance GMM	n/a	n/a	n/a	n/a	n/a	n/a	n/a	1x MAP (diag. cov. matrix with adapted means), one per utterance	n/a	n/a	MAP adaptation, one per utterance
Supervectors	n/a	n/a	n/a	n/a	n/a	n/a	n/a	stacked means	n/a	n/a	stacked means
Supervisor Size	n/a	n/a	n/a	n/a	n/a	n/a	n/a	12288	n/a	n/a	1536
SVM Composition	n/a	n/a	n/a	n/a	n/a	n/a	n/a	1vs1	n/a	n/a	?
SVM Normalization	n/a	n/a	n/a	n/a	n/a	n/a	n/a	?	n/a	n/a	?
SVM Kernel	n/a	n/a	n/a	n/a	n/a	n/a	n/a	linear	n/a	n/a	?

Table 3.1: Overview of all age and gender estimation systems described in Chapter 3 (Part A).

Criterion	Gajsek et al. (2010)									
	Classes	3x G	7x A/G	7x A/G	7x A/G	7x A/G	4x A, 3x G	7x A/G	4x A	3x A
Age Boundaries	FRISC	FRISC	FRISC	FRISC	FRISC	FRISC	FRISC	FRISC	FRISC	FRISC
Corpora	agender	agender	agender	agender	agender	agender	agender, CMU Kids, PF STAR Ch., BN ALERT	agender	agender, Hebrew corpus	Rio de Janeiro corpus
Frame Filtering	silence (energy-based)	phoneme rec., RASTA filter	silence (VAD)	n/a	silence	n/a	silence	silence	-	-
Score Optimization	-	lin. GMM, MCLR	fusion weights	part of fusion	part of fusion	-	-	-	-	-
Evaluation Method	one-shot	one-shot	one-shot	one-shot	one-shot	one-shot	one-shot	one-shot	train+test+eval	cross validation
Features										
Frame Features	12x MFCC + Deltas, short-term energy	13x MFCC + Deltas + Acc	13x MFCC + Deltas + Acc	-	12x PLP + Deltas, Energy, FO	-	12x MFCC + Deltas	-	12 MFCCs + delta + acc, log energy	-
Frame Step/Win	?	12 / 20 ms	?	n/a	12 / ? ms	n/a	?	n/a	?	?
Frame-based Cls.	GMM	GMM-UBM	GMM-SVM	n/a	ANN, GMM-UBM	n/a	GMM-SVM	n/a	GMM-SVM/SVR	GMM-SVM
Global Features	-	Challenge Set	Challenge Set	Challenge Set	Challenge Set	Challenge Set	-	FO, jitter, shimmer, H2N, ko, AQ, OQ...	-	-
Global Feat. Cls.	n/a	SVM	SVM	N. Bayes, SVM	SVM, ANN	SVM	n/a	-	-	-
Fusion	-	MCLR	weighted scores	MEAN, CS	lin. log, regress.	-	-	-	-	-
Feature Vector Size	25	39	39	450	?	450	24	39	-	-
Feature Reduction	-	JFA	LDA	CFS	-	-	-	-	Genetic + ANN	-
GMM-UBM-SVM	n/a	training set	UBM set	n/a	?	n/a	training set	n/a	5x EM on training set	128
Instance GMM	n/a	MAP, one per target class	MAP (rel=12), one per target class/utterance	n/a	MAP, one per target class	n/a	-	n/a	MAP, one per speaker	128
Supervectors	n/a	n/a	various	n/a	n/a	n/a	Gaussian weight	n/a	stacked means	1536
Supervector Size	n/a	n/a	9985	n/a	n/a	n/a	512	n/a	1536	1536
SVM Composition	n/a	n/a	1vs1 + 1vsRest	n/a	n/a	n/a	?	n/a	1vsRest	1536
SVM Normalization	n/a	n/a	mean-stddev	n/a	n/a	n/a	?	n/a	1vsRest	1536
SVM Kernel	n/a	n/a	linear	n/a	n/a	n/a	polynomial	n/a	linear	1536

Table 3.1: Overview of all age and gender estimation systems described in Chapter 3 (Part B).

conveys information not present in aggregated features. Details on this will be presented in Section 4.1.4.

Long-term Features or Fusion: Indicates whether the suggested solution evaluates or uses long-term features, i.e. features aggregated over a full utterance. While not part of the main design presented in this thesis, these features are expected to contain supplementary information that might improve the classification result when combined with the short-term information. The FRISC approach was extended with such information in an auxiliary study presented in Section 4.4.

Generative and Discriminative Classification: This column denotes whether the design employs both types of classification methods. Due to the diverse nature of features related to vocal individuality, it is expected that a single type of classifier may not provide optimal results. This is especially true when frame-based features are used, which can model speaker differences well using generative models, but need to be combined to a set of target classes using a discriminative method. The GMM-SVM supervector approach is the most obvious application of this logic, although studies exploring the use of GMM and SVM (or other generative / discriminative methods) in a different way do also fulfill this criterion.

Training Focus on Discriminative Aspects: Examines whether the generative models provide sufficient material for the discriminative step. A system that trains more than a single GMM per target class formally meets this aspect. Obviously, this only applies when generative and discriminative methods are used successively, such as in the GMM-SVM-supervector approach.

Background vs. Utterance GMM Weighting: Our studies have shown that the details of the adaptation step when a GMM-UBM approach is used do have an impact. More precisely, our conclusion is that the preference of new training material over the background model has a positive impact on the classification performance, therefore this weighting is a performance characteristic of the solution. However, all solutions exploring this parameter (often expressed through the so-called *MAP relevance factor*) have been accepted in this category.

Source Domain Optimization: Summarizes optimizations performed on the raw signal level before or independent from the feature extraction. A typical example is silence frame removal.

Score Domain Optimization: Assesses whether the system performs a separate step to optimize the system on a score-level to account for different requirements in the scenario, such as an identification task or a detection task. Specification of a parametrized detection cost function is another example.

Feature Domain Optimization: Reduction of a large feature space can potentially improve the classification performance. This column summarizes all systems implementing one or more automated methods for feature space optimization, such as PCA.

Incremental: Being able to supply a result incrementally with increasing expected accuracy is an important requirement in some domains, such as automotive applications. Many systems can only work on full turns, i.e. they need to wait until an “end-of-speech” signal is given. The systems given a check mark in this category are reported to work in incremental manner. Systems for which the information is not given are expected to not offer this functionality.

Real-time: This category is used to denote systems satisfying the high-performance demands of on-line domains. More precisely, implementations for which the computation performance is no worse than real-time on desktop hardware, will meet the criterion, while those with lower performance or systems for which no performance information is reported fail this criterion.

Resource-adaptive: If the system provides an implementation that enables its optimal use on a wide range of platforms, such as large servers but also embedded platforms, by respecting the resources that are available on the platform, it passes this criterion. This is related to the scalability of the approach.

It should be pointed out that these criteria are aligned to the research questions of this thesis. This is in no way supposed to narrow the contributions of these related studies in other areas not covered by this scheme. Also, the scheme is not intended as a direct benchmark using a single metric, which – apart from being very dependent on the one metric – would not have been possible due to the different specifications (e.g. classes, data) underlying the systems.

A study of these tables reveals that there are some aspects which are rare or not present in other work. Most noteworthy, many of the contributions neither build hypotheses on how the aging process is supposed to affect the features they are using, nor do they seek to find or confirm such relationships in their data. Also, there are only few contributions that perform a systematic iterative exploration of a multi-parameter space. Some studies do present results on certain parameters, but combining these findings with other studies is hardly possible because of the different specifications. Judging parameters against each other requires a single set-up in which all other conditions remain fixed. Instead of such a “deep” evaluation (i.e. the investigation of many different settings), the “broad” path (i.e. the fusion of rather different methods) is more popular in literature. One conclusion that we can draw from the entirety of challenge submissions is that the fusion of several systems nearly always prevails in terms of classification performance. This is to be expected, but it actually encourages our strategy to not exclusively follow the optimization of the global maximum, but to tackle the problem of optimizing a single system through thorough research and experimentation, independently from the meta layer.

System	Procedure						Design Aspects of the solution positively influencing...											
			...Classification Performance				...Processing Performance											
Source	Classes	Bottom-up Argument	Experimental Parameter Space Exploration	Best Practices Evaluation (e.g. one-shot)	Identification and Detection Task	Clean and Transparent Data Selection	Frame-based Features	Long-term Features / Fusion		Generative + Discriminative Classification	Training Focus on Discriminative Aspects	Background vs. Utterance GMM Weighting	Source Domain Optimization	Score Domain Optimization	Feature Domain Optimization	Incremental	Real-time	Resource-adaptive (embedded, scalable)
Minematsu et al. (2003)		~	X	X	X	X	✓	X	X	---	---	~	X	X		X	X	X
Müller (2005)	8x A/G	✓	X	X	X	X	X	✓	X	---	---	X	X	~		X	X	~
Metze et al. (2007) - A	7x A/G	X	X	✓	X	~	✓	X	✓	---	---	?	X		✓	X	X	X
Metze et al. (2007) - B	7x A/G	~	X	✓	X	~	X	✓	X	---	---	X	X	X		X	X	X
Metze et al. (2007) - C	7x A/G	X	X	✓	X	~	✓	X	X	---	---	?	X	X		X	X	X
Metze et al. (2007) - D	7x A/G	X	X	✓	X	~	✓	✓	X	---	---	X	X	X		X	X	X
Bocklet et al. (2008)	7x A/G	X	~	X	X	?	✓	X	✓	X	X	X	X	X		X	X	X
Sedaaghi (2009)	2xA + 2xG	X	X	X	X	X	X	✓	X	---	---	X	X	X		X	X	X
Schuller et al. (2010)	4xA + 3xG	X	X	✓	X	~	X	✓	X	---	---	X	X	X		X	X	X
Bocklet et al. (2010)	4xA + 3xG	~	X	✓	X	~	✓	✓	✓	✓	X	X	✓	X		X	X	X
Gajsek et al. (2010)	3xG	X	X	✓	X	~	✓	X	✓	?	✓	✓	X	X		X	X	X
Kockmann et al. (2010)	4xA + 3xG	X	X	✓	X	X	✓	✓	✓	X	X	✓	✓	✓		X	X	X
Li et al. (2010)	7x A/G	X	X	✓	X	~	✓	✓	✓	✓	~	✓	~	✓		X	X	X
Lingenfeller et al. (2010)	7x A/G	X	X	✓	X	~	X	✓	X	---	---	---	✓	✓		X	X	~
Meinedo et al. (2010)	4xA + 3xG	X	X	✓	X	~	✓	✓	✓	---	---	✓	✓	X		X	X	X
Nguyen et al. (2010)	4xA + 3xG	X	X	✓	X	~	X	✓	X	---	---	---	X	X		X	X	X
Porat et al. (2010)	4xA	X	X	X	X	~	✓	X	✓	---	---	✓	X	X		X	X	X
Mendoza et al. (2010)	3xA	✓	X	X	X	X	X	✓	X	---	---	X	X	✓		X	X	X
Wada et al. (2010)	A regress.	X	X	X	X	✓	✓	X	✓	X	X	X	X	X		X	X	X
Feld (2011)	7x A/G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓

Key

Feature is present / requirement satisfied



Requirement is partially satisfied



Feature is not present / requirement not satisfied



Feature cannot be confirmed (absence not reasonable to assume)



Not applicable (depends on another criterion)

Table 3.2: Feature comparison of related approaches in the area of age and gender classification. Metze et al. (2007) A – D refers to systems A, B, C, and D presented in the paper. In the *Classes* column, *A* refers to age classes, *G* to gender classes, and *A/G* to combined classes as used in FRISC. In the cases where three gender classes are used, *Children* are counted as their own class.

Apart from the challenge contributions, rigorous evaluations and accurate data selection are also surprisingly rare, which questions the whole result of these studies. Bad source data (heterogeneous or unbalanced) are among the most common problems, closely followed by overlapping data sets (tuning on the evaluation set) and per-speaker evaluations. Looking at the design aspects, it is also easy to see that virtually no other system explores all the different areas of design and optimization within a single system. Especially the feature selection seems to be an under-investigated area.

Finally, the most obvious deficit of the other systems is their non-consideration of processing performance related aspects. It is not realistic to believe that the selection of a system only depends on its recognition performance. Its cost and flexibility, to which these runtime criteria are immediately related, are qualities of an approach at least as important. For instance, it cannot be confirmed that any of the previous systems can work incrementally and in real-time. Algorithm restrictions and complexity boundaries may prevent such an approach to scale accordingly if an on-line application with fast response time requirements demands it. The absence of these last aspects does not diminish the contribution of these other systems, it rather points out an area where little progress has been made yet, and which is one of the main areas where this thesis delivers results. Overall, the work at hand provides a more complete view of the classification task than most other systems.

As the last row in Table 3.2 indicates, this work clearly attempts to resolve a number of the issues left open by previous work in order to make progress in the overall goal of bringing forward Speaker Classification research, or more precisely the challenges formulated in Section 1.2.

4 The FRISC Approach

4.1 Concepts and Method

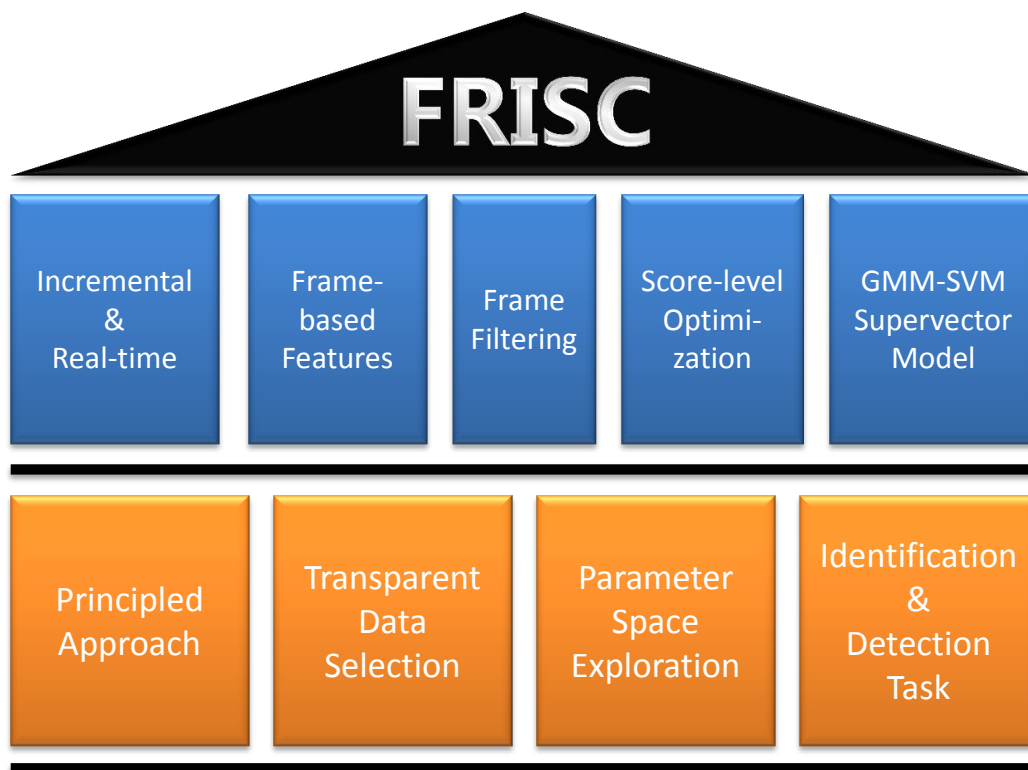


Figure 4.1: Concepts (blue) and methods (orange) that are the foundation of FRISC.

The FRISC approach is built on several main pillars, which refers to both the concepts used to construct the system, as well as the methodology, these concepts are based upon and which affect the way how the system is assembled. Figure 4.1 portrays these building blocks that compose the system as a whole, graphically indicating their relationship. The basic methods (the orange blocks in the figure) were already introduced in Section 3.5. This chapter focuses on the results of the application of these methods, which are the FRISC system design and the experimental investigation of parameters and other aspects.

The AGENDER approach was presented extensively in Müller (2005) and performed respectably well considering its novelty. However, it focuses on a single class of features, namely

long-term (or global) features, such as pitch, jitter and shimmer, which are determined over a whole utterance of arbitrary (though typically short) length. Recent progress in speech-based classification and established concepts of speech recognition serve as a motivation to return to the feature selection step and broaden our view into a different direction than before. That direction is frame-based features, which makes up the first pillar of the FRISC approach. These features are computed in much smaller, fixed intervals, which means that they carry more information and their number is not constant. In consequence, the procedure used for long-term features is not applicable anymore, hence some other way of processing was implemented: the GMM-SVM-supervector approach. The name of this method already implies the choice of two specific pattern recognition methods, which are a Gaussian mixture model for frame-based features and a support vector machine for global features. The shift to a GMM-SVM concept is another part of the foundation of FRISC. In its default configuration, it also reduces the complexity of the classification and post-processing layer, thereby favoring parameter optimization over model selection.

The third major concept introduced is the filtering of frames and supervector features. The common idea behind filtering is that in machine learning, additional information can sometimes degrade the performance of the learning algorithm, by introducing artifacts which blur decision boundaries or, conversely, favor overfitting. Filtering out frames with low energy levels, i.e. silence frames, which are supposed to contribute little information to the decision, is one example. Another example is the high-dimensional supervector. With tens of thousands of features, it is beyond any option for manual analysis, yet it is hypothesized that not all of the GMM-derived features are actually meaningful.

The fourth pillar is the optimization of classifier performance on the score level, which is also a type of meta-classification. A classifier that makes a decision can usually be tuned to be biased more towards the one or other decision (see Section 2.5.7), resulting in different performance metrics. The native, unaltered output of the classifier, can sometimes already be the desired optimum in a two-class (binary) classification scenario, but very often, it is not. Sometimes, the native classifier scores are not meaningful or do not represent a decision at all – it is up to the application to interpret the value range in terms of an actual decision. And even if they are, it may be necessary to adjust them to favor misses over false alarms or vice versa. In the multi-class case such as the age and gender recognition, we face an additional complexity by distinguishing between identification and detection performance, which are two sometimes mutually exclusive goals that can be optimized for separately.

Finally, the runtime considerations of the system complete the list of major FRISC concepts. While this aspect does not transfer into a single concept, it is rather an endeavor propagated into the design of the other concepts. It means, for example, that the selected algorithms have to allow incremental processing of audio data, which is not true for some algorithms such as cepstral mean subtraction. For the main part however, the idea manifests itself in the development platform and the build process used to compile the actual classification modules to be efficient and resource-adaptive (see Chapter 5).

In the following sections, the complete FRISC scheme is explained in detail.

4.1.1 Task Description

The motivation behind this work, the driving force for dealing with and improving Speaker Classification, has been stated in the previous chapters. It envisions many possible applications for the technology in various domains. Despite this variety, the classification technology developed in this work has emerged from a concrete task in a specific project. The goal of this project was to enable the use age and gender recognition for applications in call centers. This task is basically an advancement of the task that was promoted by AGENDER, but some details, such as class definitions and the available data, have changed. The task obviously has a strong influence on the specifications of the classes, suitability of data, performance metrics, runtime properties, certain parameter configurations, etc., which has to be taken into account when considering different application areas.

The fundamental goal remains the estimation of the age and gender of speakers. For age, a set of four classes has been chosen. This number seems reasonable with respect to the initial results that were obtained in this range. However, contrary to Müller (2005), the classes were not motivated by the feature data and the biological relationships that manifest themselves within, but by the requirements of the application. In other words, the age boundaries have been placed where it is interesting to know the class membership, but not necessarily where it is easy to determine. Such externally imposed restrictions, which are rather pragmatically than scientifically motivated, are common in industry projects. One effect is that it most certainly makes the task more difficult. A positive effect is that the outcome is usually more useful in practice, e.g. for personalization of in-car services. In our case, we distinguish *Children* (speakers under 15 years), *Young* people (speakers between 15 and 24 years), *Adults* (speakers between 25 and 54 years), and *Seniors* (speakers over 54 years). It can be assumed that these age boundaries reflect market research that shows for example an interest into a similar range of products among the members of a class.¹ For gender, we distinguish between *male* and *female* for all classes except for children, because their voices do not yield sufficient features to guarantee at least a minimal accuracy above chance level (see Müller (2005) for literature or Metze et al. (2007) for empirical evidence for human raters).

The resulting set of seven classes is listed in Table 4.1. For comparison, the age boundaries from AGENDER are shown as well. Within this thesis, classes are commonly abbreviated using the two-letter notation, using one capital letter for the age and a second small letter for the gender, e.g. *Sf* denotes *Senior female*.

The main corpus used to train and evaluate this system is called *SpeechDat-II*. This corpus is rather new and was not available for the AGENDER experiments. It was provided as part of the task and shares many characteristics with the intended target environment. It contains the voices of 954 German speakers, each taking part in 18 turns of up to six sessions. Each speaker called in to an automatic recording system, where they had to speak some text.

¹Because the distribution of ages still remotely resembles the original classes, the names of the age groups were kept, even though they are no longer accurately describe their members. In case of the class between *Children* and *Adults*, the term *Young* was preferred over the also occasionally used *Teenagers*

Class		Ages (AGENDER)	Ages (FRISC)
Children (C)		< 13	< 15
Young female (Yf)	Young male (Ym)	13 – 19	15 – 24
Adults female (Af)	Adults male (Am)	20 – 64	25 – 54
Seniors female (Sf)	Seniors male (Sm)	> 64	> 54

Table 4.1: The seven-class scheme used in FRISC. Due to the (acoustic) gender-indiscriminability of children before puberty, children form a single class. The age boundaries used in AGENDER (Müller, 2005), where female and male children are separate classes (Cf and Cm), are presented for reference.

The data consists of an altered version of the original *SpeechDat* text material, containing short fixed and free text typical for automated call centers, including single-word commands, digits, and full sentences. The audio was recorded over cell phones and landline connections in 8000 Hz, 16 bit mono format. By looking at the digital signals, the recording quality seems to be on the upper end of the scale within these specifications. The selection of speakers is approximately evenly distributed over the seven target classes, with children also being balanced for gender. A typical utterance is about 2 seconds in length, but there are also some utterances between 3 and 6 seconds. In total, the corpus consists of 47 hours of speech. 90% of the speakers were provided with labels for training and testing, the remaining 10% were only disclosed in unlabeled form before the “one-shot” evaluation.

When dealing with multilingual aspects in Section 4.5, another more specialized corpus named *Lwazi* will be introduced. This will however not be relevant outside of the aforementioned section.

There are other general corpora available containing speech samples with age and genders annotated. Examples are the well-known *TIMIT* corpus (Garofolo et al., 1998), *GlobalPhone* (Schultz & Waibel, 2001), and *BAS* (Schiel, 1998). However, all of these corpora are inferior to *SpeechDat-II* in several respects: Some of them do not contain the full range of required speakers (ages), or they are unevenly balanced, so that certain classes cannot be examined due to sparseness issues. Merging several corpora into one would be a theoretic option, but it bears the risk of introducing undesired artifacts that will cause corpus features to overlay the actual age/gender features, as a result of differences in recording hardware, environment, situation, and post-processing. In addition, quality can also be an issue: On the one hand, the recording format of these older corpora is not always on par with today’s standards, which manifests itself in a higher amount of static noise, varying loudness, or a dynamic DC offset. On the other hand, natural background and channel noise may be missing if the recording was made in a “sterile” studio environment (e.g. in case of *TIMIT*), so the quality could even be too high in this regard, i.e. not resemble realistic conditions.

4.1.2 Data Sets

Choosing appropriate data sets is almost as important as having suitable data in first place. The following factors determine the selection of data sets from the whole corpus:

Processing Stages For each stage in the process that works with source data, it must be determined if an existing data set should be re-used or if a new data-set is needed. One case in which a new set should be chosen is when a classification meta layer is introduced, e.g. when optimization (tuning) takes place. A set re-use would correspond to 100% overlap, which leads to the next factor.

Overlap/Disjunctiveness Where data independence is required, an overlap of data must be avoided. There are several degrees to which this concept can be applied. For the FRISC experiments, we distinguish three cases: (1) Non-disjunctive: Any overlap is allowed. (2) Sample-disjunctive: Samples used in one set cannot occur in the other set. (3) Speaker-disjunctive: If a set contains one or more samples from a given speaker, no other samples from that same speaker can occur in the other set.

Volume The amount of data in each set must be chosen cautiously. In general, the more data we include in a set, the better the results. In most cases, data selected for one set cannot be re-used in another data set (see above). Enlarging one set will automatically shrink all other sets, so the sets will “compete” with each other for data. Hence, it has to be considered where data will have the most beneficial effect, and what the minimal size for each set has to be in order to function.

Balance No target class should be over-represented. The amount of material should be approximately equal for all classes and groups. Any bias introduced in the classes can cause a bias of the resulting classifier. Even if such a bias is desired, for instance due to different prior probabilities, controlling it through data selection is less transparent than adjustment of the probabilistic component of the model.

Variance The data has to contain all relevant cases with respect to variables that are orthogonal to the class-dependent effects, i.e. represent an adequately large variation of features. For example, if we choose our training set only from speakers from one geographic region, but apply the classifier to data from different regions, we might see a decreased performance due to missing regional variance in the data.

Reproducibility In multiple stages of the data selection process where an automated selection of items occurs, the items are selected randomly in order not to prefer some pattern or to adapt to structures from the original data. This random selection should however work in a way that identical parameters produce identical sets, in order to make evaluation runs more comparable. This is also called the “randomly but not arbitrarily” guideline. It can for instance be achieved by initializing random number generation algorithms with a constant so-called *seed* value.

Table 4.2 lists the five data sets that have been created for the different stages in the FRISC experimental set-up. The universal background model training set (*ubm*), as the name suggests, is used to train the UBM. It contains a class-balanced selection of speakers, with roughly the same amount of material per speaker, even though the model blends everything into just a single class. In total, 36% of the corpus or 40% of the labeled data have been exploited for this set. The training set (*train*) is the set on which the training of the SVM is performed after MAP adaptation of the UBM. In the final configuration, it is equal to the *ubm* set. Early experiments have shown that the amount of data for *ubm* and *train* is more crucial than their disjunctiveness. The latter property is not critical in case of these two sets because they do not build on each other in terms of decision making. In fact it is quite common to use “as much as possible” material for a UBM (although it is still strongly discouraged to use material on which numbers are reported). The first development test set (*test1*) is applied in the score optimization process of the SVM. Using this data, the untuned SVM is evaluated and the optimal decision threshold is obtained with respect to error or accuracy. The set comprises 27% of the corpus (30% of labeled data) and is balanced by speakers per target class and again by frames per speaker (the latter is done on the basis of an approximation). The second development test set (*test2*) is what most numbers in the main experiment series are reported on. Technically, it is another test set because the series itself is an optimization process, this time in terms of various global parameter configurations. It has the same size and balancing applied as *test1*. The evaluation set (*eval*) consists of 10% of the corpus balanced by class. It represents a special case, because its labels were never available to the author at any time, and its unlabeled data not until after the experimentation phase of the project was complete. This strict separation is a core concept of the *NIST* “one-shot” evaluation paradigm, and it ensures that no unorthodox tuning of scores is performed on the evaluation set, which was maintained safely by a third party. This is also why most figures presented in this thesis do not show data from the *eval* set, but rather from *test2*. Further technical details on the balancing of speech material are provided in Section 5.5.3.

4.1.3 Overview of the GMM-SVM-Supervector Concept

The GMM-SVM supervector approach was adopted from speaker verification (Kharroubi, Petrovska-Delacretaz, & Chollet, 2001; Campbell, Sturim, Reynolds, & Solomonoff, 2006) and first applied to the problem of speaker age recognition by Bocklet et al. (2008). It combines the strengths of the generative Gaussian mixture model adapted from a universal background model (GMM-UBM) and the discriminative power of a support vector machine as margin-maximizing discriminative method. It is particularly the MFCC-based features that seem to be much more fruitful when presented with this combination of algorithms. The GMM-UBM was already quite successful as a classification method in speaker recognition, but its strong dependency on a probabilistic model seems to prevent it from achieving maximum accuracy for MFCCs. On the other hand, the raw MFCC coefficients could not be successfully employed in SVMs directly as they are too uninformative. One could say that the GMM can successfully

Name	Description	Share
ubm	Training material for the universal background model.	36%
train	Material used to train the SVM based on the utterance-specific adaptation of the UBM.	36%
test1	Material used for score-level optimization of the SVM.	27%
test2	Data set on which the results of the parameter space exploration are reported.	27%
eval	Unlabeled, "one-shot" evaluation set.	10%

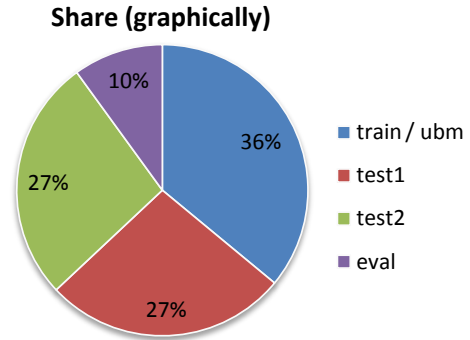


Table 4.2: Data sets which are used in the FRISC system, including the amount of corpus data they consist of in the main experiment series. Percentages are given relative to the whole corpus. Data is overlapping for the *ubm* and *train* sets.

reduce the structural complexity of the problem at the cost of increased dimensionality, while the SVM is good at solving the high-dimensional but structurally simpler problem.

Figure 4.2 depicts the general procedure. On a background data set, which contains data from all target classes², a GMM called the universal background model (UBM) is trained with the frame-based features. This GMM is not very useful as a classifier by itself, it is merely a model of the feature space for the whole (open or closed) world. The number of mixtures has to be set to accommodate the complexity of the feature space and the number of available frames. For each input *utterance*, another GMM is trained with the same features and the same number of mixtures. It is created by modifying the mixtures of the background model according to the data present in the frames for the current input utterance (MAP adaptation). The newly created GMM contains thus a portion of the background model and a portion of the utterance model. (Instead of utterances, another boundary can be chosen, such as speakers or even target classes, but this is a less realistic unit for evaluation and also requires more evaluation data. On the other hand, in a real-time scenario, there is no conceptual requirement to collect data from a complete turn or utterance.) Again, the utterance-specific GMMs are not evaluated at any time; instead, their means and optionally variances are the components that make up the feature vector of the next classification layer. This vector is created by simply stacking together the means of all Gaussians for all features in a fixed order, so that the vector, also called supervector, according to the way it is composed, has a size of $n \times m$ (n = number of Gaussians, m = number of features). Consequently, there will be one supervector for each of the original voice samples. With the vectors, a SVM is trained, which uses the original target classes as labels (which correspond to the label of the utterance that produced the vector, see Figure 2.20 on page 63). As originally, a SVM is a dichotomizer, some technique must be applied to enable discrimination between more than two classes. Testing a speech sample then works as follows: After the same frame-based

²for open-world setups even from classes that are not included in the target scheme

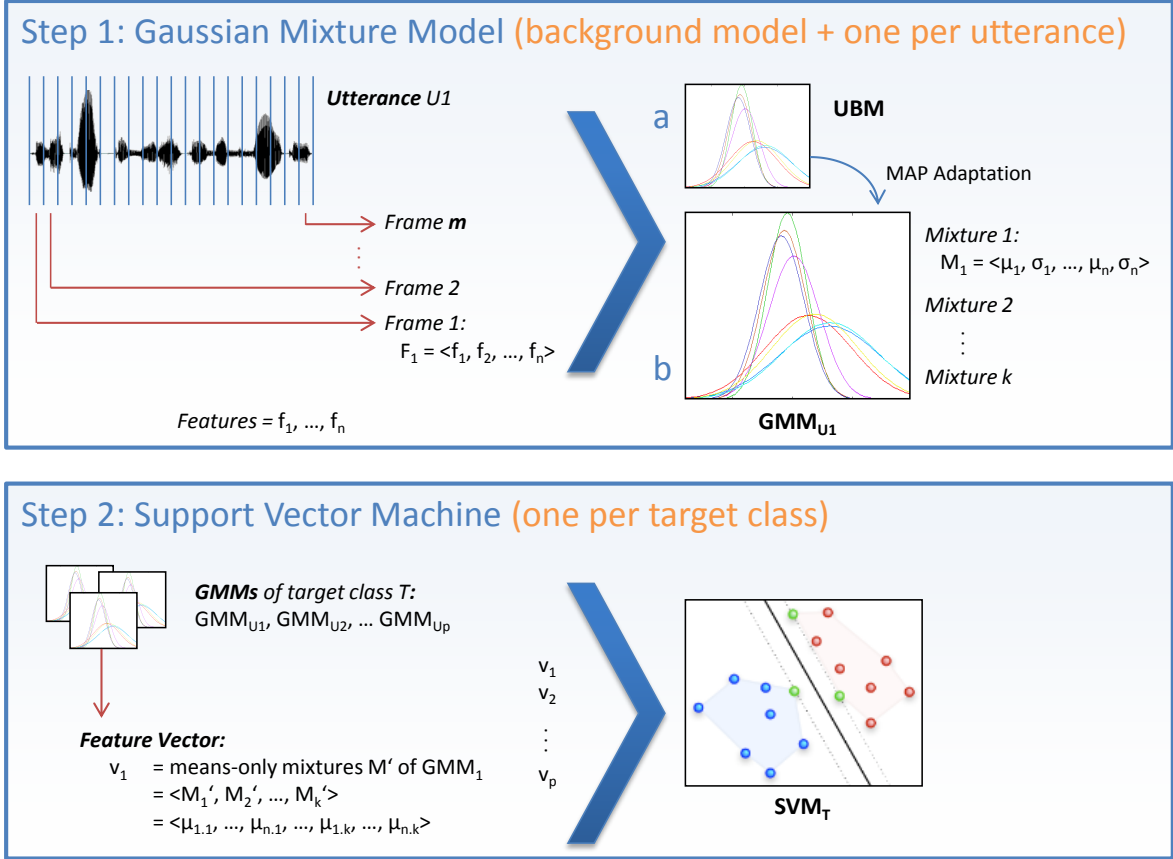


Figure 4.2: General concept of the GMM-SVM supervector approach. In step 1, the GMMs are trained for UBM (1a) and individual utterances (1b). In step 2, vectors for SVM training are generated from the utterance-based GMMs.

features are extracted from the sample as during the training case, a single GMM is MAP-adapted in equal manner as before. The supervector is constructed by stacking together the means/variances, and it is classified with the SVM. The SVM result corresponds to the final class assignment of the input sample.

Figure 4.3 illustrates the FRISC-specific adoption of the GMM-SVM supervector concept, including where the different data sets come into play and also including the linking parts. By and large, it adheres to the description given for the general case. The frame-based features, i.e. the MFCCs, are extracted directly from all audio files in the corpus. The result is a feature table containing all frames from the corpus. Certain frames with little or no speech are removed from this table (see Section 4.1.5). Afterwards, the various data sets are created by partitioning the labeled data according to the volume and balancing rules as specified earlier. The UBM is trained from all frames in the *ubm* set. For all other data sets, the utterance GMMs are generated by MAP-adapting the UBM using the frames belonging to an utterance in the corresponding sets. All utterance GMMs are exported to vectors of

stacked means, which are stored intermediately in mean vector feature tables, with each row corresponding to one utterance. The SVM is trained on the feature table from the *train* set. Additionally, the vectors in this set are saved to perform normalization later on. The supervectors in the *test1* and *test2* set are rank-normalized prior to being classified by the SVM. The scores from the *test1* set are used in conjunction with the known truth for score optimization (see Section 4.1.7). Finally, score optimization is applied to the scores in *test2*, which are also the scores used to compute the performance metrics such as accuracy and error rates. The procedure for the *eval* set is mostly identical to that for *test2*, merely the collection of statistics is performed separately.

4.1.4 Frame-Based Features vs. Global Features

Voice features can change very radically in even small periods of time. This is how words are produced by the speech apparatus. At the same time, other voice characteristics are preserved and change only slowly over time. In other words, different time intervals expose different attributes. If our intention is to recognize linguistic information, i.e. phonemes or words, small time intervals will have to be used to capture these phenomena. If we are rather interested in paralinguistic information such as speaker properties, long segments should also be considered. The left side of Figure 4.4 illustrates the extraction of the *long-term features* on the example of the mean pitch. From the raw waveform data, a pitch contour is computed, which describes points where a clear frequency was detected. The mean pitch, which is the value used in the final feature vector, is indicated by the blue line in the chart, and is simply an aggregate over time. In Müller (2005), only the long-term features were considered. The features that were found to contain age and gender information, such as the averages of pitch, jitter and shimmer, are indeed measured over longer periods of time. The empirical results confirm this observation. At the same time, there is sufficient evidence (see Section 2.1.3) to believe that *short-term features* can also tell us much about the speaker – if not more, as they are also considering dynamic speaker-specific aspects. Nevertheless, trying to do a frame-wise *evaluation* of features such as pitch sounds far less promising, since the discrete observations are dominated by the linguistic and functional (e.g. declarative vs. interrogative) contents of the message as opposed to the speaker. Additionally, they are only defined in voiced parts of the waveform (see the pitch contour in Figure 4.4).

Speaking of long-term as “global” features emphasizes the fact that some sort of aggregation over larger segments – typically “whole utterances” or turns – occurs. Yet, an utterance can be long or short, so they could theoretically convey short-term information as well. It is rather a common understanding that utterances will be composed of full words or sentences and generally are in the multi-seconds range or above. Shriberg (2007) also uses the term higher-level features, and defines them as features computed on more than a single frame. The foundation for global features can still be frame-based information. Common aggregation methods (sometimes called *functionals*) are the arithmetical mean, minimum, maximum, standard deviation, median, n -percent quantile, or average slope. Because global features

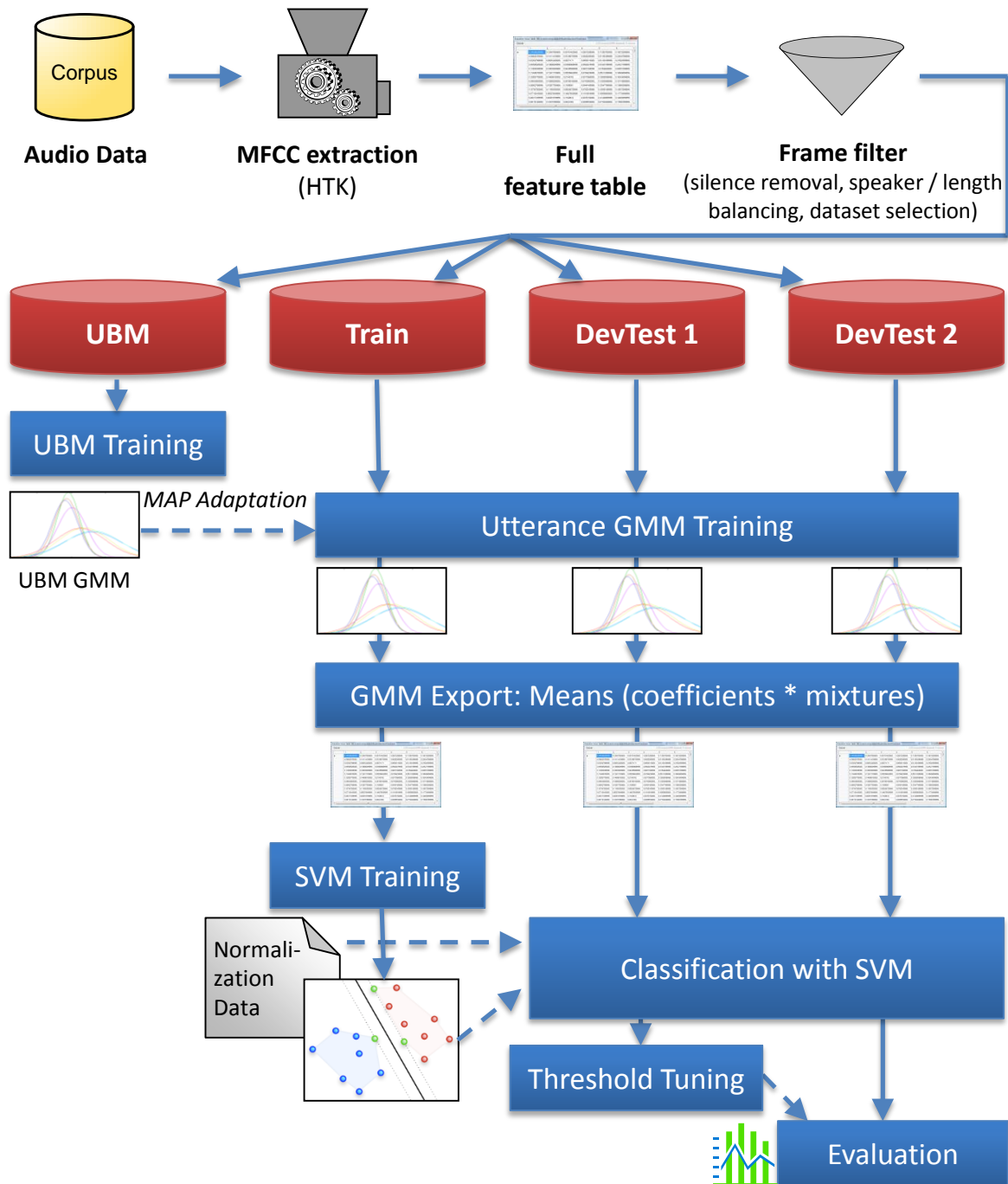


Figure 4.3: Overview of the GMM-SVM supervector approach as implemented for FRISC.

depend on the length of the sample on which they are computed, not all features can be handled in this way, and not all sample lengths should be considered for each feature, since that introduces high variation, or prevents a meaningful feature value to be computed at all.

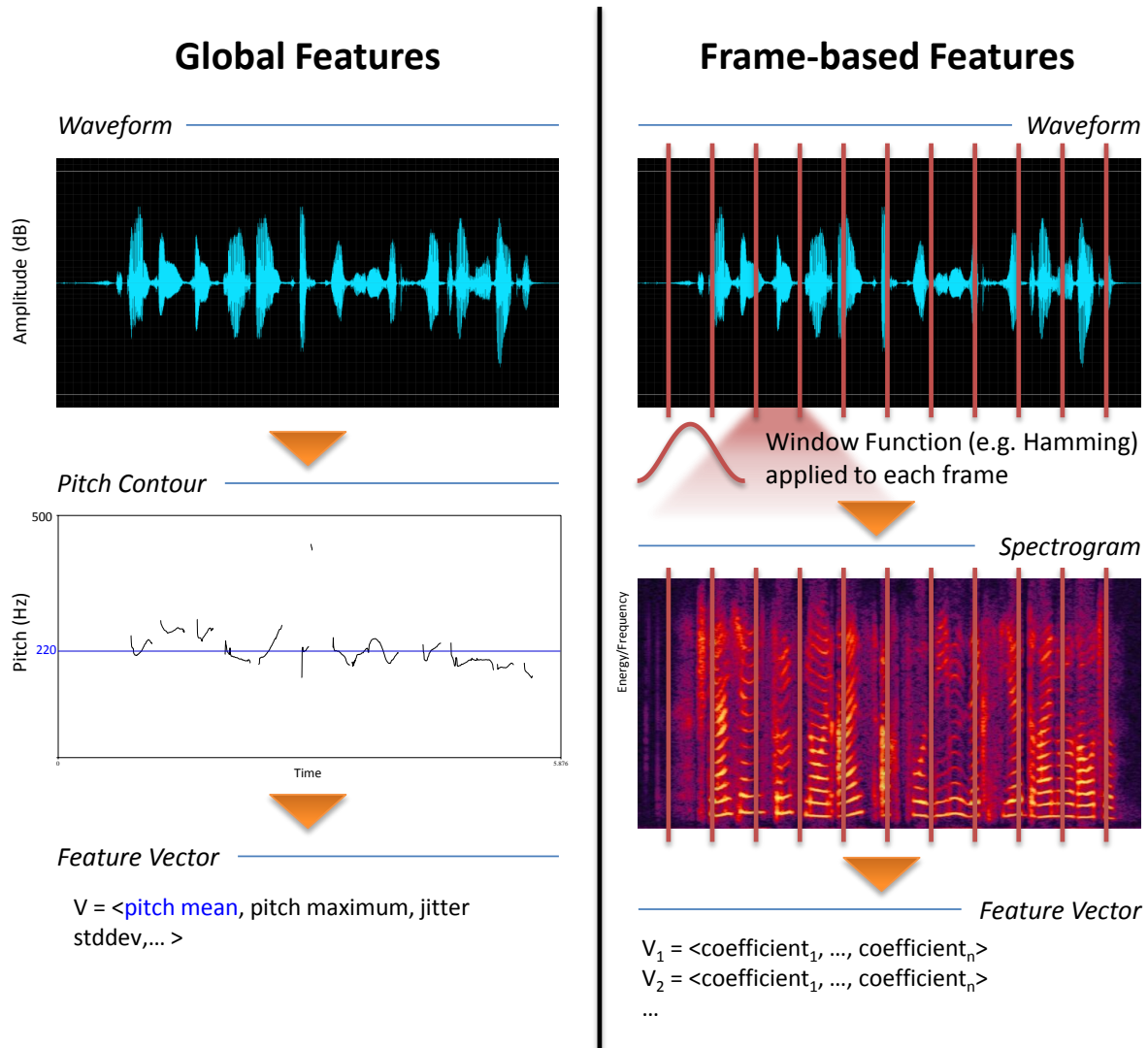


Figure 4.4: Extraction of long-term (average pitch) and short-term (MFCC coefficients) features on an utterance of approximately six seconds.

In frame-based processing, each frame represents one or more features computed on a small segment (also called *window*) of particular location and length. As can be seen on the right side of Figure 4.4, the original waveform is split up in windows of equal size (*window size* or *frame length*). Formally, the term “frame-based” does not imply a certain length, so choosing a frame length of several seconds would rather expose the long-term features (although it would not have a good exploitation rate of the available material and would not work at all

for short utterances). In the context of this work, and in many other papers in the speech community as well, frames are based on windows in the milliseconds-range. In the depicted example, the windows are exactly adjacent, which means that the *step width* (or *interval length*) is equal to the frame length. This needs not be the case; in practice, overlapping windows (i.e. the step width is smaller than the frame length) are used quite often to smooth the sampling effects. The opposite can be beneficial to avoid the number of frames growing too large. Finally, a *window function* is applied to reduce artifacts at the edges of the window. There are a number of common window functions optimized for different purposes (e.g. Hanning, Hamming, Gauss, Blackman, etc.). For the MFCC extraction in FRISC, a Hamming window was applied. MFCCs are based on the time-sensitive frequency domain of the utterance, which corresponds to the spectrogram visualization in Figure 4.4. (Note that the similarities between the spectrogram and the pitch contour are not coincidence; they both model aspects of the frequency domain.) Even if the illustration portrays a small frame width, the information is already sampled into frames. From each frame, multiple MFCC coefficients (the final features) are then derived. The short-term features evaluated for FRISC, such as MFCC coefficients, do not aid in the pattern recognition process when averaged over whole utterances, since they lack any interpretation when used in a long-term context, even though it would be technically possible to do so. Enhanced with the GMM on a per-frame basis to cover a wider feature space however they are very useful, even though the means vector in the end also represents some aggregation.

From the perspective of the implementation of feature extraction, there is not a big difference between frame-based and global features. A global feature can be extracted in the same way as if a single frame of the size of the whole utterance was used, or it can be an aggregate value over several frames. The main difference, apart from their information contents, lies in the way the learning algorithms deal with the extracted features. With global features, we know already in advance that exactly a single value for each feature will be computed for every input sample. Even when some values are “missing” for a concrete sample, they can be substituted by a placeholder, and the overall number and ordering of features remains unchanged. This is a basic requirement of most pattern recognition approaches. In the frame-based case, the number of features for one input sample corresponds to the number of features per frame times the number of frames in the input sample. This means that not only can the feature vector for a single utterance become very large, but it also loses a fixed structure when dealing with voice segments of variable length. One way to deal with this is to use only subsegments of a fixed length, but that has the major disadvantage of disregarding a large amount of potentially useful information. Another option would be to apply *dynamic time warping* (Sakoe & Chiba, 1978) to map the features to a fixed-size vector, but that would result in a loss of any temporal information and is not necessarily the best way of dealing with speech frames. The GMM-SVM supervector approach chosen for FRISC makes use of the third option, which is the utilization of meta-learning techniques to separate the frame-wise classification from the utterance-wise processing.

There is another practical advantage of frame-based processing. Global features require

Feature	Type	Main Experiment Series	Fusion Experiment	Corpus Analysis
MFCC	frame-based	•		
MFCC Deltas	frame-based	•		
Pitch	global		•	•
Jitter	global		•	•
Shimmer	global		•	•
Intensity	global		•	
Formants F1-F4	global		•	

Table 4.3: Summary of features used in one or multiple of the experiments in this work. The features are described in Section 2.4.4.

boundaries to be specified for each utterance, and which cannot be simply guessed because they cover a long time span. Frames on the other hand are always extracted in fixed intervals, so the duration of an utterance does not have to be known. Moreover, assuming the appropriate algorithms are used, classifying frames can theoretically start immediately (or one window length) after the first data arrives, and then incrementally refine the result with every frame. This enables the use of frame-based classification in situations that need immediate response, a property that global features cannot offer.

From a machine learning perspective, the ultimate goal should be to combine both frame-based and global features in a single classifier using fusion and meta-learning methods. While this has also been done, the focus of this work will be indeed on the aspects of the frame-based features, because of the exhaustive research on long-term features in prior work.

Table 4.3 lists all feature families that were considered for FRISC, including the long-term features. The latter, although not focus of this work, are still relevant for some sections of this thesis. In Section 4.5, they are used as part of a corpus analysis study, something for which they are very suitable because of their strong interpretability. In Section 4.4, they have been used in a combination system consisting of both short and long term features. A description of the features can be reviewed in Section 2.4.4.

In FRISC, all MFCC features were extracted using the *Hidden Markov Model Toolkit* (HTK). More precisely, the MFCCs are computed using the utility *HCopy* and parsed using the utility *HList* provided in the toolkit. A description of the process can be found in Young et al. (1999, ch. 5.2). In the configuration used, *HCopy* initially applies a first order pre-emphasis coefficient of 0.97, which modifies the raw signal according to the formula

$$s'_n = s_n - 0.97 \cdot s_{n-1}$$

Afterwards, the following Hamming window function performs smoothing at the edges of the signal:

$$s''_n = \left\{ 0.54 - 0.46 \cdot \cos \left(\frac{2\pi(n-1)}{N-1} \right) \right\} s'_n$$

Energy normalization and DC offset removal on the source signal are disabled in *HTK* to allow incremental (on-line) processing of audio. *HCoppy* then employs a fast Fourier transformation to derive the MFCC from the logarithmic spectral analysis. It does so by applying a filter bank with a certain number of channels (usually 20). Again, due to its restriction to off-line data, cepstral mean normalization is not applied in FRISC.

4.1.5 Silence Filtering

The input features that are obtained from an utterance are computed for every frame initially, with the number of frames being in a constant ratio to the length of the utterance. Therefore, the MFCC features are also computed in the same way on those parts of the sample that contain only little or no voice at all. Figure 4.5 illustrates this. Regions without speech are called silence regions. Frames with less than a certain amount of speech fall into these regions as well. It appears obvious that the task of classifying a speaker cannot be improved by the information from frames that contain no voice from that speaker at all. Including these frames anyway adds complexity to the decision problem and bears the risk of reducing the classification performance. The purpose of frame filtering is to remove all frames with a silence “level” above a certain threshold $T_{silence}$. There are also frames based on windows that only partially contain silence. It is difficult to hypothesize on the effect of silence in these cases, especially with respect to the threshold of when frames are filtered out. Our hypothesis is that with increasing threshold x for silence filtering, i.e. with more frames being removed, the result will first improve (due to more interference eliminated) and at some maximum start to degrade again (due to more useful information also removed). The question to be answered experimentally is which threshold $T_{silence}$ to choose.

This is not the only question. In fact the problem of finding a threshold $T_{silence}$ of silence vs. speech in a frame is underspecified because there is no universal idea of what “silence” means. There exist several definitions though, which employ power thresholds, signal-to-noise ratio (A. Davis, Nordholm, & Togneri, 2006), or spectral energy (e.g. the G.729 standard). To avoid introducing further parametrized methods, a simple metric based on the amplitude of the signal was chosen: A frame is considered a silent frame if the average absolute amplitude of the signal is below a certain threshold $T'_{silence}$. This metric is not robust enough for scenarios where a high recall is desired, but it reflects a compromise between complexity and soundness. In the experiments, $T'_{silence}$ was rarely specified explicitly, since it is not easy to interpret. Instead, the threshold $T''_{silence}$ specifying the percentage of frames to filter out was employed. Determining the threshold $T'_{silence}$ is then a three-phase process: In the first phase, the energy metric is computed for each frame in the corpus. Then, the values are sorted and the one is picked so that $T''_{silence}$ percent of frames are smaller (i.e. the $T''_{silence}$ % quantile). Finally, during the experiment, all frames below this threshold $T'_{silence}$ were removed.

Following a similar argument, noise is an interference factor that should be removed by eliminating frames that contain more than a certain amount of noise. Unlike silence, noise is not exclusively occurring interchangeably with speech, i.e. detected on the time line, but

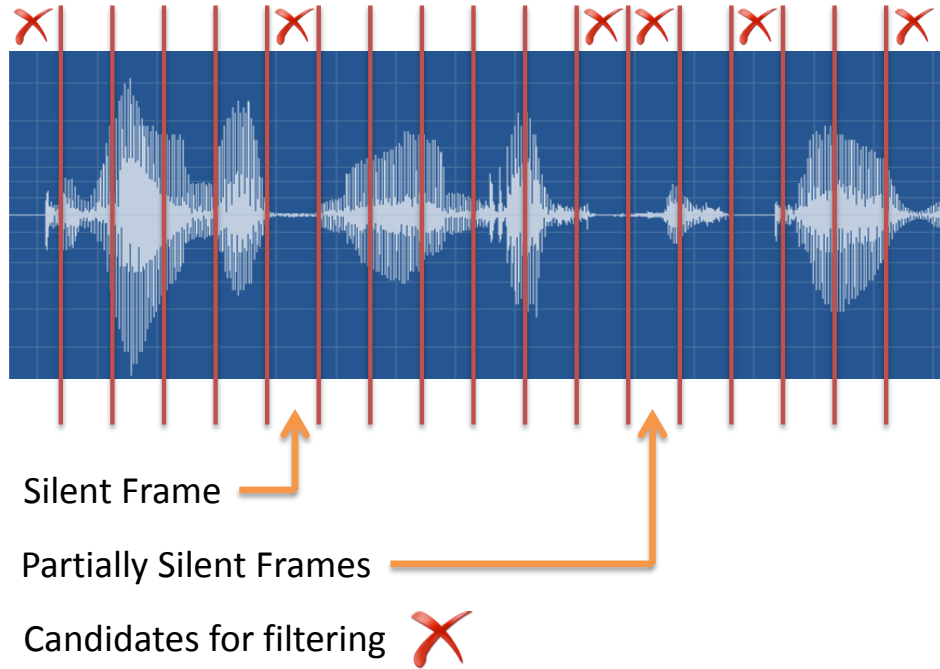


Figure 4.5: Using silence filtering, frames with less than $T_{silence}$ percent of speech are considered silence and removed.

concurrently, so it can also overlap speech regions. Removing noise by filtering frames would be possible, but there is a higher risk of filtering either too many frames or none at all. Especially in the automotive domain, there are different levels of background noise from the wheels, engine, and other mechanical parts to expected depending on the velocity and road surface. Literature suggests that noisy conditions are generally dealt with using pre-processing methods such as frequency band filters or advanced noise suppression techniques. Researching the effect of noise is a special topic in all areas of speech research, and a lot of literature can be found on it. However, it was not found a pressing issue more relevant for FRISC than for these other areas, so it is not considered for these experiments. In addition, the main research corpus did not exhibit a notable amount of noise, so that only artificial noise would have been an option. It should be noted that the architecture can flexibly integrate upstream noise filtering methods if available.

4.1.6 Feature Domain Optimization

The feature space of the GMM supervector is a high-dimensional one, and this leads to the question in how far all of the features in it are helpful for our task. This general question has also been sighted in other work before (Porat et al., 2010). In fact it is at least difficult if not impossible to make an educated statement about an individual feature such as “the mean of the 5th MFCC delta coefficient in the 43rd Gaussian”, without solely relying on pattern

recognition results. In particular, there is no way of knowing what the 43rd Gaussian models without looking at the internals of the training process, e.g. the distribution of values. We may assume that many of the MFCC-based features carry speaker-related information, but at the same time, it seems equally likely that not all of them do. It is well-known that, depending on the pattern recognition scheme used, features that do not carry information that is correlated to the target class in any way, can degrade the result of the learning process, either due to non-exhaustive learning or due to artifacts in the training or test data. Hence, only an algorithm considering all degrees of contributions of all subsets of features separately would have a chance of finding and filtering out those *nuisance* or *noisy* features. In practice, this rarely happens due to the computational infeasibility. For instance, the GMM learning process begins with a more or less pseudo-random initialization of Gaussians, which assumes equal importance of all features, and then applies the EM algorithm with a fixed number of iterations. Unless there is strong contrary evidence in the data, the feature present in the final model to some degree even after the last EM iteration.

Not only features with a negative impact on classification performance, but also features with a low positive impact can be nuisance attributes. Every feature that is respected in the decision process is also a possible source of errors, due to the uncertainties with respect to the classification problem itself (i.e. not all candidates exhibiting the same behavior for a feature), due to the unreliabilities associated with the computation of the feature, and the determination of its impact. Each such feature also adds complexity to the decision boundaries of the classifier, easily resulting in overfitting if the number of features with a low relative importance is too high, thereby decreasing the efficiency of the classifier. In addition, some algorithms work in a way that limits the total number of respected features, e.g. a decision tree that is pruned one or more times during the training, as it would otherwise result in unreasonable complexity and training duration. Similar to the GMM example from the previous paragraph, the pruning is also merely an approximation, and in some adverse constellations, the inclusion of further weak features can result in a removal of other actually stronger features. For these reasons, weak features are also unwanted features to some degree. This includes groups of features with a high correlation, i.e. containing almost the same information.

How can nuisance features be detected and removed? Obviously, each classification algorithm already has its own means for determining the importance of input features (even if it merely states that all features have equal impact). Any additional mechanics for nuisance attribute handling must occur independently from the training prior to the SVM training process on the supervector data. Once the nuisance attributes are identified, the corresponding features need to be consistently removed from all instances before they are forwarded to the SVM, so the SVM will never see those features.

In more general terms, at this point we are dealing with a problem of *feature selection* (also known by the terms *feature (space) reduction*, *feature filtering*, or *feature domain optimization*). There are two basic ways of nuisance attribute detection: heuristical and through optimization on a test set. The main difference is that the heuristic approach does not need

to apply the downstream classifier to measure actual impact of a feature set; it merely applies a heuristic to the feature selection that is supposed to improve the classification result as well. This can be considerably faster and becomes more suitable the larger the feature space grows and the lengthier the classifier training process gets. In some scenarios, it is the only practical option, even though it is typically less accurate (although a good heuristic can work better than an exact measurement if the number of samples is small).

One of the simple yet commonly used methods for feature selection is principal component analysis (PCA). Having been introduced at the beginning of last century, the method is quite mature and has been applied successfully in many different pattern recognition tasks. Some methods particularly related to the reduction of age-related supervector features have been collected by Dobry, Hecht, Avigal, and Zigel (2009). The topic is also closely related to *nuisance variability compensation* (NVC), a term which is used most often in conjunction with attempts to compensate nuisance features or feature modifications that are caused by the channel (such as noise, distortion, or frequency suppression introduced by mobile carriers).

The methods mentioned fall into the second category of feature selection methods, which means that they function in an optimizing manner and work best when run in conjunction with the classifier. For the FRISC dataset and feature space, they were not considered because even a few iterations would have consumed too much time. Instead, a heuristic method was needed. Our solution was to use the statistical *variance ratio*. It is based on a simple hypothesis: For every feature, the greater the variance of the feature's values across the target classes is versus its variance of values within a class, the more suitable it is as a discriminator for this class. What makes this a heuristic (as opposed to a rule) is, among other things, that the variance is neither the only nor a truly reliable characteristic that determines whether the feature is suitable as a decision criterion, but it is often a good indicator. To formalize this hypothesis, the average variance within any target class (var_{wi}) for a feature f and class c is defined as

$$var_{wi}(f, c) = \sum_{i=1}^{|inst_c|} \frac{(x_{i,f} - \mu)^2}{|inst_c|}$$

where $inst_c$ is the set of instances in class c , while the variance across target classes (var_{ac}) is defined as

$$var_{ac}(f) = \sum_{i=1}^{|\bigcup inst|} \frac{(x_{i,f} - \mu)^2}{|\bigcup inst|}$$

Finally, the variance ratio R is defined as

$$R(f, c) = \frac{var_{ac}(f)}{var_{wi}(f, c)}$$

If var_{wi} and var_{ac} are equal, the ratio is 1. The ratio can become larger or smaller than 1 if the balance shifts in the corresponding direction. With the between-classes variance growing larger than the within-class variance, so is the ratio R , and vice versa.

How is the final list of features to be filtered determined? This problem corresponds to the one of selecting a threshold for silence filtering Section 4.1.5. One option would consist in

selecting a fixed threshold for R for which a motivation exists, e.g. that resembles some minimum variation discrepancy. On the other hand, the absolute number of features is a relevant output parameter of the selection process, and it would be disregarded by just considering isolated threshold values. Instead, the decision follows the process for frame filtering: The threshold is automatically determined in a way that when applied, a certain amount of features remains. This amount can be chosen in a relative or absolute way. Choosing a relative value (e.g. removal of $n\%$ of the features) is generally easier and more intuitive, but both methods may be disadvantageous in some situations. A relative value decision emphasizes the detection and avoidance of noise in the data, while an absolute specification emphasizes the maximum size of an effective feature set. In the FRISC experiments, between 35 and 100% of the features were kept.

4.1.7 Score-Level Optimization

In Section 2.5.7, the basics of classifier scores have been explained. A true binary classifier, which was appropriate for a yes-no question (e.g. “Is this voice from a senior speaker?”) or a detection task, would always return one of the two possible values, yes or no. A true multi-label classifier, which is an extension of this concept, would for every input return exactly one of the class labels it was trained with, namely the one which it predicts for the input, making it ideal for an identification task. In practice, the pattern recognition algorithms which are employed often do not adhere to these criteria. Instead, they usually return a numerical value (or multiple for multi-label classifiers), a score, with the side effect or purpose to provide more insight into the decision process. This is especially the case with generative models such as the GMM, which provides the likelihood ratio as a measure of agreement rather than a binary decision. That is, however, not a disadvantage, but allows to decouple the estimation from the nominal mapping aspect. With only the nominal decision, there would be for example no way to know how “close” the decision was or how confident the classifier is in it. When the output value of a pattern recognition algorithm does not have a decision implied, it makes classification a two-step task: The estimation, i.e. the computation of the initial score, is the first step, and the mapping of the numerical score onto a class decision is a second step. Those two steps can be attended completely independently, the second actually being a form of meta classification (although normally a very simple, linear one). As such, it also has to be provided with its own set of training data – a development test set – to differentiate it from the main classifier’s training set (see Table 4.2 on page 107 for the configuration of this set in FRISC). The effect of the second part can be critical for the classification performance. If there is any “default” action to be performed in the absence of score mapping for binary classifiers, it would be to assume 0 as a decision threshold, and to map positive values to *yes* and all other values to *no*. Obviously, many classifiers, such as the GMM, do not make any implications on their scores, so the decision threshold of 0 is arbitrary unless some evidence indicating the opposite exists through insight into the score computation process. For this reason, score-level optimization is essential, but there is another reason why it is often used

for tuning.

If we assume that a classifier used in the first place puts at least some meaning in its scores, it would be possible to accept a default threshold of zero. It would even be plausible that the classifier did some validation during its training process to ensure that this threshold approximates an optimum. Still, without thorough study of the algorithm, it is not clear what the goal is that the model is optimized for. There are several different performance metrics that can be used to define a good classification process, and there may be further restrictions and external knowledge known as cost of errors (e.g. “not detecting a senior is less critical than mistaking a teenager for a senior”, or vice versa) and a-priori knowledge (e.g. “there are more in-class samples than non-class samples in the data”). The classifier, unless it was trained with special parameters, does not reflect such goals. But since the algorithm’s scores retain the flexibility for an adjustment of classification boundaries, the score-level classification step provides a means to complement this goal, which is why the step is also often called score-level optimization or tuning.

Since we are dealing with an optimization problem at the core, the performance metric that is optimized (i.e. the cost function) has an impact on the algorithm used for the mapping. One of the main performance metrics used in the FRISC evaluation is the detection cost function (see Section 2.5.7). Minimizing this function corresponds to choosing an operating point on the DET curve, i.e. a trade-off of misses versus false alarms, which fits the requirements of the application best. More concretely, for a single-class task, the goal is to find a decision threshold Θ so that the DCF is minimal. For a multi-class task, one threshold Θ_c is to be found for each class. For the detection task, the overall DCF score is based on the average scores from the individual classes, so each class can be optimized separately to get the global optimum. In turn this means that if the global optimum is found, it can be assumed that the individual classes also perform best with respect to the detection metric. The absence of inter-class relations is an advantage, since less computational effort is required to get better results. In fact, the FRISC implementation of the detection task score tuning always finds the optimum. The DFC used in these experiments always defines the parameters $C_{Miss} = C_{FA} = 1$ and sets P_{Target} to $1/C$ for C classes (which roughly corresponds to the actual distribution).

While there are many advantages to the DCF metric, there are some cases in multi-class scenarios where a different metric is desired. These are mainly those scenarios resembling an identification task. A common metric representing this family is the accuracy (see Section 2.5.7). As opposed to the DCF, the accuracy focuses on the performance of the whole system as opposed to the individual classes. With the DCF, a real multi-class decision never occurs. This is different here: A smart way of combining the scores of multiple classes in a final decision does not impact the error rates, but it impacts the accuracy. Also, this metric is often applied to compare different systems, since it is the most intuitive choice for many applications. A score tuning algorithm for accuracy has been implemented for FRISC based on an EM-method. One major technical difference between the DCF and accuracy tuning methods is that the class-specific parameters in the function used for the tuning of accu-

racy are not independent, so the optimum cannot easily be retrieved. This also means that whether the change of a single threshold has a positive or negative effect on the accuracy depends on the other thresholds, and it is not always a linear relationship.

Identification task optimization can be useful for comparing upper boundaries, i.e. the “best” version of a system. In practice, when used as the decisive metric for tuning, it yields a disadvantage that should not be neglected: Since only the multi-class performance is optimized, the performance of individual classes may suffer. What has been observed several times during the experiments is that the performance of a single class degrades enormously to almost zero recognition rate, while most other classes improve slightly. Even worse, a very small improvement in the accuracy can sometimes have a radical change in the class error rates. This type of artifact was often witnessed in the accuracy-driven experiments. Unfortunately this behavior is difficult to control objectively. For this reason, the FRISC experiments report both metrics, but rely on error rates for parameter space exploration.

4.2 Experiments Overview

The following paragraphs give a short overview on the different experiment series that have been conducted in conjunction with FRISC.

The most extensive experiment used to evaluate the FRISC approach will be referred to as the **main experiment series**. This experiment was set up to confirm or refute hypotheses on the various advancements in the age and gender recognition process and to gain a better understanding of how different parameter combinations affect the result. This series follows a rigorous and consistent procedure, making the results directly comparable. While the results that were achieved in the end are not the best reported in this thesis, the main contribution lies in the investigation of the methods and parameters, and the relative improvements they enable. This experiment is presented in Section 4.3. It was also separately published in [Feld, Burkhardt, and Müller \(2010\)](#).

Following the main series in Section 4.4 will be reports of results on a modified version of the FRISC approach, which is actually a combination of the features from AGENDER and FRISC, plus some fine-tuning. This **hybrid approach evaluation** performs two types of late fusion: It combines frame-based and long-term features, as well as classification and regression methods. Not surprisingly, it yields the best absolute results. Although there is no equivalent in-depth experiment series available for this version, it mainly serves to confirm once more how far things can go when concepts are combined using known fusion methods. The prototype has been created and tested in collaboration with Meraka Institute in South Africa, and has been accepted as the best-performing implementation within these specifications in the industry project for which age and gender recognition was initially developed.

The studies on **multi-cultural aspects** of Speaker Classification are presented and discussed in Section 4.5. These experiments were performed on a different corpus (*Lwazi*) and therefore cannot be directly compared to the other experiments. They consist of corpus analysis and regression analysis methods.

Finally, a study on the performance of FRISC on **synthesized voices** in comparison with the human's ability to recognize the same is introduced in Section 4.6. Since this collaborative study had its focus on the phonetic aspects presented in different work, it is merely described superficially.

4.3 Experimental Investigation of System Parameters

The concepts that make up the FRISC approach and that were introduced in the previous sections of this chapter are based on literature studies, observations on the data and hypotheses derived from them. It remains yet to be confirmed whether they do support the goal of a more performing and more robust age and gender recognition. In pattern recognition, this confirmation can only be given by actual evaluation results from the implemented system. Because we are dealing with a very constrained scenario, e.g. specific speaker properties, a given recording environment etc., it is also to be expected that some hypothesis will be refuted by the data.

One purpose of this section is to detail the evaluation of the FRISC approach described in the previous sections. It will be interesting to see if the techniques introduced earlier have a positive effect on the numbers, and how large this effect is. A second aim is to give a summary of the high-level exploration and optimization of certain system parameters. Some of the parameters that were introduced earlier can only be tested to a very limited degree when separated from the rest of the system. An example is the amount of silence frames which is removed in the filtering phase of the approach. To measure the real effect of this parameter, a full evaluation run of the complete system with multiple parameter settings has to be performed.

In theory, it would be possible to evaluate all parts of the classification process separately. The usefulness of the resulting numbers would however not be very high, for several reasons: First, the number does not express the performance of Speaker Classification. It is for example possible to test how well scores can be mapped to classes by using a certain score tuning method, but this number is by itself not relevant to the problem at hand. Second, the effect when used in conjunction with other components is not considered. Even if a changed parameter appears to be an improvement according to some measure applied, it can still have no effect or even a degradation on the overall result. Third, for many parts of the process which are not classifiers, it is not clear what an expressive performance metric would be at all. Taking for example MFCC feature extraction separately, it is very difficult to express the quality of the resulting data without considering the classification that follows in the architecture³.

According to these reasons given, the more relevant and beneficial to the reader should generally be the numbers obtained by testing the whole FRISC approach. A number of insightful conclusions can only be drawn this way. If this is the intended procedure, it is

³There are other features for which a statistical analysis can provide some insight, see e.g. Section 4.5.2.

essential that only a single property is modified at a time, as the cross-effects of interdependent variables could aggravate with increasing complexity of the system. In practice, there is one more considerable drawback of this alternative. A single experiment covering the full FRISC system is composed of many computationally demanding tasks, both because of the algorithmic complexity and the volume of data involved. A single experiment would run more than 12 hours on average up to over a full day even on technically advanced hardware (see Section 4.3.1 for details). A reduction of training data was not considered an option because it has shown to affect the results on a scale that matters.

With such a magnitude arrives a number of related problems. First, the total number of experiments that can be performed is obviously limited. The experiments conducted in this series required over 20 days of processing (cumulative time according to Table 5.6 on page 208, only counting successful runs). This in turn affects the optimization procedure; it forbids the type of system parameter exploration that tests hundreds of settings for a single parameter to find the best since it would take months to complete. Secondly, any errors in the experiment configuration, such as a bad entry of settings, or an error during the execution (e.g. errors in the code, running out of memory or disk space, network disconnection), could often only be discovered and resolved on the next day. Unfortunately, with code that can only really be tested in the actual experiment, such error conditions occur more often than desired, and are the reason for much “lost” time. Third, it is normal that progress made on core parts of the system happens over time. Feature extraction routines are optimized, additional parameters are introduced, some concepts are changed. Obviously, during a series of experiments that should have comparable results, incorporating such evolutionary changes is not possible without restarting the complete series. The bottom line is that at some point, basic changes will be prohibitive. For more information on the technical details of the FRISC experimental setup including the individual share of time the components require and what measures were taken to optimize the runtime, see Section 5.5 and Section 5.6.

4.3.1 Experimental Set-up

Technically, the main experiment series is designed as a comparative, explorative optimization task. Due to the reasons mentioned earlier, the number of experiments was limited to 35 manually selected configurations. More precisely, the experiment series follows a guided *best-path* procedure, which is a type of hill-climbing applied to a fixed, discrete search space:

Initially, the list of parameters to be explored is selected and a fixed order is created. For reasons of clarity, this order should follow the point of occurrence of the parameter in the architecture. Next, for each parameter, the list of settings is defined and a reasonable starting value (i.e. default) is chosen (it should preferably be aimed for an “average” using the background knowledge available). This list can still be changed up to a certain point during the experiment should it become evident that further settings may provide a better insight or a closer approximation of the optimum. The experiment series begins with all parameters set to their defaults. Then, the experiment is run with all alternative values for

the first parameters, and the results for each are noted. When complete, the same procedure is repeated with the next parameter, while the setting of the first parameter now remains at the setting which accomplished the best result in the previous step.

This “greedy” strategy is admittedly suboptimal with respect to discovering the global maximum. It is possible that cross-dependency effects between parameters result in the tuning of one parameter to have an effect on the already confirmed setting for a previously evaluated parameter. It may also happen that one parameter has such a strong effect that it “hides” the effects of values upstream in the pattern recognition process when a bad setting is chosen, which encourages the cautious selection of defaults. For example, if a strongly counter-intuitive kernel function is picked for the SVM, then all possible settings for the MFCC extraction window length will produce equally mediocre results, preventing the identification of the optimal setting. Further, the choice of only a few settings for each parameter may cause the optimum of the single parameter to be missed as well. It is likely though that most parameters’ performance functions are continuous, so that either a good approximation is possible or the reverse will be revealed in the experiment.

The choice of the best-path strategy is a necessary compromise. Not being able to run an exhaustive search on the parameter space, its single big advantage is that it requires a small number of experiments. It is an explorative search in the absence of a goal value.

Some thought was put into the hardware set-up as well. There was a basic decision to make between using a distributed (i.e. cluster-based) or single-machine solution. Distributing load on different machines, possibly hundreds, is often done in other data-intensive experiments with a gain in speed and in turn an earlier completion of the experiment. In the FRISC evaluation, a single powerful machine with multiple cores has been utilized⁴. The reasons for this decision are manifold.

First, only some parts of the architecture can be parallelized within reasonable bounds. The MFCC feature extraction, the training of the utterance GMMs, or the training of different target SVMs are examples of acceptable candidates for concurrency. Other components, such as the training of a single SVM class or the UBM training, cannot be parallelized well due to the strong relations between data at the various stages of the training process. Section 5.5 has details on this. Having monolithic tasks that cannot well exploit the power of multiple machines or processors is a problem that plagues other application areas too. Even parallelizing speech recognition requires pretty smart engineering to be successful (Chong, Friedland, Janin, Morgan, & Oei, 2010). On the other hand, with a considerable investment in performance optimization, sometimes this can be changed by clever arrangement of shared data and synchronization, and in a few cases, it even results in a decisive improvement. For the task at hand, an increased attempt to optimize the process further than it was made, would have hardly resulted in a runtime performance gain justifying the effort.

Then, even if much of the work could be done in parallel by multiple machines, there is still the time needed to transfer the data from a shared network location to the machine

⁴Intel Core i7-940 with 4 cores at 2.93 GHz, 12 GB RAM, 2x 1 TB hard drives with 7200 RPM in RAID configuration, 64-bit Windows OS

and copy the results back to the storage. Since both ways convey massive amounts of data, even if work is split, this would introduce a new bottleneck. Using a network site for data storage was actually tried at some point during the experiment, and it resulted in more than a tripling of the experiment runtime. For the same reason, an Internet / cloud based solution was never an alternative either. It is a general observation that some parts of the system have very high requirements as for the data link bandwidth while others have less. Both the MFCC feature extraction as well as the training of the GMMs are examples of the first category, as both of them are working on frame-based data. A fast hard drive and a high amount of working memory for caching of disk files proved to be solid basis for the handling of these high-bandwidth tasks. They were complemented by optimized storage structures for intermediate results and feature data (see Section 5.3.3).

As a further point, even if some tasks could be parallelized in theory, the required resources would increase considerably. For instance, the SVM training of a single class requires a lot of memory, because each class is trained with all data. Thus, this memory would be required for all machines in the cluster, so a cluster of average machines would have almost certainly had a worse performance than the single powerful machine solution. Having such resources available cannot be taken for granted. Additionally, if a cluster was used for the parts that benefit from it, the single powerful machine would still have been required for the monolithic tasks. Apart from that, a complex workflow that involves combining cluster-based (parallel) with single-machine (sequential) processes, requires an additional orchestration overhead that poses both a challenge to engineering and robustness that does not pay off easily.

All experiments were set up using the SPEACLAP framework, which has been developed mainly for this purpose as part of this thesis, and which will be introduced in greater detail in Chapter 5. The whole main series of experiments is modeled as a single experiment design that exposes the parameters of interest as the experiment configuration (see Figure 5.8 on page 172). It is implemented as an SPEACLAP script connecting the various components as sub-tasks, including information about which of them can be parallelized. After an experiment is started, the framework gives a detailed overview on the current progress. When the experiment is finished, or if it is aborted due to an error, an email is generated with a complete report of the run. The report is also saved locally together with all evaluation results, so that the progress of the experiment series can be tracked. In consideration of the length of a single run, each experiment was started separately, which would also give an opportunity to review the results. A common practice was to start an experiment configuration in the evening and have it run over the night until later on the following day.

4.3.2 Results

Table 4.4 contains an overview of the results of the ten parameters evaluated in the FRISC main experiment series. For each parameter and setting, which corresponds to one line in the table and one evaluation run, the *accuracy* (i.e. classification performance in the identification task) and *uniform error rate* (i.e. classification performance in the detection

task) are reported. In addition, for some combinations, confusion matrices and DET plots are presented throughout this section. Note that the scale of these DET plots was shifted to center on the area where most changes occur. This is also supported by the fact that the expected performance (or baseline) of this task is lower than for the majority of classification problems for which a scale is used as in Figure 2.22 on page 68. The following section progresses through the table in the order of system components, recalling the definition of each parameter, summarizing the results and discussing them.

4.3.3 Discussion

Front End (Feature Extraction)

The front-end of the FRISC classification pipeline is the component that deals with the extraction of features from the raw audio signal, which are the frame-based MFCCs. The *step width* parameter describes the period between two sound samples and hence the sampling rate of the output feature vectors. A smaller step width results in more feature vectors. A step width of 10ms is common for GMM-SVM models, which is why it was chosen as the initial setting. The other settings chosen were 5ms, 25ms and 50ms. Settings smaller than 5ms would have caused a total number of frames considered too large for further processing. Changes of this and the next parameter affect the structure of the instances, resulting in basically different data sets. As a consequence, slight changes in the results are to be expected in any case. The uniform error rate generally seems to increase with greater step widths, hence a finer time resolution indeed adds information to the classification process. The best setting with respect to the error is thus reached at 5ms. Figure 4.6 shows the confusion matrix and DET curve at this setting. With 10ms, the accuracy has its peak at a different setting. Yet, as it does not change more than 0.12 percentage points between 5, 10 and 25ms, this can be considered a random artifact.

The *window size* corresponds to the number of samples that contribute to the MFCC computation for a single frame or feature vector, including the samples at the edge of the window being smoothed. If the window size is larger than the step width, which is typically the case, then windows overlap, causing some redundancy and a smoother transition across the signal. If the window size is smaller than the step width, then some information is disregarded. This is usually avoided. It was seen fit to specify the window size in relation to the step width. We evaluated a window size that is equal to the step width, a window size of three times the step width, which was also chosen as the default setting, and a large window setting of six times the step width. The smallest window size caused a visible degradation in both performance metrics, which may be due to the information lost as a result of the smoothing. Both larger, overlapping window size settings perform almost equally well, with the largest window size doing slightly better (see Figure 4.7). This indicates that some overlap should be present.

The number and choice of MFCCs to be used in the feature vector was varied as well. In general, higher coefficients are known to convey more speaker-specific characteristics, whereas

Position in System	Experiment # / Parameter	Value	Accuracy	Uniform Error
Front-end	(1) Step Width	5ms	40,35%	26,43%
		10ms	40,47%	27,63%
		25ms	40,37%	28,38%
		50ms	37,35%	28,95%
	(2) Window Size	1x step width	36,69%	30,01%
		3x step width	40,35%	26,43%
		6x step width	40,20%	26,09%
	(3) MFCC Coefficients	12 (1-12)	40,20%	26,09%
		19 (1-19)	39,01%	28,06%
		8 (12-19)	37,09%	26,56%
	(4) Delta Coefficients	Without deltas	40,20%	26,09%
		With deltas	39,73%	27,66%
Frame Filter	(5) Intensity-based Removal	0%	40,20%	26,09%
		20%	39,86%	27,11%
		40%	37,69%	27,05%
		60%	38,41%	26,74%
GMM Training	(6) Number of Mixtures	64	39,59%	26,57%
		128	40,20%	26,09%
		256	39,68%	27,22%
		512	40,23%	26,61%
	(7) MAP Relevance Factor	0.1	42,54%	24,95%
		2	42,14%	25,19%
		16	40,20%	26,09%
		100	37,41%	28,03%
	(8) Initialization	Random	42,54%	24,95%
		K-Means	41,92%	26,54%
NVC	(9) Feature Removal	0%	42,54%	24,95%
		10%	42,29%	25,05%
		30%	41,94%	25,24%
		65%	41,65%	25,57%
SVM Training	(10) Kernel Type	linear	42,54%	24,95%
		polynomial (2nd)	25,98%	42,36%
		polynomial (3rd)	24,51%	36,76%
		radial basis	13,41%	50,00%

Table 4.4: Summary of results from the FRISC main experiment series. **Bold face**: initial setting (as in previous experiments); **Shaded green**: setting with best result (kept for successive experiments). Explanation and interpretation is provided in the text.

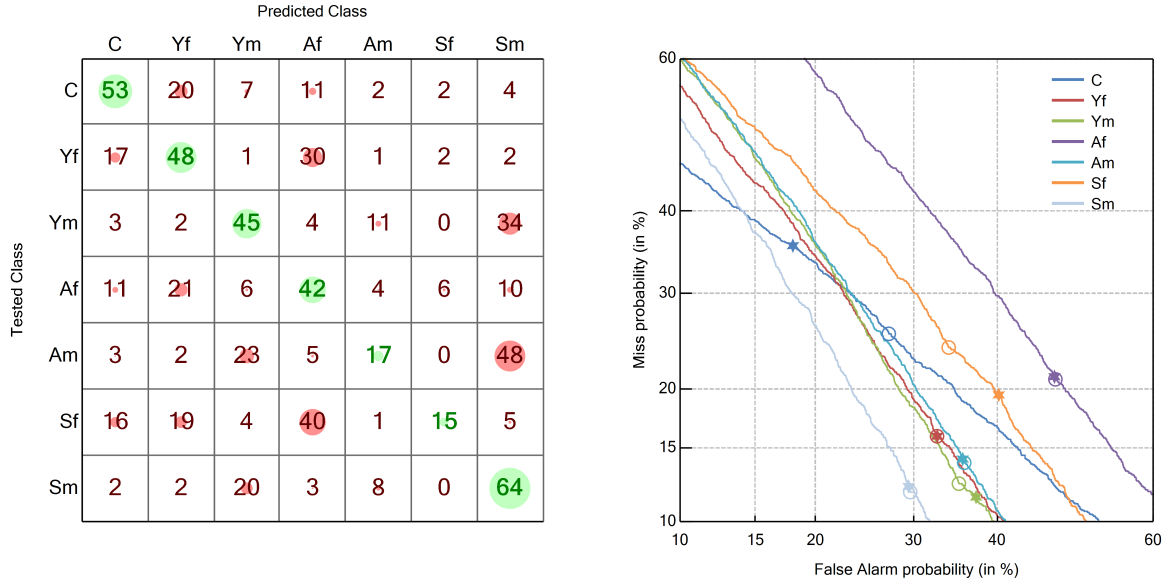


Figure 4.6: MFCC step width = 5ms. Measured performance: $\text{Acc} = 40.35\%$, $P_{FA} = 34.31\%$, $P_{Miss} = 18.56\%$, $P_{Err} = 26.43\%$

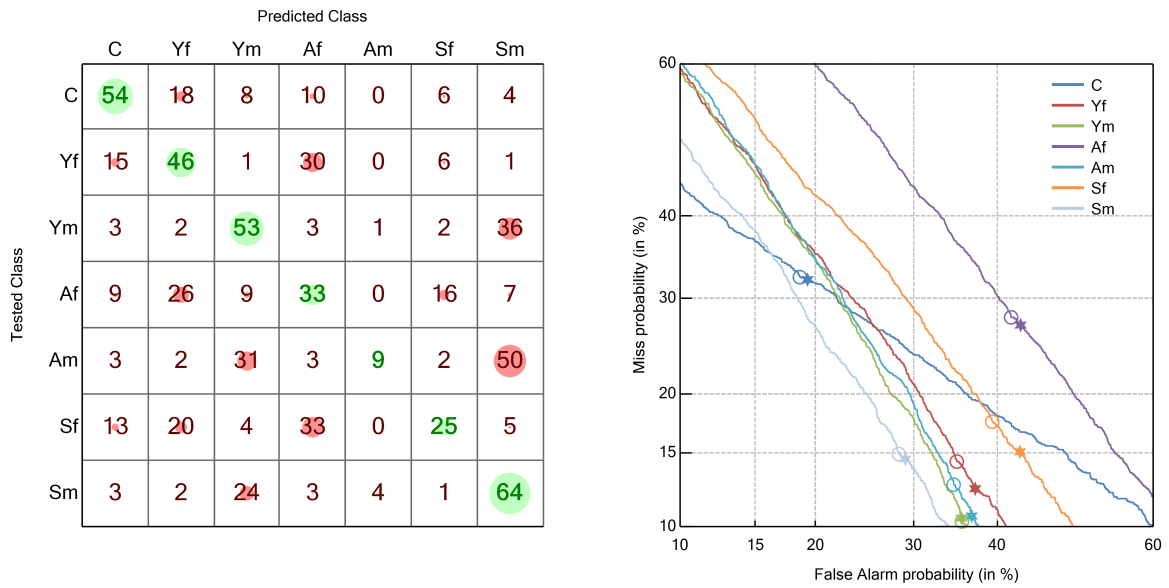


Figure 4.7: MFCC window size = 6x step width. Measured performance: $\text{Acc} = 40.2\%$, $P_{FA} = 34.73\%$, $P_{Miss} = 17.45\%$, $P_{Err} = 26.09\%$

lower ones convey more phoneme-related information. To limit the number of experiments, a selection of three settings was made: a *lower* MFCC setting covering coefficients 1–12 (a selection common in literature), a *higher* MFCC setting covering coefficients 12–19, and a *full* setting covering the union of the former, e.g. coefficients 1–19. Considering even higher coefficients was not deemed reasonable (see Section 2.4.4). The experimental results showed that the information from the lower MFCCs outperformed the other settings in the detection and even more in the identification task. Also, combining both regions degraded the result. One reason for this could be the existence of noise or nuisance attributes in the higher MFCCs which caused confusion. It should also be taken into account that no cepstral mean normalization was performed due to the off-line restriction of the method, which might have positively affected the result.

To put emphasis on the development of the MFCC over time, first order derivatives (deltas) of the coefficients as well as higher orders can be included in the feature set, which is quite commonly done in related work. We performed an experiment using only the deltas as additional features in the MFCC vector, resulting in a vector of 24 features. However, neither the accuracy nor the uniform error rate could be improved on our data by the inclusion of deltas.

Intensity-based Frame Filtering

For the next experiment, frame filtering was applied as described in Section 4.1.5, with varying settings for the parameter $T''_{silence}$ specifying the target percentage of frames to remove. Apart from the initial setting of using all frames, settings of 20%, 40%, and 60% were tried, with 60% being well above the share of non-speech in the data. Our hypothesis that the removal of actual silence should improve the classification result could not be confirmed by this experiment, since the setting without frame filtering enabled produced the best result. This means that the models picked up useful information from frames with lower intensity. However, there is no clear tendency in the data for the other settings, since the removal of 60% of frames works better than the removal of only 20%. This might be indicative of a suboptimal method for choosing the silent frames, so future studies might focus on the investigation of different methods for frame filtering. Figure 4.8 shows that even though the error differs by only approximately one percentage point for 0 and 60%, the confusion matrix is quite different in the second case, e.g. more male teenagers are confused with male seniors when the silence is removed. It would hence also be possible that some speaking rate information was conveyed through the non-speech features.

GMM Training

Training of the four generative models used in the approach, i.e. the UBM and the utterance GMMs *train*, *test1*, and *test2*, represents another area of the system in which multiple parameters of interest were investigated. The first and probably most well-known parameter is the number of mixtures (or Gaussians) in the GMMs. Due to MAP adaptation, this number is

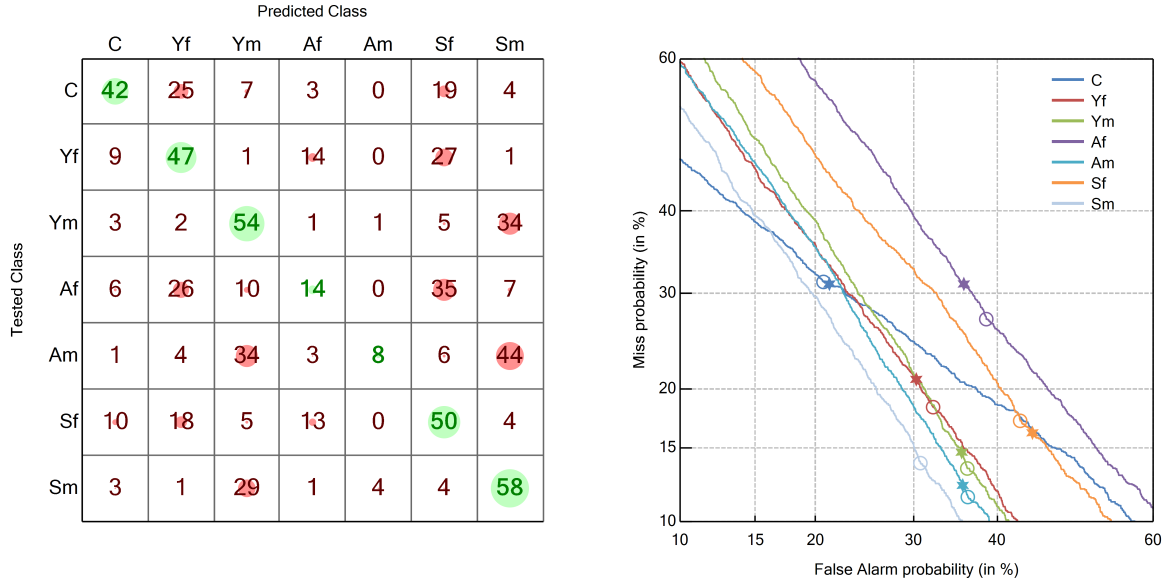


Figure 4.8: Intensity-based silence removal = 60%. Measured performance: $\text{Acc} = 38.41\%$, $P_{FA} = 34.06\%$, $P_{Miss} = 19.42\%$, $P_{Err} = 26.74\%$

identical in all models. The optimal number must be chosen with respect to the classification problem and the data available. Both too high and too low numbers can cause suboptimal results due to over- or underfitting, respectively. Our initial guess at 128 mixtures is set relatively low, which seems reasonable if we consider that the age SVM model is rather more general than the utterance-based GMMs. Comparing this with other powers of two, namely 64, 256, and 512 Gaussians, the numbers confirm our choice, although the differences between the settings are not overly large (and the accuracy with 512 mixtures is even slightly higher). The number of iterations of the EM algorithm was not investigated as part of this series since it is believed to eventually converge for a robust set of features (this was confirmed in preliminary experiments). This number was 5 as in Reynolds et al. (2000).

Following the number of mixtures, the next position in the system concerns the degree of MAP adaptation when deriving the utterance GMMs from the UBM. The parameter which controls this ratio of background vs. new training material in the MAP algorithm is the so-called *MAP relevance* parameter, or short MAP_{rel} (see Section 2.5.5). A higher value prefers the UBM, and a common default trade-off is a value of 16. Gajšek et al. (2010) consider the range of 8 to 16 to be a typical interval for good values; however, we chose a larger range to also determine the response in border cases: 0.1, 2, 16, and 100 were the settings tested in this series. In fact the numbers show that a lower setting, i.e. a preference of the new training material, yields a better performance. Considering that the training utterances were rather short, this makes sense. The value of 0.1 was selected for the further experiments (see Figure 4.9).

By default, the UBM was initialized using a random selection of feature vectors. It was

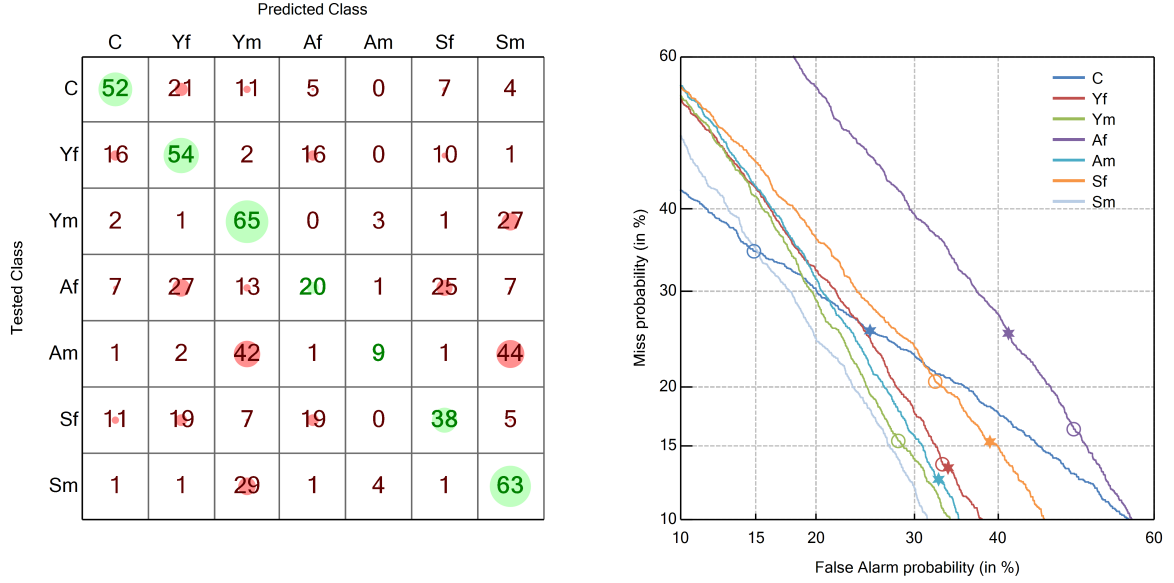


Figure 4.9: MAP relevance factor = 0.1. Measured performance: $\text{Acc} = 42.54\%$, $P_{FA} = 34.06\%$, $P_{Miss} = 15.83\%$, $P_{Err} = 24.95\%$

assumed that a more elaborate initialization function might be able to improve the result, so an initialization using the K-Means algorithm with 5 iterations was explored in another experiment. It turns out that K-Means is actually not helpful in this case and even degrades the result. It is also possible to use an uninitialized GMM (i.e. with all mixtures set to zero mean and variance). This however leads in most cases to considerably worse results.

Feature Space Reduction

The idea of the feature optimization step was to reduce the large feature space of the SVM, removing those features that are likely to have a negative impact on the model. The method used to select these features was described in Section 4.1.6. Our hypothesis is that there is an optimum between 0 and 100% feature removal, which is most likely noticeably apart from each of the two extremes. For this first evaluation, four settings were chosen, which describe the amount of features being removed: 0% (the default, i.e. all features are kept), 10% (only the most confusing features are removed), 30%, and 65%. It is really surprising and counter-intuitive that the reduction of dimensions does not show any positive effect. In fact the decrease in both uniform error rate and accuracy is monotonous, which supports the validity of the procedure. Nevertheless, it is possible that the choice of inter vs. intra class variability as the selection criterion failed to work as intended for the cepstral features, which would follow the argument of an “incompatible” feature reduction method also mentioned by [Lingenfelser et al. \(2010\)](#). A more elaborate factor analysis should be used in future studies to confirm these observations.

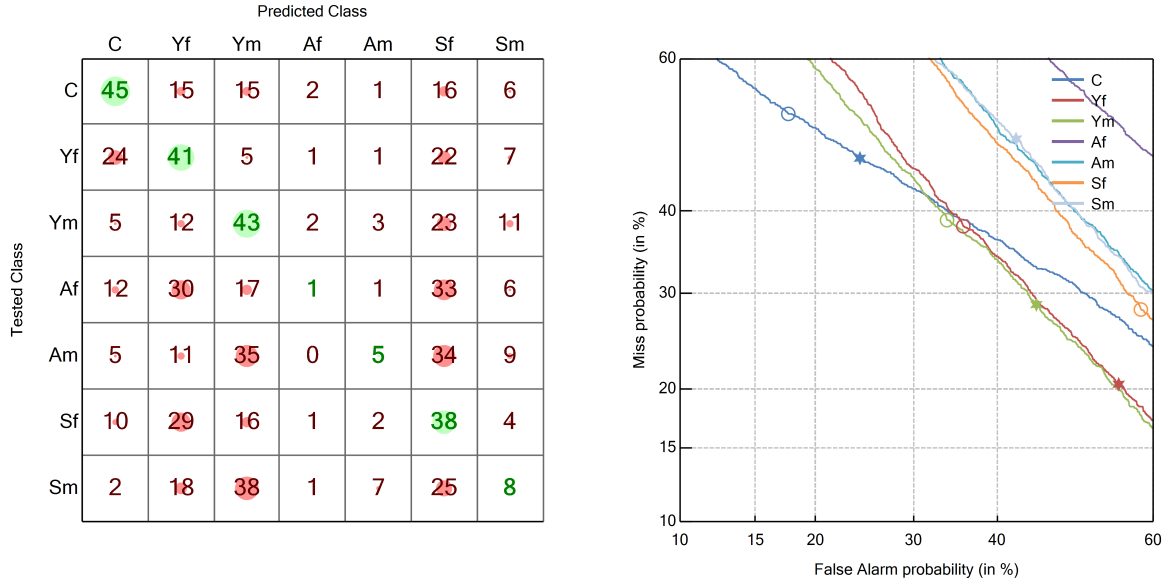


Figure 4.10: SVM kernel type = polynomial (2nd). Measured performance: $\text{Acc} = 25.98\%$, $P_{FA} = 37.79\%$, $P_{Miss} = 46.94\%$, $P_{Err} = 42.36\%$

SVM Training

Training of the support vector machine involves a number of parameters, including the SVM implementation. For this study, only a single parameter, with however a traditionally strong impact was varied: the kernel function. Since previous studies using the GMM-SVM approach, such as Bocklet et al. (2008), have determined the best results for the linear (or first degree polynomial) kernel, we were assuming to make the same observation in this study. The comparison was done with a second (square) and third degree (cubic) polynomial, as well as a radial base function (RBF) kernel. As it turns out, the linear kernel indeed outperforms the other kernels by a large margin. However, the kernel parameters of neither kernel were optimized due to the experimental complexity overhead involved, which can be especially problematic for the RBF kernel. Another observation is that detection and identification task results do not always follow the same ordering for this experiment. For comparison, the results for second degree polynomial SVM are shown in Figure 4.10. In the confusion matrix, it becomes obvious that the kernel function is not able to discriminate some classes (*Af*, *Am*, *Sm*) at all using the given features.

One-Shot Evaluation

As the results reported in Table 4.4 were obtained on the same *test2* data set, the series as a whole represents a parameter tuning setup and cannot be consulted for reliable absolute results in the best configuration. As discussed in Section 2.5.7, there was an additional *eval* data set whose labels were not known to the author. This “blind” data set was classified once

Parameter	Optimal Setting
Front-end Step Width	5ms
Front-end Window Size	6x step width
MFCC Coefficients	12 (1-12)
MFCC Deltas	without
Intensity-based Frame Removal	0%
Number of GMM Mixtures	128
MAP Relevance Factor	0.1
UBM Initialization	random
Feature Space Reduction	0%
Kernel Type	linear

Table 4.5: Summary of the FRISC parameter configuration leading to the best uniform error rate in the main experiment series.

using FRISC in the optimal parameter configuration summarized in Table 4.5. The predicted labels were sent to the independent evaluation site, which reported the final results given in Figure 4.11. Hence, the result is 38.0% accuracy, which is obviously lower than the 42.5% reached on the evaluation set. The difference of 4.5 can at least partly be attributed to the fact that utterances in the *eval* set were on average shorter. Unfortunately, detection error rates are not available from this evaluation.

Conclusion

The results described in the previous sections, reported on the FRISC Speaker Classification approach, provide new insights on the potential for positively influencing the classification performance through various parameters of the system. Several parameters have been pointed out which should be chosen according to the recommendations given as part of the summary in order to achieve best results, e.g. MFCC step width, MFCC window size, number of GMM mixtures, or MAP relevance factor. The observations made on the majority of the parameters seems reasonable and corresponds to the expectations. On the other hand, a couple of parameters did not behave intuitively and should be examined further. Since this experiment series was carried out following a rather strict design, adding parameters, methods, or settings it is not as straightforward. However, sufficient points of reference were named for upcoming work in the Speaker Classification field to build upon and investigate further. Obviously, the results are not completely independent from the data, which means that different conditions might require adjustment of the findings. Still, the relative importance and approximate direction for system design is expected to be similar even in these cases.

Taken by itself and as an absolute measure, with regard to the classification problem, the 38 respective 43% accuracy is a respectable result, although not reference. The latter was not a primary goal of this experiment. Even if a direct comparison with other systems is

		Confusion Matrix						
		Predicted Class						
		C	Yf	Ym	Af	Am	Sf	Sm
Tested Class	C	41	24	6	18	6	6	5
	Yf	18	42	5	24	1	8	3
	Ym	3	2	44	7	25	3	18
	Af	7	27	3	31	2	27	4
	Am	2	4	29	3	21	4	37
	Sf	5	12	6	26	3	36	11
	Sm	1	2	14	3	27	3	51

Figure 4.11: Confusion matrix showing the results of the evaluation of FRISC on the evaluation set.

difficult due to the different conditions, it is expected that further top-down engineering of the pattern recognition aspects would lead to better absolute results. An example might be a fusion with the baseline set of 450 acoustic features that was part of the Interspeech 2010 Paralinguistic Challenge (Schuller et al., 2010), and done in several of the contributions. The following section presents a proof of concept by using features from the original AGENDER approach in combination with the short-term features as presented herein.

4.4 Post-processing Level Fusion: A Hybrid System

Our main evaluation series has focused on short-term features, inter alia because they represent the less investigated aspect of Speaker Classification. These features are expected to expose a different, partially overlapping, but essentially complementary view to long-term features on the speech data. Therefore, it is expected that the fusion of both features in a single system works better than either feature family in separation. In conjunction with the main evaluation, a hybrid system was designed that combines both short and long term features in a single system. Since both types of features have already been investigated individually, the goal of this experiment was to determine how they would work in conjunction. A noteworthy improvement was anticipated with respect to reports on similar configurations in related work, and the generally known power of system fusion in the post-processing step of pattern recognition. Furthermore, we were interested to find out whether a combination of classification and regression approach could be used to improve the age estimation. As a

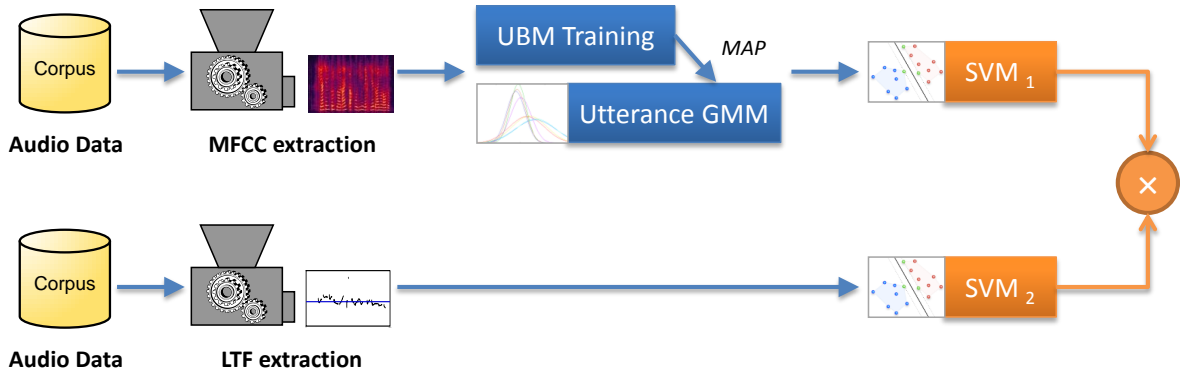


Figure 4.12: Experimental set-up which combines short and long term features into a single system. The upper part corresponds to the system as evaluated in Section 4.3.

side effect, the experiment would allow us to obtain a more realistic benchmark in terms of absolute classification accuracy. The study that is presented in this section is the outcome of a collaboration with the Human Language Technologies group of Meraka Institute at CSIR in Pretoria, with a major contribution of the South African partner. It was separately published in Heerden et al. (2010).

4.4.1 Combining Short and Long Term Features

Figure 4.12 displays the system combining short and long term features in an overview. The frame-based MFCCs were extracted according to the parameters in Table 4.5, and MAP-adapted utterance GMMs were created from a UBM as described earlier in Section 4.1.3. These supervectors were classified by a SVM into one of seven classes. As a further optimization in contrast to the previous experiments, the SVMs employed RBF kernels implemented by *LIBSVM*, which estimated the C and γ using a 10-fold grid search on a disjunctive subset of the training data.

The long-term features were derivatives of pitch, jitter, shimmer, and intensity (see Section 2.4.4), in particular mean, minimum, maximum, and deltas, which were all extracted on the full utterance using *Praat*. Furthermore, the first four formants F1 – F4 were extracted using Burg’s algorithm with a step width of 20 ms and a window size of 25 ms and added to the feature set in terms of mean, standard deviation, and delta. The limiting upper frequency was chosen at 5500 Hz, matching the value for adult females. The long-term feature vector was also classified using an RBF SVM.

The classes, corpus, and data sets used for this study were the same as in the main experiment series, i.e. the results reported herein were measured on the *test2* data set.

The changes to the SVM kernel alone were able to improve the classification accuracy on the MFCCs to 45.2%, which is an improvement of 2.7% absolute. This shows that the choice of kernel parameters has a considerable effect. (It has to be noted though that the training set was not identical due to the RBF training discussed above.) The long-term features achieved

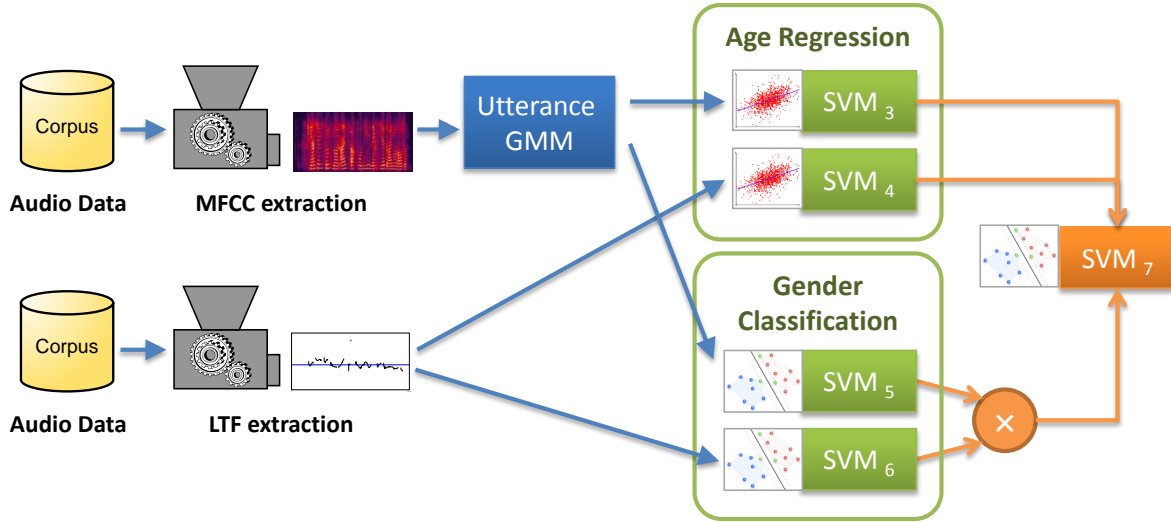


Figure 4.13: Configuration of the regression-based system. The age regressors (SVM₃ and SVM₄) are combined with a gender detector (SVM₅ and SVM₆) to form a final seven-class result (SVM₇).

45.7%, which is realistic compared with the 49% achieved on a set of 450 features by Schuller et al. (2010). By multiplicative combination of the SVM scores, the accuracy rose to 46.9% for the short/long term hybrid system.

4.4.2 Combining Classification and Regression

Apart from the introduction of feature fusion, the combination of classification and regression approaches was tested in the same study. Like the classifiers, the regressors were trained on both short and long term features as SVMs employing an RBF kernel. In this case however, a function for age prediction in years replaced the class decision. Since the regressors only estimate the age, a separate gender classifier for *Female*, *Male*, and *Children* was trained on the same data. These three pattern recognition components were coupled using a single seven-class *output* SVM (see Figure 4.13).

The result of this output SVM combining the age and gender regression results scored a total of 48.4% accuracy, which is 1.5% absolute more than the corresponding classifiers. Thereby, the regressors themselves were able to predict age with an average of 18.8 years root mean squared error and a corresponding correlation coefficient of 0.48. The gender detection reached an accuracy of 88.5%.

In a final experiment, classifiers (short and long term) and regressors (through the output SVM) were combined using multiplicative score combination as shown in Figure 4.14. This configuration achieved the best value measured so far on the *test2* set, which is 50.7%. The confusion matrix on this result is shown in Figure 4.15.

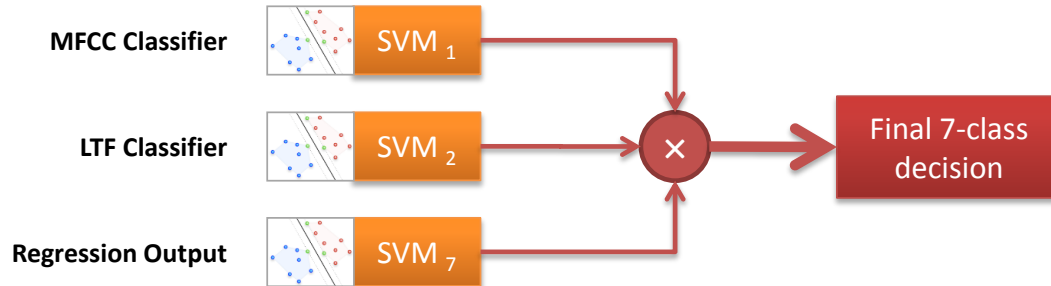


Figure 4.14: The final system, which fuses the sub-systems from Figure 4.12 and Figure 4.13 using score multiplication.

Confusion Matrix

	Predicted Class						
	C	Yf	Ym	Af	Am	Sf	Sm
C	59	16	6	8	1	8	2
Yf	13	56	0	20	0	11	0
Ym	0	0	47	0	20	4	29
Af	5	18	4	38	3	30	2
Am	1	0	26	1	31	3	38
Sf	6	13	1	24	0	56	1
Sm	0	0	9	0	16	3	73

Figure 4.15: Confusion matrix for the system depicted in Figure 4.14. The accuracy of this system corresponds to 50.7%.

4.4.3 Conclusion

The supplementary experiments presented in this section as an extension to FRISC have once more confirmed that the fusion of different methods on the post-processing layer (late fusion) have the potential to easily lift existing results to a higher level. It also reassures us to keep studying all feature families, since each one has its merits. However, while applications certainly depend on high accuracies, we still argue that the foundation of good classification results happens on the level of study and optimization of individual methods, rather than their fusion, which is its own sub-field of science.

4.5 Influences of Language and Culture

Globalization is an important topic in the IT world today, and part of the goal of pushing Speaker Classification is to ensure that it is a technology that everyone can benefit from. As stated in the research questions, we will address the issue of language and culture independence of FRISC.

Speech is not only sensitive to age and gender, but conveys many more characteristics. A number of these aspects are related to language and culture (see Section 2.1.4). Their influence on speech features overlaps the age and gender related cues, which gives us sufficient reason to investigate whether this has an impact on the general idea of classification. Until today, the effects of culture on the various facets of Speaker Classification have received comparatively little attention. When looking at speaker recognition, various authors have reported that the typical approaches are rather insensitive to the language being spoken (Bellegarda, 2007). On the other hand, speaker recognition can actually pick up culture as an additional discriminating feature and benefit from it. This is unlike age and gender recognition, where it has to be considered a nuisance attribute. But even some speaker verification systems have shown statistically significant, yet relatively small in magnitude, differences in the performance of multiple languages from the same corpus (Kleynhans & Barnard, 2005). Emotion recognition, on the other hand, has been shown to depend strongly on the language being spoken (Shami & Verhelst, 2007). These observations basically encourage us to believe that the general concept, i.e. using the proposed acoustic features with a classifier for seven-class discrimination, would be independent of language and culture. At the same time, they generate sufficient doubt to wonder whether they will also work equally well and without changes in all such contexts. Hence, in this section, we will conduct a set of experiments to investigate the actual behavior.

To recapitulate, there are three possible outcomes we might observe:

- There is no effect of culture on the features and hence on the classification accuracy. If the features were to remain unaffected by language and culture, it would be the optimal case, since we could use an existing FRISC system anywhere without modification. However, this case is unlikely, as certain effects on our features have already been reported in various studies.

- There is a measurable effect on the features, but the impact of age and gender is still intact. In this case, language and culture would represent some kind of noise that is mixed with the other information. It could overlay the information or shift the feature space, something that can be corrected using the right filter. The most practical solution in this case would be the same that is applied in other cases of noisy conditions: to train the models with the same anticipated “noise” applied, i.e. to train a different model for each language or cultural grouping. This case corresponds to our hypothesis.
- The effect on the features is so strong that either none of the original information is preserved, or it becomes too weak for classification. It could also be that it affects the features in a way that forces us to use different class boundaries. This would mean that FRISC could not be applied without further re-engineering, e.g. choice of different features. However, it is also rather unlikely, since our features were particularly chosen to be independent from linguistic effects, and the acoustic impact of language and culture is not believed to be overly strong.

Having a different model for each language case of course means that we also need to select the right model before we can run Speaker Classification. The most straightforward solution would be to detect the language spoken in parallel with the classification using multiple models, and then use the result from the model that matches the detected language. Another solution would be to reach a compromise between model generality and best possible accuracy. For example, instead of having a model for each South African language, a single South African or even African model could be more suitable.

We follow the same principled strategy as in previous chapters, which means that our focus will be on observable and explainable effects. Comparing scores of the final systems might seem like an intuitive way to approach our question, but the commitment to a particular system, combined with a grave change in corpus data, might mix certain influences and conceal the actual effects. Therefore, these experiments focus on the inspection of acoustic features, in particular global features that can be explained well, and run performance experiments in a closed set-up.

The experiments were performed in collaboration with Meraka Institute in Pretoria, South Africa, and were separately published by [Feld, Barnard, Heerden, and Müller \(2009\)](#). For the specific case of automatic age classification, which we consider in this section, we are not aware of any cross-cultural or multi-lingual studies, although there are a lot of useful ASR related applications associated with this task. The experiments follow a two-step approach: First, we have set out to investigate the influence of language on a prototypical feature set that has been employed for age classification. In a next step, the effect of applying language-specific models trained on one language to either the same or a different language, is evaluated. Through this we hope to be able to find out in how far our current Speaker Classification approach – the combination of features, classification architecture and pre-trained models – is dependent on language. As a data basis for these experiments, we have chosen the *Lwazi* corpus, which is presented in the following section.

4.5.1 The Lwazi ASR Corpus

For an optimal investigation of the hypothesis, a corpus is needed which fulfills certain requirements:

- Contain several languages from different language families with a substantial diversity according to the aspects introduced in Section 2.1.4
- For comparison, contain several languages from the same family, with differences mainly concentrating on the linguistic level
- An ample amount of material for each language
- Annotation of speaker ages
- A suitable distribution of ages
- Absence of corpus effects, which can occur when different corpora are merged into one.

While we obviously cannot test for differences in all possible feature dimensions, and corpora with multiple ethnical groupings a fairly rare, requesting an assortment of fundamentally different language families should give a good exemplary condition of high divergence, i.e. a “worst case” scenario. Still, these requirements are quite strict, and the general availability of good corpora is low, particularly with respect to cross-cultural languages.

We considered two corpora for this study: *GlobalPhone* and *Lwazi*. The former (Schultz & Waibel, 2001) contains an assortment of fourteen languages, including some from different families, such as German, Chinese, and Russian. The corpus is well organized and the samples are in high quality due to in-lab recording. In spite of the high sample quality, we noticed channel differences between certain corpora, such as English and Turkish. This can probably be attributed to the fact that all recordings were done on site, using the same equipment and location for a single corpus, but different resources compared to the other corpora. For ASR and other tasks, this would not pose a big problem, but in case of these experiments, there was a high risk of these attributes interfering with our target attributes. The *Lwazi* corpus⁵ (Barnard, Davel, & Heerden, 2009), on the other hand, contains only languages from South Africa. Nonetheless, the official languages of the nation unite multiple language families with very different structure, as well as different cultural backgrounds, and are perfectly suitable in this regard. The samples are comprised of inbound phone calls to a call center, which means that they are recorded in a central place. The recording quality is below that of *GlobalPhone*, both in terms of channel and background noise; yet, these effects pose less of a problem as they are independent from the language. Due to this consideration, the *Lwazi* corpus seems like a perfect candidate to confirm or refute language and culture independence of the approach. Both corpora unfortunately do not feature speech from children.

⁵Information on the Lwazi corpus is available on-line from the website of Meraka Institute, CSIR, South Africa: <http://www.meraka.org.za/lwazi>

Language	Code	Family	Sub-Family	Total minutes	Speech minutes
isiZulu	zul	Bantu	Zunda	525	407
isiXhosa	xho	Bantu	Zunda	470	370
Afrikaans	afr	Germanic	Low Franconian	213	182
Sepedi	nso	Bantu	Sotho-Tswana	394	301
Setswana	tsn	Bantu	Sotho-Tswana	379	295
Sesotho	sot	Bantu	Sotho-Tswana	387	313
South African English	eng	Germanic	Anglic	304	255
Xitsonga	tso	Bantu	Tswa-Ronga	378	316
siSwati	ssw	Bantu	Tekela	603	479
Tshivenda	ven	Bantu		354	286
isiNdebele	nbl	Bantu	Sotho-Tswana	564	465

Table 4.6: The official languages of South Africa, their ISO 639-3:2007 language codes, language families, and the amount of speech contained in the *Lwazi* corpus.

The *Lwazi* corpus was developed for ASR as part of a project that aims to demonstrate the use of speech technology in information service delivery in South Africa. In particular, the three-year *Lwazi* project (2006-2009) produced the core tools and technologies required for the development of multilingual voice-response systems in all eleven of South Africa’s official languages, and piloted the use of these technologies in government information service delivery.

The *Lwazi* ASR corpus consists of annotated speech data in the languages listed in Table 4.6, which also summarizes the amount of speech available in each language. This data was collected in South Africa over the telephone, by soliciting callers in each of the languages from a variety of backgrounds. Approximately 100 male and 100 female first-language speakers contributed speech in each of the languages, and an approximate balance between mobile and fixed-line telephones was maintained across languages. This corpus was restricted to adult speech; details of the distribution of speaker ages are provided in Figure 4.16. Unfortunately, as can be seen, there are almost no children in the corpus and only a relatively low number of elderly people. Consequently, the results given in Section 4.5.4 do not consider the *Children* class. For some languages, extensive dialectal variation exists within South Africa. However, this variation is not well documented; for the purposes of the corpus, the intent was to concentrate on the dominant dialects of each languages, but dialects were not rigorously controlled.

The languages in Table 4.6 fall into two broad families, with Afrikaans and English being Germanic languages and the remaining nine languages belonging to the Bantu family of languages (in particular, the Southern Bantu sub-family). The co-location of these widely different groups of families in the same country is a historical “accident”; although their many years of co-existence have lead to some mutual influences, the two groups remain separated by a wide linguistic gulf. For example, the Bantu languages of South Africa are tonal languages

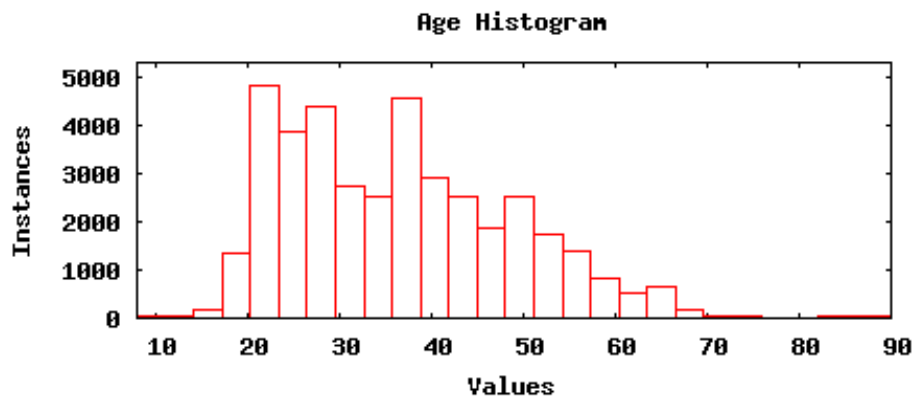


Figure 4.16: Distribution of ages for a subset of seven languages in the *Lwazi* corpus.

characterized by an extensive system of noun classes; they are strongly agglutinative, with affixes playing a variety of syntactic and semantic roles; their syllables tend to have regular CV or V structures. In all these respects the Southern Bantu languages differ from English and Afrikaans, which are fairly typical Germanic languages. Hence, these languages are a good testing ground to search for differences in the way that speaker age is expressed in speech.

With over 14 GB of speech data, the *Lwazi* corpus is quite substantial and working with it requires significant computing time and processing power. Also, it was created under developing-world conditions, where limitations in infrastructure and the availability of skilled personnel are expected to impact on corpus quality. An initial random listening and signal analysis was therefore undertaken; it revealed some interesting facts about the material. Compared to widely-used speech corpora such as *GlobalPhone* or *TIMIT* (Garofolo et al., 1998), there is considerable background noise, which is a consequence of the fact that many speakers were speaking on mobile telephones and from everyday locations. For the same reasons, the amplitudes of the speakers are much more variable than in standard corpora. On the signal level, the data contained varying DC offset and even some clipping on some speakers. Again, that is explained by the absence of constant recording conditions with respect to the sender’s microphone and the channel. All of this is part of the compromise when trying to find a large number of native speakers for these languages, and it has to be taken into account when comparing the results with evaluations done on other corpora. Note that the effects were mostly randomly distributed over all languages – there were no visible artifacts restricted to a single language. Thus, for a comparative study on classification performance, these observations are not considered critical.

4.5.2 Feature Analysis

To get an initial idea of how the influence of language and culture manifests itself in a speaker’s voice, a semi-automated corpus analysis was performed. As explained earlier, rather than

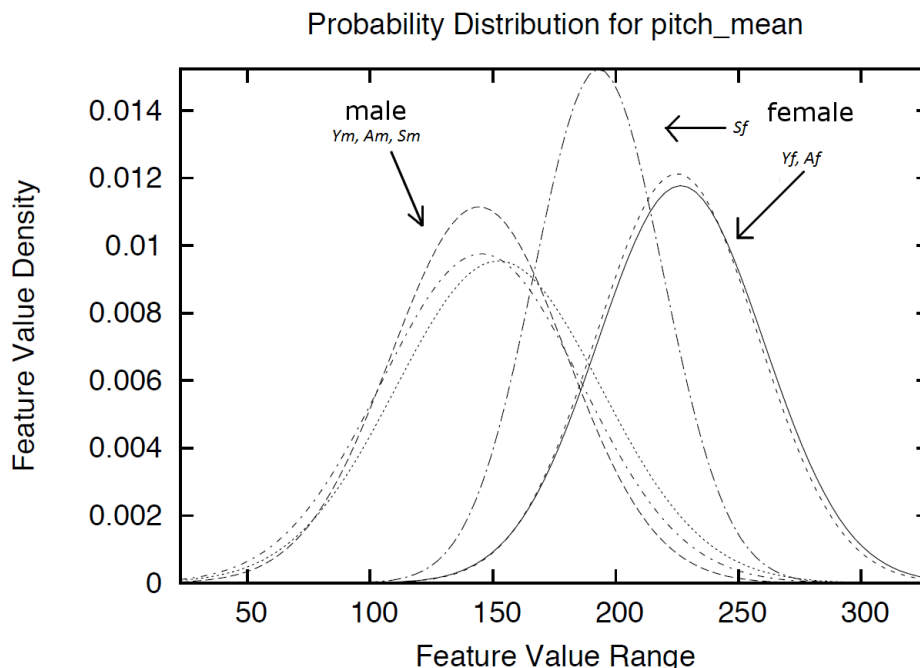


Figure 4.17: Feature value distribution of mean pitch over the age/gender classes for South African English speakers. The curve peaks represent the class mean while the width indicates its inner-class variance. This Figure shows the typical separation of female and male voices. There were no voices of children in the data.

simply taking a random collection of features and processing them with various out-of-the-box classification algorithms, it is more purposeful to take a look at the expressiveness of some of the available features individually. This analysis is comparable with the corpus analyses performed by Müller (2005). This not only saves time, but also gives a better understanding of the decision criteria and simplifies the task of fine-tuning the classifier later on. A representation that is well suited for this purpose is the approximation of a normal distribution of feature values (see Section 2.5.1). Sketched over all utterances in the corpus or a particular language, it provides a graphical comparison of the differences between the target classes, i.e. ages and genders in our case. This task can be automated to some extent, but many of the more interesting relations are hard to recognize by a machine and can usually only be spotted by manually looking at the results. The class boundaries have been chosen identical to the previous FRISC experiment series (see Table 4.1 on page 104), minus the *Children* class. We had to restrict our experiments to a subset of languages as the age labels were not yet fully available for the remaining languages.

The features that were selected for the corpus analysis are acoustic long-term features computed on full utterances that have already proven useful before. They are derivatives of the pitch, jitter and shimmer families of features (see Section 2.4.4) computed as averages on

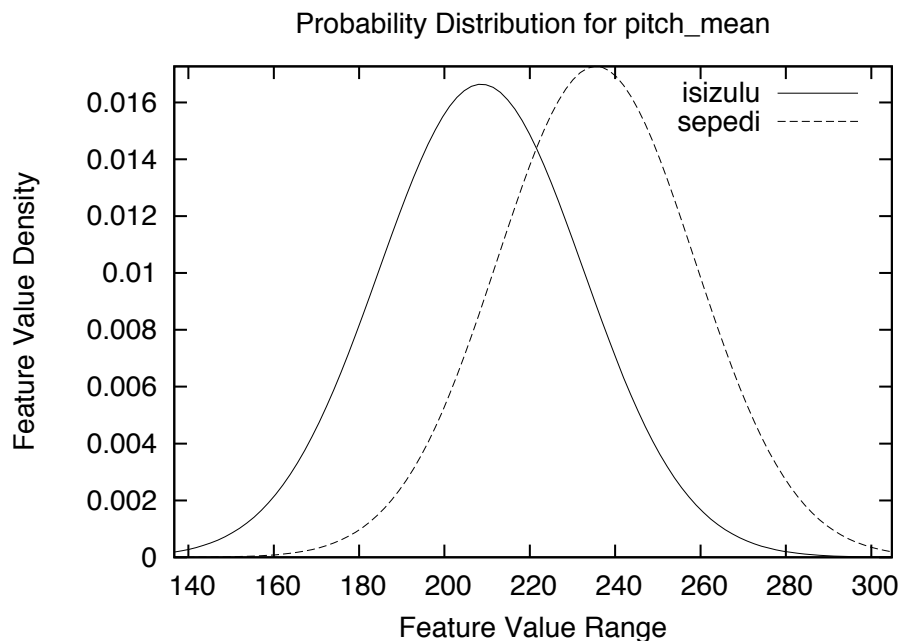


Figure 4.18: Comparison of the mean pitch distributions for isiZulu and Sepedi of speakers in the *Young female* class.

whole utterances and were obtained using *Praat*. This decision was made in spite of FRISC being largely based on MFCC features, which was shown to generally produce lower error rates, for two reasons: On the one hand, the long-term features are more transparent and expressive to humans than for instance raw MFCC coefficients, and thus the preferred way to get an initial measure of applicability. They provide a better understanding of the phonetic causes for speakers of some languages being classified better than those of others. On the other hand, it is likely that there is quite some overlap concerning the information in the two feature sets, and it is fair to assume that the observations carry over to some extent.

From the data we were looking at, a large number of distributions can be examined, comparing either languages grouped by age class or age groups across languages. In many of these charts, there is indeed a notable difference in the distribution of values for the individual languages. A selection of these graphs is provided in this section in order to illustrate some of the most interesting cases where major deviations are visible. Two of the features with a very characteristic language-specific average are the mean and standard deviation of pitch. Figure 4.17 shows the typical distribution for speakers of South African English, with male speakers having generally lower pitch than female speakers. The *adult female* voices are a bit higher than expected, which is probably due to the slightly uneven distribution of ages (bias towards < 30). It confirms that pitch is a good feature for gender recognition, and to some extent helpful to distinguish ages. In order to see how language affects this circumstance, we next study that feature for the individual languages and a specific age class. In Figure 4.18,

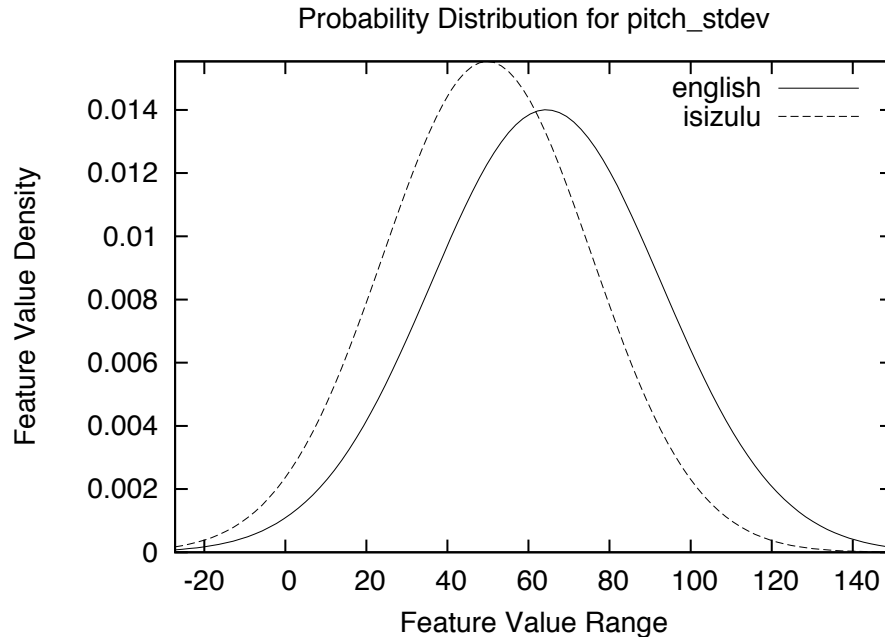


Figure 4.19: Value distributions of frequency tremor for South African English and isiZulu of *Adult female* speakers. The separation is not as clear as in Figure 4.18, but it still shows a considerable difference.

this was done for *Young female* speakers of isiZulu and Sepedi. Our analyses revealed that voices of isiZulu speakers are on average 25 Hz lower than those of Sepedi speakers, which would make them easily confusable with seniors if the system was trained only on data from Figure 4.17. Figure 4.19 shows a similar behavior for the standard deviation of pitch with different speakers.

The observation that some languages are more different than others in terms of long-time features surfaces in several of the charts, and the actual ordering of languages changes depending on the feature that is considered. The Germanic languages in the *Lwazi* corpus are usually rather close. These aspects appear to be stable over a set of languages, which is in favor of the hypothesis that different Speaker Classification models would be needed to achieve optimal performance for each language. For example, a particularly low jitter (micro-variations in the pitch level) can be observed for adults speaking Sepedi (see Figure 4.20), while a high shimmer (micro-variations of the amplitude) appears to be characteristic for adult female Zulu speakers (see Figure 4.21).

4.5.3 Regression Analysis

In order to confirm the findings obtained on the long-term features described in the previous section using actual numbers, a simple pattern recognition set-up was created. As some age classes were not backed by a large number of speakers, a regression task was carried out for

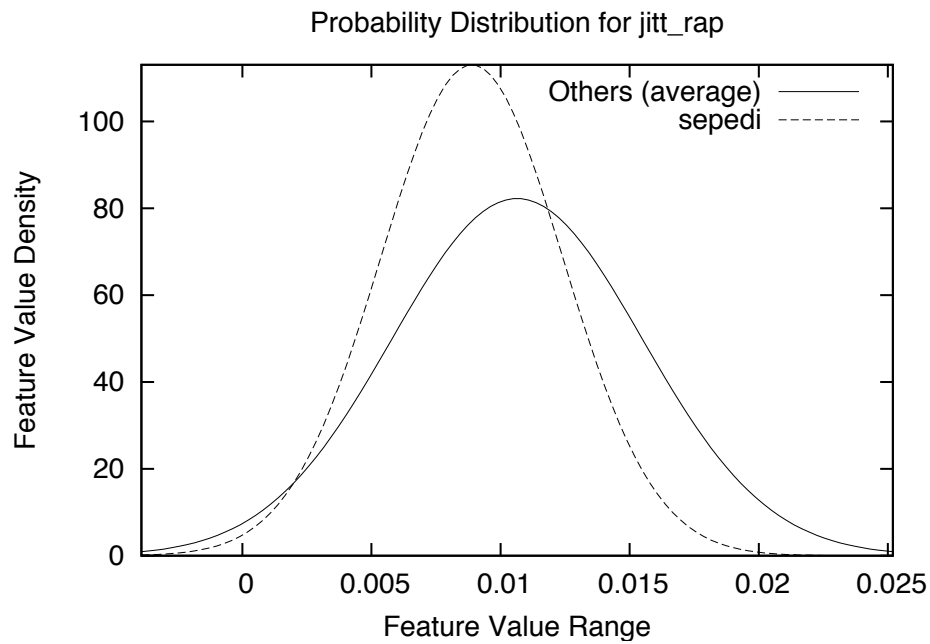


Figure 4.20: Jitter (here: relative average perturbation, RAP) as an example of a criterion where one language (Sepedi) has an average that is rather distant from that of all other languages. The statistics contains only speakers of the *Adult male* class.

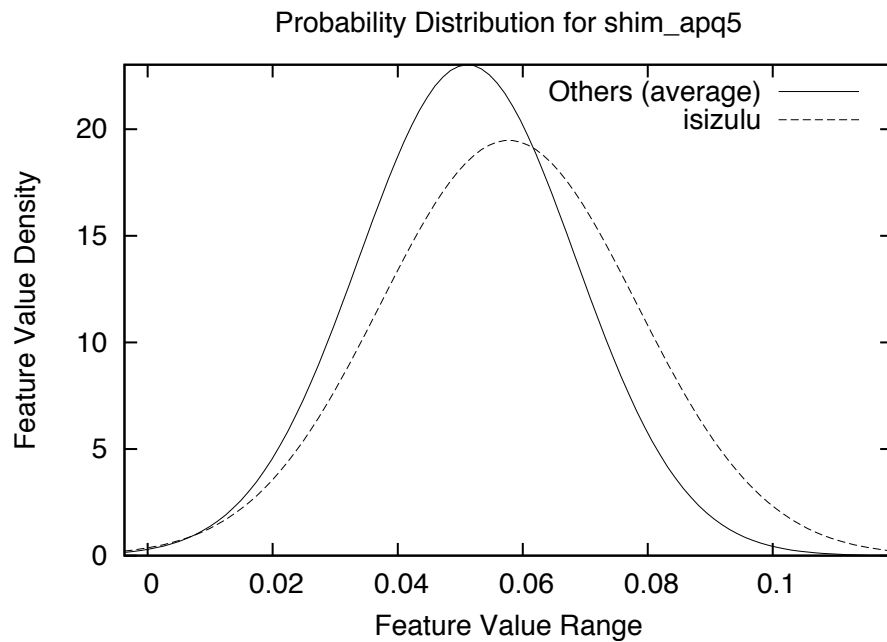


Figure 4.21: For *Adult female* Zulu speakers, the value range of shimmer (here: the amplitude perturbation quotient for 5-point periods, APQ5) is characteristically higher than for the rest of the languages.

Language	training speakers / utterances	test speakers / utterances	Mean age	StdDev age
Afrikaans	159 / 4767	40 / 1193	34.7	14.3
English	155 / 4623	37 / 1105	37.7	15.8
isiZulu	149 / 4349	38 / 1122	35.4	14.1
isiXhosa	131 / 3865	34 / 1007	36.8	10.3
Sesotho	153 / 4576	36 / 1049	34.3	12.7
Setswana	152 / 4481	39 / 1149	36.0	13.6

Table 4.7: Data used for regression training and testing.

this analysis. Least-squares linear regressors were developed using training data from each of the six languages listed in Table 4.7. These languages were selected since they span a variety of the (sub-) families found in the *Lwazi* corpus where a comparison seems of particular interest, and reliable meta-data was available for their speakers. Two measures of predictive accuracy (mean absolute error and correlation coefficient) were then computed, employing the models trained on each language separately on the test data from all languages. In particular, the following steps were carried out:

First, each of the six sub-corpora was divided into training and test sets. The ratio of training to test data was approximately 80:20, with no speaker overlap between these sets. The feature vectors listed in Table 4.8 were calculated for each utterance in the training and test sets of all languages. Next, each training set was scaled separately so that each feature has zero mean and unit variance; for each language α the regression vector w_α was then calculated as

$$w_\alpha = (X_\alpha^t X_\alpha)^{-1} X_\alpha^t t_\alpha,$$

where X_α is the matrix formed by stacking all the scaled feature vectors (each extended with a “bias” term of 1) together and t_α is a vector consisting of all the true ages corresponding to the feature vectors in X_α . The ages of the speakers of all utterances in the test sets were estimated as

$$y_{\alpha\beta i} = x_{\beta i}^t w_\alpha,$$

where $x_{\beta i}$ is the extended feature vector for speaker i from language β . For each pair (α, β) , the average of the absolute difference between the estimated and actual ages for all utterances in the test set was calculated, as well as the Pearson correlation coefficients between the estimated and actual ages.

4.5.4 Results

Figure 4.22 shows the mean prediction errors and correlation coefficients resulting from our linear regression. We see that the highest accuracy by both measures is generally achieved when training and test sets are drawn from the same language, suggesting that the age factors are expressed differently in the different languages. When training and test languages agree,

Number	Feature	Number	Feature	Number	Feature	Number	Feature
1	pitch_min	8	intens_mean	12	jit_l	16	shim_l
2	pitch_max	9	intens_min	13	jit_la	17	shim_ldb
3	pitch_quant	10	intens_max	14	jit_ppq	18	shim_apq3
4	pitch_mean	11	intens_stddev	15	jit_rap	19	shim_apq5
5	pitch_stddev					20	shim_apq11
6	pitch_mas						
7	pitch_swoj						

Table 4.8: Feature vector used in the regression experiment. Definitions of these features are available in Section 2.4.4.

the correlation coefficients range between approximately 0.2 and 0.36, suggesting that this is a challenging task; the corresponding mean values of the absolute errors range between 7.7 and 12.8 years. (As a basis for comparison, when we apply these same methods to a previously-used corpus of German utterances, we obtain correlation coefficients and mean absolute errors of 0.38 and 17.2, respectively. The range of ages in that corpus is substantially larger than in *Lwazi*, which explains both the higher correlation coefficient achieved – since it is easier to predict more extreme ages – and the larger mean absolute error.)

When comparing the cross-language predictors, it is interesting to note that the language families are apparently not particularly relevant to age prediction. Thus, the predictor for English ages with the largest correlation coefficient is derived from Sesotho data, and the isiZulu and Setswana predictors are quite accurate when applied to data from the other language in this pair. In contrast, the Sesotho and Setswana regressors do not perform well when applied to test data from the other language in this closely-related pair of languages.

Figure 4.23 shows the regression weights w_α calculated for all languages. Since all features were normalized to have the same mean and variance, these weights are directly comparable. Many features show significant variation across the different languages. The most consistently important value corresponds to feature 13, which is a long-term average of the jitter in pitch frequency; however, even that feature contributes little to the isiZulu regressor. Feature 10 (related to the minimum intensity within an utterance) has a large negative contribution for English, but contributes somewhat positively to the age regressors for isiZulu and isiXhosa.

4.5.5 Conclusion

The cross-language comparison of the age regressors shows that the best predictions (in terms of both measures employed in this study) are obtained when training and test data are drawn from the same language. This confirms that the age predictors, in terms of the features employed here, are somewhat language dependent – a conclusion that is further strengthened by the fact that the regression vectors have significantly different shapes for the different languages. However, the predictors are not particularly accurate when applied to test data within the same language family. This observation may indicate that there are other relevant

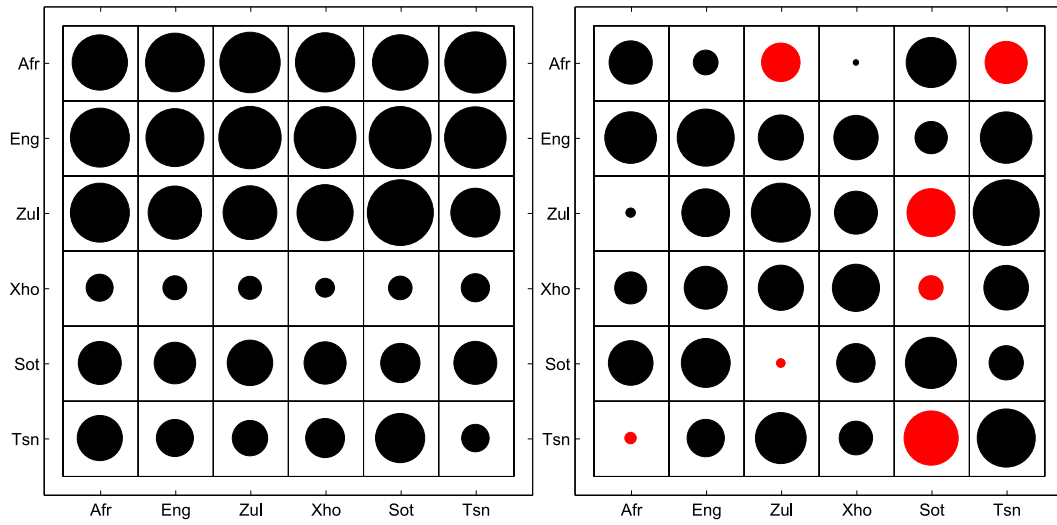


Figure 4.22: Regression results on the *Lwazi* corpus when the linear regressor from one source language is applied to all six target languages. The rows correspond to values for the same target language, and the columns correspond to the language used for training. *Left*: Mean absolute prediction errors (larger bubbles represent a bigger error). *Right*: Correlation coefficients (negative values are indicated by red areas).

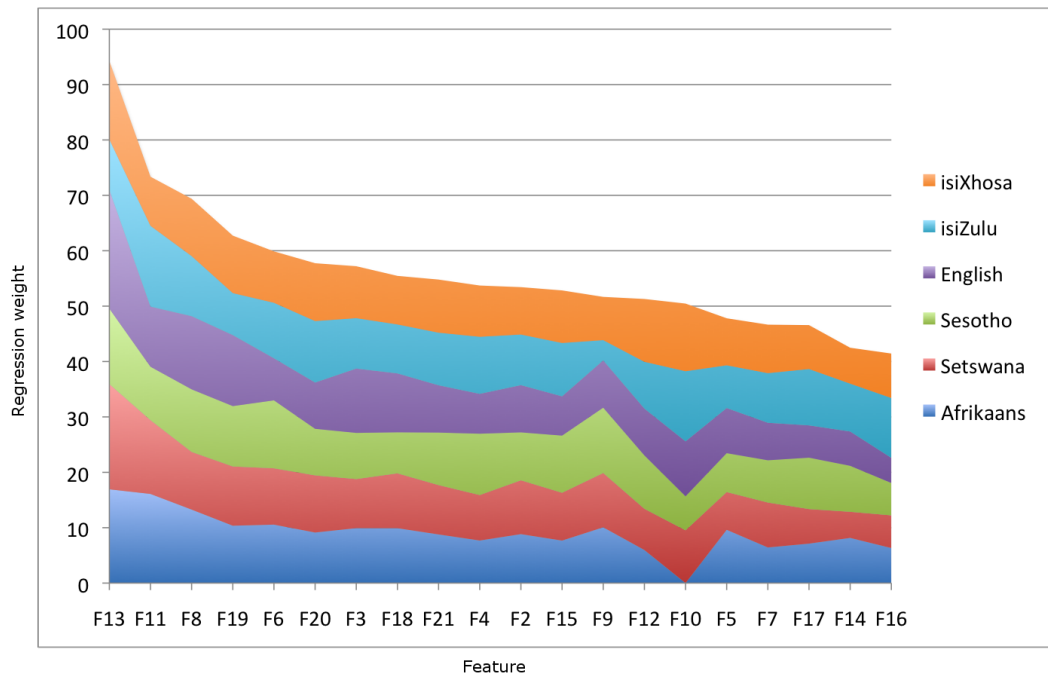


Figure 4.23: Regression weights calculated on training data from six different languages.

variables – possibly cultural or socio-economic – which play an important role in the observed inter-language differences.

The results from the distribution analysis show that the various languages do not behave in a consistent fashion with respect to age changes. Thus, even this basic paralinguistic information source seems to have significant language-specific (or culture-specific) aspects. The differences between features in this respect suggest that in the end, a single long-term feature set may not provide the optimal performance for all languages.

Overall, these results confirm three important aspects of the hypothesis on inter-cultural speaker age and gender recognition: First, there are differences visible on the feature level, which confirms in the data what literature suggests. Second, the application of simple regressors trained on one language performs better on that same language than cross-language. Third, the result of cross language family application of classifiers is still sufficiently good to confirm that the approach itself is not culture-dependent, i.e. can generally achieve a similar performance in other languages.

Not all possible questions could be pursued by this initial study into the subject of inter-cultural Speaker Classification. For instance, a study comparing actual tuned GMM-SVM classifiers for each of the languages might show whether the feature set and classification method play a role. Such an evaluation however would be quite substantial due to the volume of data involved. Also, it is still an open question in how far language-specific models can be generalized, i.e. how a model trained using background data from various languages would perform compared to inter-language and cross-language models of a single language. This shows that there is still potential for further investigation and follow-up studies in this topic.

4.6 Comparison with Synthesized Speech: Analysis-by-Synthesis

This section introduces a further type of evaluation arranged in the context of FRISC. The studies that were presented in the previous sections, including related work from Chapter 3, all have in common that they were based on natural voices. There is however also a complementary approach, which draws its motivation from the fact that the classification process makes a number of assumptions about how age and gender are expressed in speech. This can be considered a kind of reverse model of human speech production or parts of it. Therefore, instead of testing with noisy real-world data, it might also be appropriate to compare this model to another model that deals with speech production. Some of the most advanced and practical models can be found in the field of speech synthesis. Causing the voice to sound like a particular age or gender makes it more natural and allows it to carry additional semantics. A similar example is to enrich the voice with emotion, as has been done in the *MARY* (Schröder, Charfuelan, Pammi, & Steiner, 2011) speech synthesis platform. By generating speech on one side by a synthesis model that makes certain assumptions about the speaker (e.g. her age), and then classifying that speech by an independent system also incorporating a model of the speaker properties, but created from a different perspective, the dimension

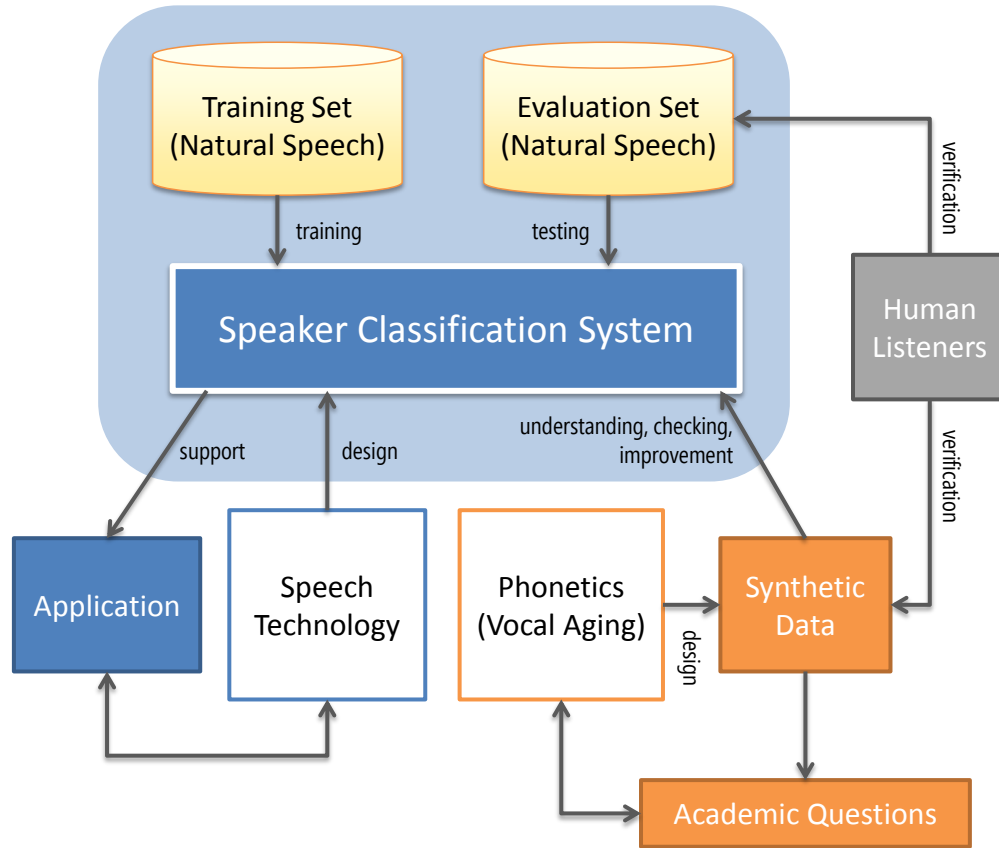


Figure 4.24: Scheme of the general idea behind the study on synthesized voices: Phonetics can be used to understand and help improve the automatic age classification system by the use of controllable synthetic data.

of the problem is reduced to a more manageable range, and influences of features are visible much more clearly. This helps to uncover flaws in either model and is called the analysis-by-synthesis paradigm. It has been previously applied inter alia by Schötz (2006). One piece is still missing in this description: To prevent both models from losing contact to the actual physiological foundation, e.g. by too much abstraction or by introducing artificial features that support either model, but are not really existent in nature, the human has to be kept in the loop. More precisely, the synthesized speech has to be rated according to the same criteria that the machine applies by human listeners, and their result has to be compared to what the system produces. Combining these three components is supposed to give a valuable and different kind of insight into both speech technologies and traditional speech-based evaluations. These concepts are depicted in a more global view in Figure 4.24.

A set-up as the one just described is necessarily embedded in an interdisciplinary context centered around a strong phonetic component that can explain the models. Such a study on speaker age was undertaken as a collaborative effort between the Department of Com-

putational Linguistics and Phonetics at Saarland University, which represented the speech synthesis and phonetic expertise (Lasarczyk, 2010), and the German Research Center for Artificial Intelligence, which represented the view of automatic Speaker Classification through the FRISC approach. The study⁶ consisted of four steps. The first step involved creating a phonetic age model of human voice aging. The second step was the synthesis of several test utterances for the experiments. As a third step, the automatic recognition component was evaluated, followed by the last step, which was an experiment on the same data but with human listeners. Finally, both results were compared.

Model creation and synthesis were performed by the phonetics party. Essentially, the model is based on a schematic of the human vocal tract with a particular emphasis on the glottis, which is similar to the *Glottal Excitation System* approach taken by Bocklet et al. (2010) in their age and gender recognition system. Using an interactive manipulation software called *VocalTractLab* (Birkholz, Kröger, & Neuschäfer-Rube, 2011), findings from literature were transferred to the model, with a focus on the long-term features pitch, jitter, and shimmer. The exact configuration was also subject to perceptual optimization. The samples that were synthesized consisted of 36 simulated ages (using the aforementioned model) falling into the three classes *Young male*, *Adult male*, and *Seniors male* (using the same as the FRISC class boundaries, see Table 4.1 on page 104). In order to not escalate the task onto the prosodic and linguistic layer, three different diphthongs were selected for the speech material.

The FRISC system was applied to each of the samples independently using the exact classifiers that were trained as part of the main experiment series on natural voices, and which achieved the best performance (see Table 4.5 on page 132). The models included the *Children* and *female* classes, which were not part of the evaluation data and thus introduce an additional challenge for the system. Since these classifiers use MFCC features, there was no direct relationship between the generating and the discriminating component. Two kinds of observations were made with respect to this data: First, the scores were basically meaningful. There was a clear relationship visible in the data between the scores and the condition, overlaying the contents. Also, the gender was picked up correctly in almost all cases, i.e. the female (and children) classifiers consistently produced lower scores than the male classifiers. Second, the absolute performance was considerably worse than in the evaluation on human voices. With an average accuracy of 29% (best class value of 54% for adult samples), the system is still well above chance level for seven classes (14%), but also far below the levels reached in the main evaluation. This is not really surprising, because the divergence of features between training and evaluation, seen from a signal processing point of view, is tremendous. The variable with the greatest influence on the scores in general is the fundamental frequency.

In contrast to the system performance, the performance of human listeners fulfilling a similar task is considerably higher with an average accuracy of 61% at a chance level of 33%. This number can serve as a confirmation of the basic appropriateness of the phonetic models. It should also be noted however that, even though the samples were randomized, humans

⁶A publication of full details is expected in the near future (Lasarczyk, 2012).

have the ability to dynamically adapt their models to the data while they hear it (i.e. they can decide according to the current and all previous samples), which the system does not, and which might also have an impact on the ratings.

As a conclusion, we can take note that even the artificial voices can produce age cues that are picked up by the system. The reference to this study is intended as a pointer into an area with high potential to reveal more insight into Speaker Classification systems. In order to make further progress, a more sophisticated and extensive study would be needed. In this case, the production of synthesis material can also be considered a limiting factor, since large amounts would be needed to simulate real text-independence. With the possibility of more interdisciplinary studies according to the analysis-by-synthesis paradigm, this seems to be a realistic goal.

5 An Efficient Framework of Applied Speaker Classification

A large part of speech-based experimenting, i.e. the task when analyzing speech data, exploring methods and new ideas, and putting together expressive numbers and figures, is very practical work. Accomplishing this involves a lot of repetitive tasks that are data-intensive and that can only be performed by a computer. There are at least three general directions one can take here: One is to create a custom solution that fits the task best, but requires a lot of so-called “boilerplate” code to be written. This is code that does not solve the actual problem, but supports the methods used to answer the question, and that is typically tedious to write because it consists mainly of routine matter. Such little conceived solutions also lack a far-sighted design and can quickly run into issues because of a lack of scalability: Resulting from deficits in architectural design, new concepts are rather realized via modification than via extension, which can turn into an “engineering nightmare” when the experiment design requires switching between and comparing previous and new methods. For those parts that deal with more sophisticated functions and algorithms, it can further become quite a challenge to create the required implementations. Moreover, it has to be ensured that they work as intended and are compatible to what is described in other literature. And last, a custom-forged module does not easily translate into a robust system. As time is a valuable resource and reusability is not a goal of custom implementations, good programming practices tend to loose priority, which fosters errors and reliability issues, and can even result in considerable efforts spent into a complete re-organization of the whole system.

The second option is to look for existing tools that can provide the required functionality. In practice, for complex tasks, more than a single tool will be necessary to facilitate a given functionality. While writing boilerplate code is kept to a minimum in this approach, there are other disadvantages to this choice: First, using different tools requires an integration effort that can sometimes weigh out the cost of re-implementing the method, or result in a scenario similar to the previous one. When the tools stem from different sources, integration may fail because of compatibility issues with respect to hardware or software requirements, data exchange formats or the interpretation of data. Then, some tools are not provided with source code, meaning that when some functionality is missing, an alternative has to be found. Even if they are, it is often just not feasible to start dismantling them, because learning how they work and how they are structured has its own challenges.

Finally, the third possibility presumes the existence of a single framework that can be used for all of the experiment-related tasks. Building on a single resource creates a certain dependency, therefore the framework in question has to be very flexible to cover the widest field of applications possible, yet specific enough to offer solutions to the most common problems. In addition, it needs to be extensible by means of scripting or an add-in mechanism, so that missing, often very specific functionality can be worked around by adding small pieces of code. If such a framework exists however, there are clear advantages to using it: The handling is much more homogeneous than with individual tools, and all parts of the system cooperate more seamless. Information representation is clearer and redundant representations can be avoided. Overall, there can be gains in the areas of usability, stability and also performance, although of course this still depends on the individual framework as well.

At the time AGENDER was created, there were no existing frameworks that could cover the needs that emerged. The chosen approach was mostly a combination of custom methods and tools. However, in the course of setting up the preliminary experiments that should form the basis of FRISC (see Chapter 2), it became clear that all the novelties introduced and the numerous experiments that had to be performed required a new design. Some of the components used reached their limitations both with respect to functionality as well as the amount of data they could handle, and the whole approach would just not scale. Neither was the design able to handle our own requirements anymore, nor would it be usable by third parties wanting to adjust parameters of the classifiers later on or re-train with custom data, which was a request commonly encountered. This type of endeavor is also more difficult to realize with a collection of loosely-coupled tools, since they expose the end user (the industry partner in this case) to the unnecessary complexity of the internal workings and are more likely to fail under different system configurations.

Many of the Speaker Classification systems surveyed in Chapter 3 refer to applications with real-time requirements, but none of them did provide a solution that is actually created with this aspect in mind. There is no attempt to optimize for efficiency or scalability known to the author in the area of age and gender classification. One of the reasons is that such optimizations are rarely feasible as an “add-on” to an existing approach. Rather, they require careful planning and an architecture that supports it from ground up. Since such an architecture takes quite some effort to develop, it is purposeful to concentrate on a platform that can be reused in other projects.

As a consequence, the author decided to start the development of a new framework that was particularly made for speech-based classification tasks. As of this writing, the number of frameworks and environments that are available for this type of task is increasing, with each framework having its strengths and weaknesses, yet none sufficiently complete with respect to our requirements. Section 5.1 motivates the same, as well as the overall reasoning behind the framework and its design plan, by presenting a formal list of scenarios and requirements. Section 5.2 presents the architecture of the overall framework, while Section 5.3 continues with a description of the development environment (IDE), detailing the individual aspects that were considered for the design of the framework and a discourse on the various areas it provides support for. The aspects of embedded modules are given in a separate Section 5.4, as they reach beyond the actual IDE. Section 5.5 returns to FRISC and describes in detail how the framework was used to set up the main experiments described in the previous chapter. Finally, Section 5.6 contains a set of technical benchmarks and performance measures that emphasize the claim of the framework to be suitable for working with the data intensiveness that speech represents.

5.1 Scenarios and Requirements

The conceptual design of the SPEACLAP framework, which is short for *SPEAker CLAssification Platform*, was guided by several usage scenarios. All of them were immediately relevant

to the author for either their own research activities or for collaborators from science and industry. They were:

- **Experimentation.** Setting up and running experiments can without doubt be considered the primary use case. Researchers in the field of audio-based classification and signal processing are among the persons that are most likely to make use of this functionality. Tasks supporting this scenario are the management and basic analysis of corpus data, extraction and storage of features computed on audio files, training and evaluation of classifiers, and the execution of complete experiments with a predefined parameter set. The typical workflow is depicted in Figure 5.1 (here including module deployment).
- **Deployment.** Resulting from close cooperation with the industry in a number of projects, the deployment of classification technology has always been an important point. As long as the pattern classification happens within the development platform, it usually can be adjusted easily, but suffers from the framework's runtime overhead with respect to speed and memory footprint, and cannot be integrated into applications and other platforms very well. A deployment (or runtime) version of a classifier is a lightweight version that is optimized with respect to the aforementioned criteria. In SPEACLAP, this version is also called an *embedded module*. A vehicle's on-board hardware is an example where the ability to have platform-specific versions is beneficial.
- **Evaluation and Comparison.** Over time, an experiment set-up and also the metrics used for evaluation may change. It can be very helpful if previous versions of classifiers can still be accessed and compared with new ones, even if they are already archived. Similarly, an application designer may have to decide between a number of finished classifiers using a common evaluation instrument and metric. This use case also applies to the role of the evaluation site in a *NIST*-style evaluation, which performs an evaluation on classification results (predictions) instead of actual classifiers and generates the typical statistics such as DET curves and error rates (see Section 2.5.7).
- **Re-training.** The performance of a pattern classification system heavily depends on the available data. When more or better data becomes available, a simple rebuild of the classifier can cause a considerable change in the classification performance. The idea behind this use case is that a re-training can be performed without knowing details of the classifier's internals, such as pre-processing steps, audio features, or classification algorithm. Thus, a classifier created by speech technology experts could be re-trained even by users without background knowledge. Even an adjustment of classes is possible. For example, if a change of age boundaries is necessary after an age classifier is deployed, this would be enabled by doing a re-training and re-build of the module.

The application context is roughly defined as the area of audio-based classification. This includes Speaker Classification as the most important area of Chapter 4 in this work, but also

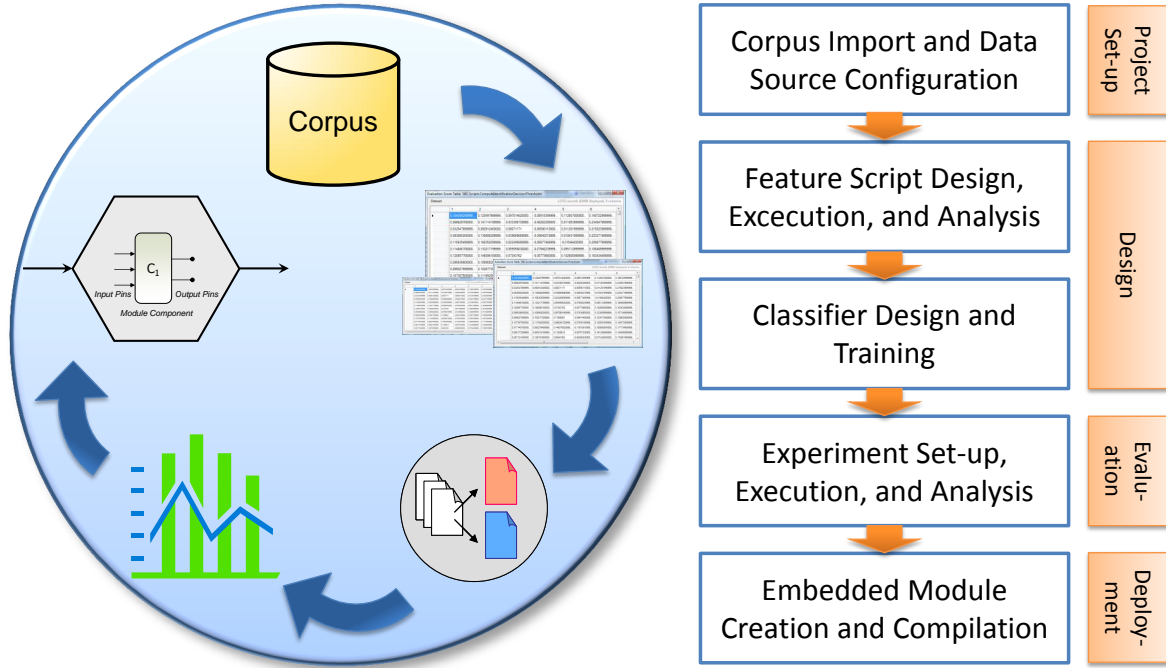


Figure 5.1: The most common SPEACLAP usage scenario *Experimentation* with the addition of a deployment option. The depicted steps are corpus management, feature extraction and analysis, classifier design and training, system evaluation, and embedded module deployment.

speaker verification/identification, speaker diarization, acoustic event detection, and others. To some extent it would even be applicable to speech recognition, although other tools are more tailored to this special purpose. Digital sampled signals that do not represent audio, but are stored in the same way and can be treated like audio, can also be used with the framework. This has been done once as part of preliminary experiments for a gesture recognition task, where the gestures were represented as acoustic signals produced by an electric field-sensing device called *Theremin* (Endres, Schwartz, & Müller, 2011). Here, the raw audio frequencies represented distances from an antenna, which could also be interpreted as 2D coordinates for two antennas.

Resulting from these usage scenarios and the application to the context of audio-based classification, a number of requirements for an audio classification framework aside from the main purpose can be concretized and formalized as follows:

Tailored to Speech. Some existing frameworks are quite powerful, but for the handling of speech data, the user has to resort to external tools, e.g. for file format conversion, resampling, visualization, and listening. A dedicated speech signal processing framework should offer solutions to these and other common tasks related to audio data. For instance, scripts for energy computation, frequency analysis, and extraction of wide-

spread acoustic features should already be built into the platform.

Transparency. The background rule that guided this work has always been to hand over the computation work to the machine, but to involve the experimenter and give feedback at every point where a smart decision can make a difference. For the framework, this translates to giving detailed background information about all processes being performed, allowing intermediate representations of data to be studied, and generally enabling the experimenter to customize as many aspects of the workflow as possible.

Integrated analysis tools. Data analysis is part of almost every classification task. The techniques used for this analysis are recurring and therefore should be included in the framework's toolbox. Examples of such widely used techniques are graphical and statistical analysis functions like the Gaussian distribution analysis, histogram, or correlation analysis.

Extensibility. Even the most extensive framework cannot satisfy all possible demands. This limitation can be circumvented by allowing its functionality to be extended by the user. This is quite different from connecting the application to another application: Using *integrated* scripting functionality, all data remains within the framework, an object model allows custom and application-specific functionality to be combined, and visualizations and other tools can be applied directly. For very popular application, a real ecosystem of extensions can emerge, as is for example the case with *MATLAB*, where scripts for various kinds of problems can be found in the web when needed.

Support for large data volumes. Audio data can become very large, with some corpora encompassing many gigabytes of raw audio data. Even the derived features or structures, when computed on a per-frame basis, are massive. As more data is one of the keys to better results, the framework should avoid imposing any limitations in addition to what the hardware of the user's machine allows. Moreover, methods should be optimized for high throughput and parallel processing. Not being able to process high volumes of data is one of the drawbacks of several other tools, and was one of the most obvious reasons to create the framework. A particular example of how SPEACLAP accomplishes this requirement is the selection of alternate data storage engines.

Common data representations. Solutions involving multiple tools are usually plagued by data representation problems: Each tool requires data to be provided in a different format, which leads to a lot of parsing and formatting code to be written, which in turn costs the researcher valuable time and lowers the system performance. This framework was created with the goal to use canonical formats and structures for all data within the framework, e.g. corpora, meta data, features, and evaluation results. Even the features, which can be physically stored in many ways, use a single *FeatureTable* interface for script-based access. In addition, import and export of common formats such as CSV should be supported.

Results Archival. From a researcher’s perspective, the formal experiment results are probably the most valuable outcome of pattern recognition tasks. Documenting these results properly is therefore critical. A manual documentation can be tiresome, and in long experiment series, single results can easily get lost or be confused over time. The IDE should therefore take care of this process, preserving all data in conjunction with the experiment description in a persistent format. This also makes it possible to follow a development over a longer period of time and also compare results between completely different projects.

Integration into applications. As stated initially in the research questions, the framework is not only about creating classification technology, but also about deploying it. Thus, there has to be an interface that allows the technology to be integrated into applications. Simply using scripting within the IDE or an API to access the IDE externally would only work with a small number of applications. An IDE is generally too monolithic and has a focus that interferes with the demand for seamless integration. Therefore, the framework requires a concept such as that of satellite *modules* which can be integrated into applications, and of a *build process* to generate these modules.

Application-specific classification modules. The modules referred to in the previous paragraph are in turn subject to more specific requirements: (1) They have to be platform-specific to make them run in various environments and to have them take advantage of platform-specific functionality. (2) They should be optimized in terms of runtime performance, i.e. satisfy real-time requirements and work in incremental manner, if necessary. (3) They should be scalable or even resource-aware, which means that they can run both on platforms with limited resources, possibly trading of accuracy versus speed, but can also take advantage of extra memory and processor speed in high-end environments.

Rapid development. Rather a meta-goal, the importance of creating a framework that allows a researcher to quickly set up a classification system for any new task has influenced many other design criteria. One of the main reasons why the custom script-patching approach is often favored in the first place, is because it frequently initially leads to results in less time. If the purpose of a classification task is merely to judge the feasibility of some idea, this strategy may indeed be faster than setting up certain powerful but complex frameworks. Therefore, one requirement to the framework was to be at least as fast as setting up an experiment manually through batch scripts. To accomplish this, the user interface design is of prime importance.

Integrated environment. To support productivity, it was a goal to require the designer to leave the IDE as little as possible. Therefore, all major functions should be accessible from inside the main application. This promotes consistency across the development process, saving time and reducing errors. It is still possible to connect external tools when needed, but this should happen “under the hood”, without the user having to

start multiple programs. Even the impression of multiple “sub-applications” as part of a single framework should be avoided if possible, as is the case with *WEKA*.

Intuitive usage. Another requirement linked to productivity, making the application easy to use and easy to learn was important. One more reason for reverting to custom solutions is the effort involved with learning a new framework. Looking at the existing tools, this concern seems justified. Especially applications created out of the research community show a tendency to be very powerful, highly sophisticated, but difficult to learn, particularly for users outside the community. Unlike many other tools, usability has a high priority for SPEACLAP, in order to make it accessible to a large audience.

This chapter is not so much a description of the internal workings of this framework (although such information is provided, too, when particularly relevant, e.g. for embedded modules), but rather an illustration of features from the perspective of how an experimenter using the framework could benefit most from them. This includes a description of some of the interfaces that can be used to extend the framework with custom scripts.

5.2 Proposed Framework Architecture

Formally, we distinguish between the SPEACLAP framework and the SPEACLAP application or platform. The framework is a superset of the application, which encompasses the context in which it is used, including data access and deployment scenarios. The application (which is implicitly referred to when the term SPEACLAP is used by itself) only reflects the development environment portion, although that is the largest and most important part of the framework.

Figure 5.2 depicts the architecture of the whole framework. We can see that it has aspects dedicated to the design phase of a system (left) and others dedicated to the running phase (right). The application that is used to design and test the system, the SPEACLAP application or integrated development environment (IDE), clearly belongs to the design-time – it is not something that end users will see, neither is it intended to run on the back-end servers in a call center. An expanded view on this component is shown in Section 5.3. The training corpus and feature database belong into the design phase as well, as do any external tools utilized by the framework “under the hood”. A special type of external tool is the SPEACLAP cluster service, which is a service executed on a remote machine to coordinate processes such as feature extraction or classifier training across the network. When the system has been created and the classifiers have been evaluated from within the IDE, it may be time to deploy it into a runtime or “live” context. This is when measures such as speed and scalability start to matter. The IDE is able to spawn so-called embedded modules by compiling them for the target platform. The result is usually a component such as a library, which can be integrated into other applications. There are various “levels” of integration supported by the framework. One involves the use of a client library, which performs the classification off-board, i.e. on a remote server, which is appropriate for low-performance devices. As a special case, the device might also offer a hybrid solution as depicted in Figure 5.2 (lower right), where the

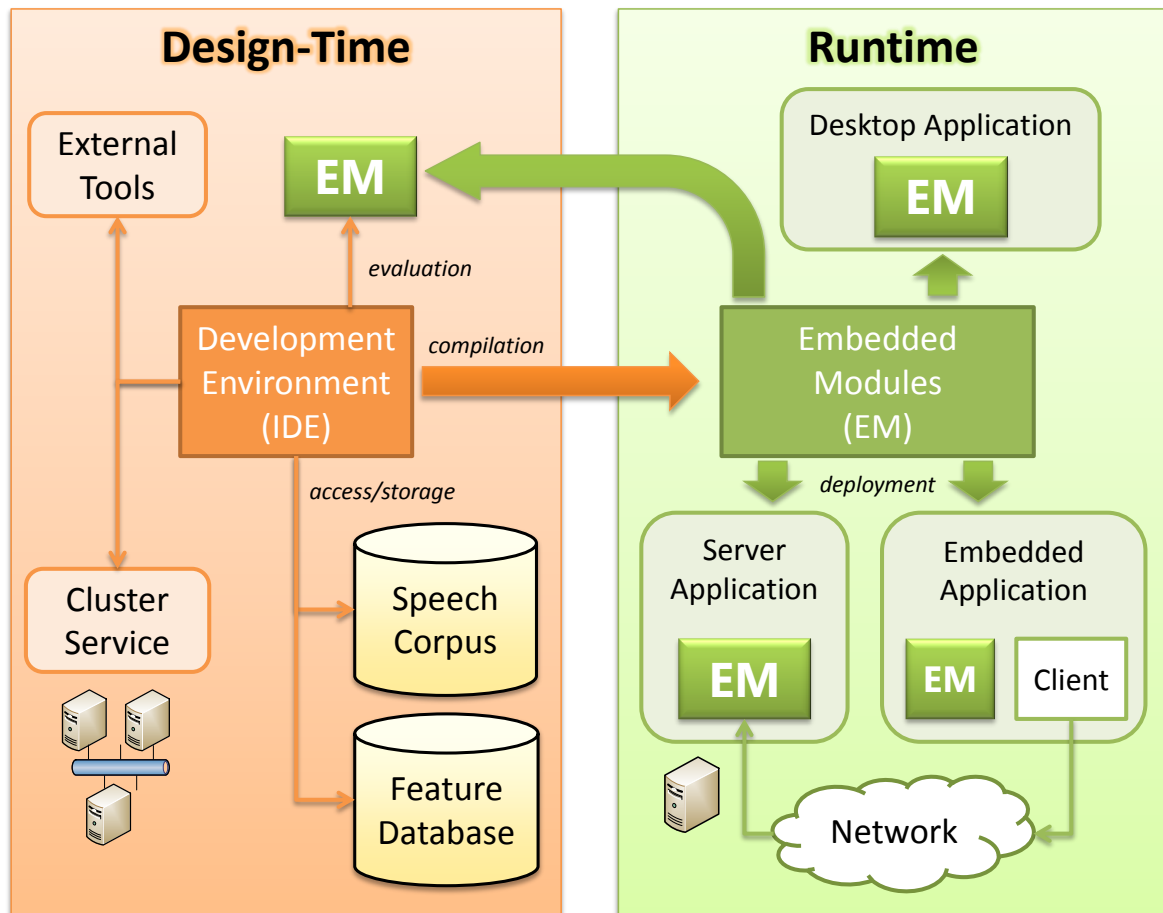


Figure 5.2: SPEACLAP framework architecture overview. For a description, see the text.

embedded module is used as a fallback with lower precision in case that no connection to the server can be established. Last, the compiled embedded module itself can also be evaluated and benchmarked in the IDE.

5.3 The Integrated Development Environment (IDE)

The term IDE refers to the user interface portion or front-end of the framework. It emphasizes that a single parent application hosts a variety of tasks and tools, providing a seamless appearance and consistent interface for users to work with, as opposed to switching between tools and windows. The term is known from other widely used development environments, such as *Microsoft Visual Studio* and *Eclipse*.

Following the usability lead goals of integration and rapid development, an architecture has been created that satisfies the requirements listed earlier. An overview of the various components and sub-systems of SPEACLAP is presented in Figure 5.3. This graphic shows

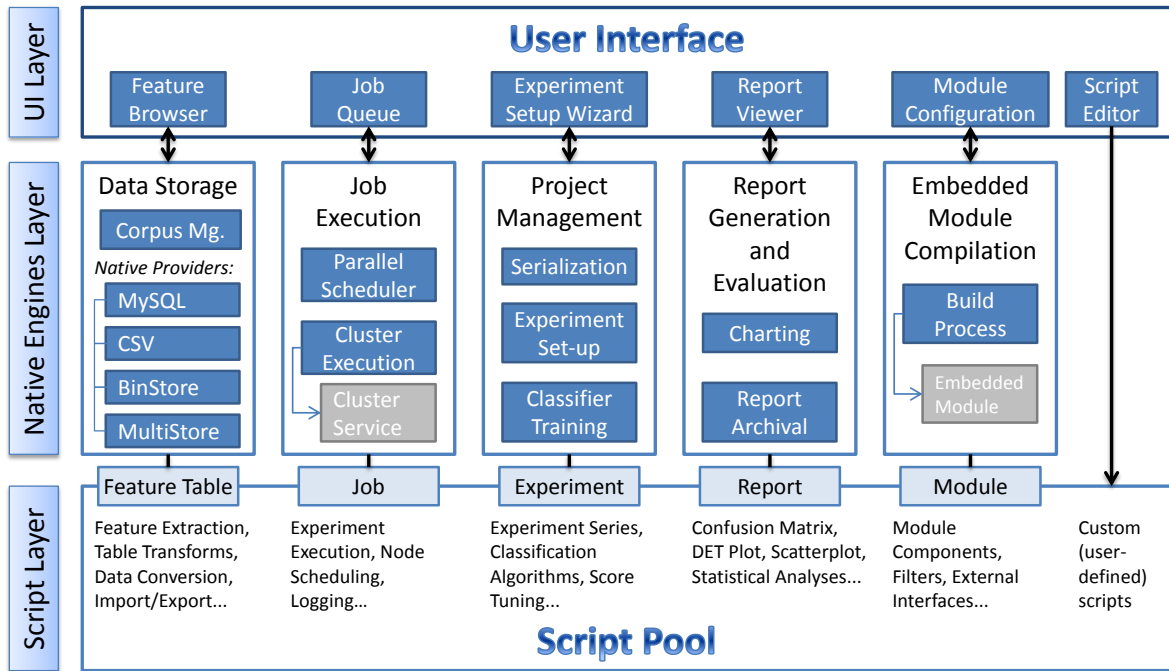


Figure 5.3: Three-layer architecture of the SPEACLAP application. The script layer, which is extensible through custom scripts, is connected with the native layer through interfaces. Displayed UI items, interfaces, and scripts are selected examples.

that the main parts of the IDE can be split into three interacting layers: The user interface layer implements dialogs and visualizations. It is used to access the functionality implemented in the native engines layer. This layer contains the “static” aspects of the platform, such as the logic for project management, data access, and job execution. While these concepts are themselves static, the implementations can usually be extended through the third layer, the script layer. The script layer is connected to the native layer via interfaces, which tell the engines how the scripts should be called. For example, classifiers are called through a classifier interface which describes the methods that a classifier should support. Finally, much of the essential functionality of the IDE is implemented in terms of scripts contained in the script layer (also called script pool). Examples are given in the Figure.

In more detail, the areas of the native layer are:

- the **data storage** engine, which consists of functions for corpus management and feature storage;
- the **job execution** engine, which takes care of running lengthy processes, parallelizing them or running them on remote machines when required while respecting dependencies between them;
- the **project and experiment management** system, which is responsible for basic

functionality such as saving project files, setting up complex experiments, rebuilding on demand, etc;

- the **report generation and evaluation** engine, which is used to display reports on data, e.g. corpora, features, or classifier scores generated by an evaluation run;
- and the **embedded module compilation** engine, which implements the complete build process that is used to generate modules for deployment with the aid of scripts.

5.3.1 General Features

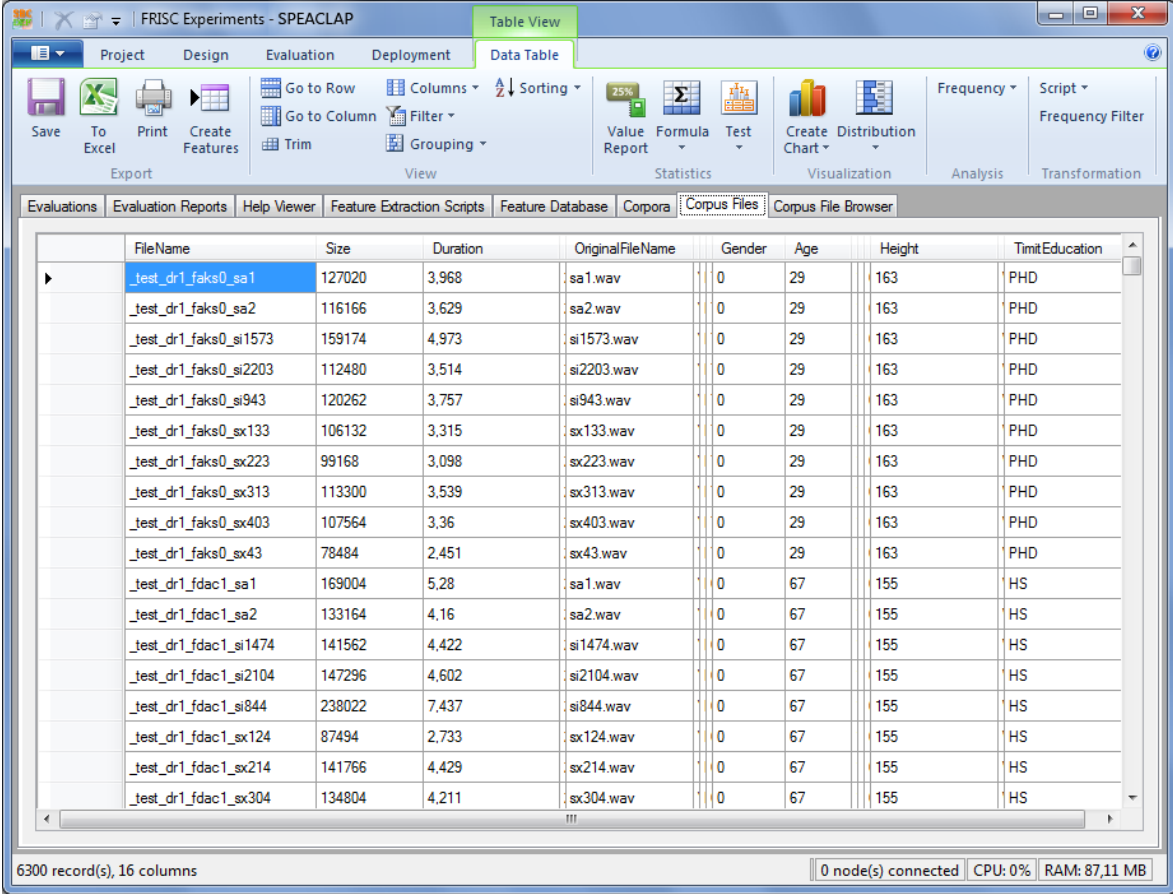
In this section, several basic, task-spanning characteristics of the SPEACLAP are introduced.

User Interface

The first comprehensive overview of the IDE was given in [Feld and Müller \(2009\)](#). After that publication though, the IDE received a major user interface upgrade, both improving usability as well as giving it a more modern look and a similar experience like with other current applications. Figure 5.4 shows the main screen of SPEACLAP with a table containing corpus meta data from *TIMIT*.

One of the noteworthy interface changes that went along with this update is the use of a tabbed document environment instead of a multiple document interface (MDI). Most modern IDEs and even web browsers use this style, where each document or document-like window is represented by a tab in a tab list. Tabbed documents make it much easier to navigate between documents than using multiple windows, which can occasionally overlap or even fully occlude each other. Also, MDI requires the user to keep adjusting the layout all the time by resizing and moving the windows, while tabs always use the full window space. The only downside of tabs is that users cannot see several tabs simultaneously, which they may need e.g. to perform drag-and-drop operations. It has been discovered that in SPEACLAP, such cases are rather rare. Even if it should become a constraint at some time, the appropriate approach nowadays would be to add split tabs (i.e. the window is split into two tabs) or docking tabs (tabs that can be docked to the outer part of workspace, which does not change when navigating).

A second new interface concept is the *ribbon*, which replaces the traditional pull-down menus and toolbars. It has become widely known since its introduction in *Microsoft Office 2007*, and has since been adopted by several other applications. It is focused very much on usability, combining the strength of menus and toolbars alike: Like menus, it allows categorization and provides labels for most items. Like toolbars, it shows icons and allows visible commands to be executed with a single mouse click. The core design principles of the ribbon can be summarized as follows: (1) Although seemingly intuitive, icons alone do not help people much in finding functions ([Wiedenbeck, 1999](#)). Internal studies conducted at Microsoft have confirmed that only few people remember more than approximately eight commands in a program by the icon alone ([Harris, 2005](#)). Usually, the text (e.g. in form of a tool-tip when hovering over the command) is used as confirmation before clicking on



The screenshot displays the main window of the SPEACLAP software. The interface features a ribbon menu at the top with tabs for Project, Design, Evaluation, and Deployment. The 'Evaluation' tab is active, showing a 'Data Table' view. Below the ribbon is a toolbar with various icons for file operations, viewing, and analysis. The main content area shows a table with 16 columns: FileName, Size, Duration, OriginalFileName, Gender, Age, Height, and TimitEducation. The table contains 6300 records. The status bar at the bottom indicates '6300 record(s), 16 columns', '0 node(s) connected', 'CPU: 0%', and 'RAM: 87,11 MB'.

FileName	Size	Duration	OriginalFileName	Gender	Age	Height	TimitEducation
_test_dr1_faks0_sa1	127020	3.968	sa1.wav	1	0	29	PHD
_test_dr1_faks0_sa2	116166	3.629	sa2.wav	1	0	29	PHD
_test_dr1_faks0_si1573	159174	4.973	si1573.wav	1	0	29	PHD
_test_dr1_faks0_si2203	112480	3.514	si2203.wav	1	0	29	PHD
_test_dr1_faks0_si943	120262	3.757	si943.wav	1	0	29	PHD
_test_dr1_faks0_sx133	106132	3.315	sx133.wav	1	0	29	PHD
_test_dr1_faks0_sx223	99168	3.098	sx223.wav	1	0	29	PHD
_test_dr1_faks0_sx313	113300	3.539	sx313.wav	1	0	29	PHD
_test_dr1_faks0_sx403	107564	3.36	sx403.wav	1	0	29	PHD
_test_dr1_faks0_sx43	78484	2.451	sx43.wav	1	0	29	PHD
_test_dr1_fdac1_sa1	169004	5.28	sa1.wav	1	0	67	HS
_test_dr1_fdac1_sa2	133164	4.16	sa2.wav	1	0	67	HS
_test_dr1_fdac1_si1474	141562	4.422	si1474.wav	1	0	67	HS
_test_dr1_fdac1_si2104	147296	4.602	si2104.wav	1	0	67	HS
_test_dr1_fdac1_si844	238022	7.437	si844.wav	1	0	67	HS
_test_dr1_fdac1_sx124	87494	2.733	sx124.wav	1	0	67	HS
_test_dr1_fdac1_sx214	141766	4.429	sx214.wav	1	0	67	HS
_test_dr1_fdac1_sx304	134804	4.211	sx304.wav	1	0	67	HS

Figure 5.4: The main window of SPEACLAP, which consists of the “ribbon” menu/toolbar unit, the tabbed documents interface (showing some open tabs and the *Corpus Files* tab active), the main document content (in this case a corpus table view), and the status bar containing e.g. system metrics.

an icon. Therefore, the ribbon uses much of its space for labels shown in addition to icons. (2) The grouping of commands into a few major topics (called ribbon *tabs*), which are in turn structured into sub-topics laid out across the same ribbon tab (called *group*), and the initial visibility of the default tab, aid in discovering commands that would otherwise be hidden deep inside complex menu structures. It also makes the interface more accessible to new users, i.e. the learning curve is less steep. The topics are most effective when chosen in terms of tasks. In case of SPEACLAP, the tabs have been designed according to the four task categories *general project management*, *system design*, *evaluation*, and *deployment* (see Figure 5.5). There are also some context-sensitive tabs, which appear only when a specific type of document is active. (3) The ribbon follows the paradigm of never removing commands or sorting them in a different place (with the exception of contextual tabs). One of the most frustrating user experiences is not finding an item in the place where it used to be before,

e.g. because some function is disabled or the system is in a different mode. Commands can be disabled on the ribbon, but the groups always stay intact. (4) Traditional toolbars cannot very well accommodate different screen sizes, since the buttons are static. The only option is to change the number of buttons shown, which is not very comfortable when the overhanging controls have to be accessed through a special “extension” menu. The ribbon, on the other hand, can flexibly readjust its layout to fit the same commands onto a smaller space. It does so by changing icon sizes and layout flow, and can selectively turn off labels. It does so in a prioritizing manner, i.e. it saves space on the least important commands first.

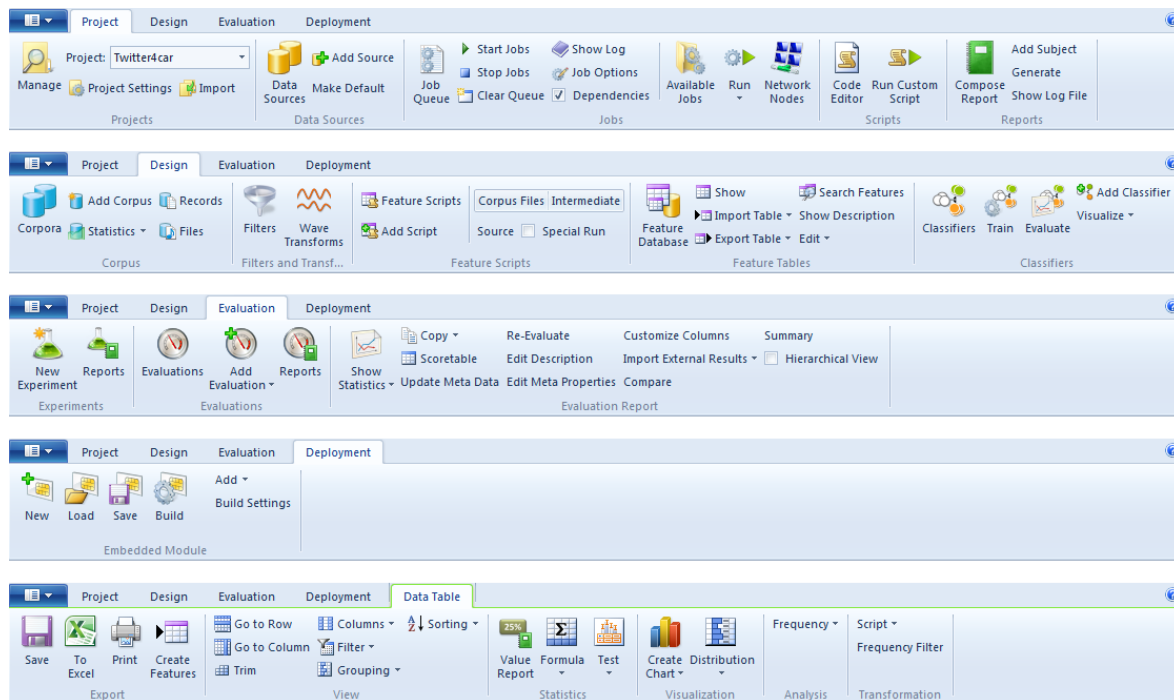


Figure 5.5: All four basic ribbon tabs of SPEACLAP (*Project*, *Design*, *Evaluation*, and *Deployment*) in fully expanded state, as well as a contextual tab (*Data Table*) that is active when data in table format is displayed. Further contextual tabs are *Code Editor*, *Text Document*, and *Chart*. Access to settings resides in the “application menu” (not shown) accessed by the button in the upper left corner of the ribbon.

A disadvantage of the ribbon is its higher space consumption when expanded, and the inability to show commands from several tabs at once. However, it also allows some frequently used commands to be added to a “quick access area” in the title bar for immediate access.

Project Management

The basic unit that ties together the parts related to a classification task is called a *project*. A project could refer to an actual project, or relate to a speech property that is being researched. It can also be reasonable to have a project for a specific set of source data on which a multitude

of experiments are run. Each project consists of its own data, feature extraction scripts, classifiers, experiments, evaluations, and embedded modules. All of this information is stored in an XML-based project file, which is maintained by the application. It is possible to import complete projects (this is referred to as *attaching* the project) or single objects from other projects. Each project also has some project-specific settings, such as output paths, that can be configured. All of this can be done from the project management tab.

A typical classification experiment consists of the steps outlined in simplified form in Figure 5.1: First, a new project is created (except when the experiment is a follow-up to another experiment, in which case the existing project can be used). For new projects, some data storages have to be configured to tell the application where it can store features etc. Next, the corpus data is organized. This means that all audio samples that will be relevant to the project are selected and labels are stored in a database. This process is also called corpus import. Then, feature extraction scripts can be configured and run. This will build up a feature database, which can be further analyzed using the integrated statistical tools. Afterwards, classifiers can be created based on any combination of features. Some classifiers can be tested quickly, but more complex systems are evaluated in the following step. In this step, an experiment can be set up, which will produce evaluation reports each time it is run. These reports can again be analyzed to perform tuning and repeat the experiment with different parameters. Finally, an embedded module can be created to integrate the classifier into other applications. The ribbon shown in Figure 5.5 was created in a way that follows exactly this ordering of tasks. Of course, there are also several optional steps that can be undertaken at each stage, such as the configuration of corpus file filters.

Sometimes, the same project should be used simultaneously on multiple machines, e.g. to run different experiments. For this case, projects can also be opened in “real-only” mode. This allows the project to be shared without causing conflicts.

Extensibility through Scripting and Plug-Ins

SPEACLAP heavily relies on the concept of scripts to facilitate its extensibility ambitions. Many kinds of operations allow different implementations from which the user may choose one, and which should be extensible without modifying the IDE’s source code. Whenever such an operation was encountered during the development, a script interface was created that specifies the contract for the script function. At runtime, the user is able to select the actual script to be used from all available classes that implement the corresponding interface. An example of a very generic script interface is the experiment (the interface is named `IScriptJobCreator`), which can generate a list of jobs that make up a single experiment, including their dependencies. All script interfaces employed throughout the framework are listed in Table 5.1.

There are many scripts already included in the application, but the user is not limited to these scripts. There are two ways to add new custom scripts: through the integrated code editor shown in Figure 5.6 or as an external assembly. Using the code editor is a

Interface Name	Description
IScriptAggregationFunction	General aggregation function that aggregates several numeric (<i>Double</i>) values. Examples are mean, max, and standard deviation.
IScriptAudioConversionTool	Represents a (usually external) tool to convert files from one format to another. This allows new audio file formats to be supported. For example, an integrated script uses the command line utility <i>SOX</i> to convert between several formats.
IScriptAudioFeatureExtraction	Script for feature extraction from raw audio data, which is provided as an array of 32-bit numbers. This can be used for simple functions, such as determining the energy in the signal.
IScriptClassifierFeatureTable-Exporter	Exports a feature representation of a trained classifier. For example, a GMM classifier could be exported as pairs of mean and variance for each Gaussian.
IScriptClassifierScoreModifier	Used for score optimization. Contains one method to compute a score modification structure (e.g. threshold) on the provided scores, and another method to apply that structure to the provided scores to modify them. It can also be considered a simple type of classifier.
IScriptContentSerialization	Applied to classes which require a special type of serialization, such as classifiers with very large models. Using this interface allows the data to be serialized into multiple files, for example.
IScriptCorpusMetaDataReader	Imports existing meta data for corpus files. This is used to support existing corpus formats, e.g. <i>TIMIT</i> or <i>GlobalPhone</i> .
IScriptCustom	Implements a generic script that can be run without a context.
IScriptDataTableExporter	Exports arbitrary data tables to another file format to be used with third party applications.
IScriptDataTableImporter	Imports data tables from another file format to be used with SPEACLAP, e.g. as feature tables.
IScriptDynamicMetaDataGenerator	Generates meta data for corpus files. This is often used to define classes for numeric ranges, e.g. to convert the 'Age' of a speaker into an 'AgeClass'.
IScriptEvaluationPerEvalHook	A script that is executed once at the end of an evaluation to produce additional results.
IScriptEvaluationPerResultHook	A script that is executed for each file during evaluation to produce further statistics.
IScriptExperimentSeries	Sets up the workflow for multiple experiments with different parameters, including optimizing series such as hill-climbing.
IScriptFeatureExtractor	Produces features based on corpus files (audio data) or other (intermediate) features.
IScriptFeatureScriptOutputParser	Parses the output of command scripts for feature extraction into a default table-based format.
IScriptFeatureTableExporter	Exports feature tables to another file format to be used with third party applications.
IScriptFeatureTableImporter	Imports feature tables from another file format to be used with SPEACLAP. Using this interface eliminates many size limitations.
IScriptFileScoreProvider	A script that is used to compute scores for each file in a corpus. This can be commonly used in balancing, where corpus files are assigned scores based on their length.
IScriptItemListFilter	A filter that is applied to a list of string values and produces a list of 'included' values. Item list filters can only be based on the values of a single column (the filter column). A common example is the corpus file filter, which is applied to a list of corpus files.
IScriptJobCreator	Creates a custom set of jobs for a specific task. This is mainly used to configure experiments.
IScriptLiveItemFilter	A filter that is applied to a single instance to determine whether the instance is included in the selection or not. Live filters are sometimes slower than Item List Filters and not applicable if the whole list of items has to be known, but require less memory and no extra table iteration.
IScriptModClassifier	A classifier that can be used in an embedded module.
IScriptModComponent	A generic component in an embedded module. Some other interfaces inherit from this.
IScriptModFeatureExtractor	A feature extractor that can be used in an embedded module.
IScriptModInterface	An additional external interface for wrapping an embedded module, e.g. a Java interface.
IScriptModPostProcessor	A post-processor that can be used in an embedded module.
IScriptMultiLabelClassifier-WinnerDetermination	Is used in conjunction with multi-label classification results provided as class-specific scores to determine the winner. The default is to choose the class with the highest score.
IScriptNetNodeAssignment	Used to assign cluster nodes to commands when job execution is performed on remote PCs.
IScriptNormalization	Used for normalization of values. Contains one method to compute the normalization data on a set of samples, and another method to apply the structure to a new value.
IScriptParameterScript	Converts external feature extraction script parameters from one value to another before they are sent to the script.
IScriptReportExporter	Exports certain types of custom report results. For example, charts can be saved as bitmaps.
IScriptReportGenerator	Generic interface for scripts that generate any type of report. Each script can decide whether it can work on a specific type of input, and can select between different types of output it can provide (text, charts, images...).
IScriptTableModification	Modifies the structure or contents of a data table. Can be used to pre- or post-process features.
IScriptValuePlotter	Creates a chart based on a set of numeric (feature) values. Some features may have a particularly useful visual representation.

Table 5.1: List of all script interfaces implemented in SPEACLAP. Each interface allows the extension of the framework through writing of user-defined scripts.

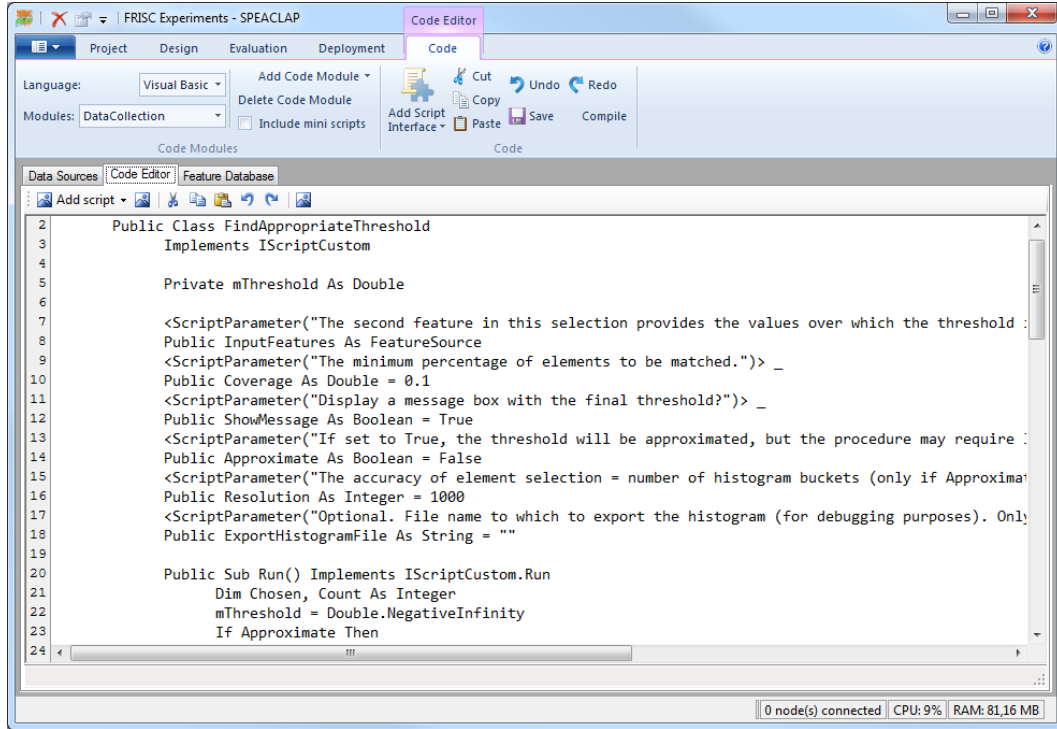


Figure 5.6: The integrated code editor of SPEACLAP.

very comfortable way of extending the framework. Each project can have one or more code modules, which can be edited directly from within the IDE. All the user has to do is select the desired interface from a list and fill in the functions that are created as part of a script template. The code can be written in either C# or Visual Basic .NET, which is directly compiled. It is even possible to write temporary scripts “in-line” at the time when the script is needed, i.e. from the script selection dialog, such as a small value conversion function. In case of longer scripts, the code editor may not be the best choice in terms of comfort, as it is missing IntelliSense and other supportive functions known from major programming environments. Therefore, it is also possible to write scripts with an editor such as Visual Studio, compile it, and put the resulting .NET assembly in a special “plug-in” directory that is scanned on startup. The experiments described in the previous chapter were set up this way.

A lot of scripts are parameterized. SPEACLAP will automatically discover all public fields of the class containing the script and list them as parameters that can be configured in the same dialog where the user also selects the script. It is even possible to provide customized editors for parameters of a special data type.

Listing 1 shows an example of a feature pre-processing script. The basic outline (class definition and method frame) was created by the code editor after the `IScriptFeatureTransform` script interface has been selected. The remaining code was inserted by the user

Listing 1 Sample feature pre-processing script for trimming value ranges.

```
class TrimmingScript
    implements IScriptFeatureTransform

    public double Range = 10.0

    function double Transform(double OldValue)
        if OldValue > Range then Return range
        else if OldValue < -Range then return -Range
        else return OldValue
    end function
end class
```

and demonstrates a simple trimming that can be applied to any numeric feature values. The public field `Range` is the script's only parameter.

Job Execution Engine

In machine learning, we often encounter processes that need a considerable amount of time to complete, either because they involve large amounts of data or because they are computationally expensive. In SPEACLAP, there are two concepts to mitigate the implications of this: The first is that options for clustered computing are an integral part of the system, which allows time-consuming computations to be distributed to other machines. The other concept is that of job queue management, which allows the user to configure and plan several processes in advance and then run them sequentially.

Processes that may potentially be lengthy are implemented as *jobs*. Common examples for jobs are feature extraction, classifier training, classifier evaluation, and embedded module building. Jobs can be added to a global queue, provide status information about their progress and have an associated rich-text log containing messages, warnings and errors. The job execution engine is responsible for executing the jobs and for distributing the commands to different machines on the cluster (nodes). On a single machine with multiple processors, the job engine can also schedule multiple jobs to run in parallel. While the job itself is an integrated program that is run inside the IDE, the job can create *commands*, which can be executed independently on remote nodes. There are different types of command, which represent various script calling interfaces that will be further discussed in conjunction with feature extraction scripts in Section 5.3.3, such as executable. Currently, executables are the only commands that can be run remotely.

Remote execution is facilitated through the SPEACLAP *Cluster Service*. This service component implemented in .NET is essentially a TCP server and platform-independent. It has been successfully employed on a Linux cluster in conjunction with the Mono runtime. It can

be automatically installed and started from within the IDE over an SSH connection. Once the service is running, the node is available to SPEACLAP. A global setting specifies the number of nodes to be utilized. When the job execution engine receives new commands, it waits until a node is idle and then sends the command to that node. The result of the command is returned to the engine using a backchannel in the service.

As operating system and file paths may differ between the IDE and nodes, command lines cannot be transferred 1:1. Instead, *command packaging* is performed. For this to work, the commands created by jobs must be composed of individual segments. Segments that refer to corpus files, temporary files, output files or other special items that have to be adjusted for the remote system must be added as a special type of segment. This way, both the job engine and the remote service can perform the required adjustments to the command line, such as replacing a *temporary file* argument with a newly generated local temporary file.

SPEACLAP allows the user to view and control the status of jobs in the customizable job queue tab (see Figure 5.7). In this tab, jobs can be started and stopped, logs can be studied, and settings can be changed. The settings include the number of processors used and the amount of output generated. It is also possible to have all logs sent to a specified email address when the jobs have finished, which can be helpful with lengthy experiments. Each job reports a status in terms of current operation, running time, percentage completed, generated warnings and errors. It is up to the implementation of the job to provide meaningful values for each of these (except for the running time). For jobs that run external commands, the state of the command queue is also logged.

New jobs can be created using one of several methods: (1) By selecting the job in the *Available Jobs* tab and selecting *Run* or *Enqueue* on that tab. *Run* will simply execute the job immediately, while *enqueue* adds it to the current job queue and allows it to be executed later together with other jobs. To understand how the *Available Jobs* list works, one has to know that many of the objects in SPEACLAP have jobs implicitly associated with them: A classifier has an associated training job, an evaluation has an evaluation job, an experiment has a whole list of associated jobs etc. The *Available Jobs* list actually scans through all objects in the current projects which provide such implicit jobs. (2) By selecting an object directly and then choosing the *Run* or *Enqueue* commands from the ribbon. For example, if a classifier is selected in the *Classifiers* tab when *Run* is pressed, that classifier will be trained. (3) Using a method specific to the type of selected object, e.g. choosing *Train* from a classifier's context menu. (4) Through code from within a script.

Between some jobs, dependencies may exist. For example, a classifier training job may depend on a feature extraction job that computes the features used in the training. Taken together, a job queue represents a dependency graph, where each job node can have a dependency on one or more other nodes. This graph imposes an ordering in which jobs must be executed. This ordering is especially important when parallel execution is used, because it can sometimes prevent that the maximum number of parallel processes is started. SPEACLAP does only start jobs when all incoming dependencies are satisfied. Some dependencies are automatically detected by the IDE based on the data source, while others have to be specified

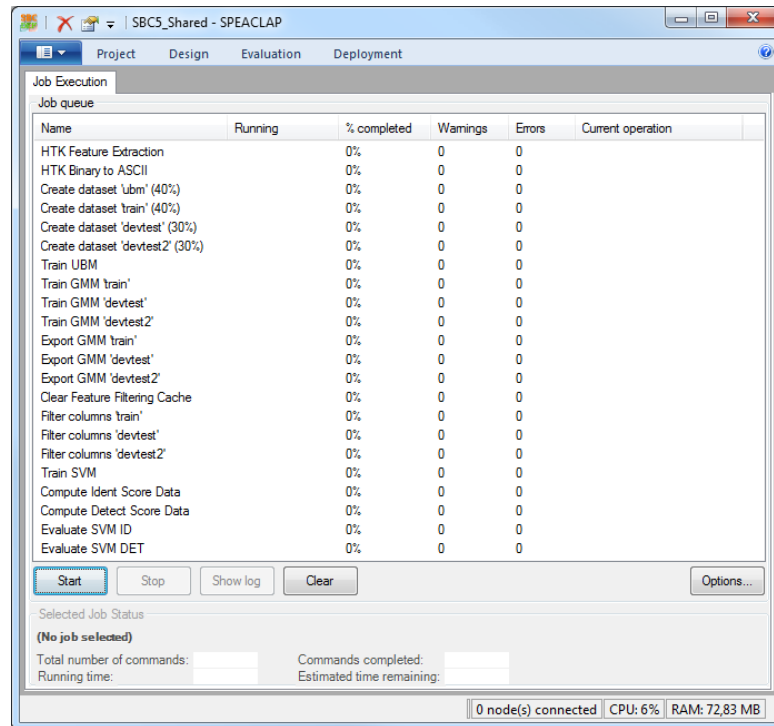


Figure 5.7: The job queue window lists all jobs with their current status.

in the experiment definition.

Each job can also have child jobs, which it can create at any time during execution. In the job queue, these are indicated by indention. When one or more child jobs are spawned, the execution of the main job is deferred until all child jobs have completed. The advantage of this mechanism is that the child jobs will also be parallelized if possible. An example is the training of a multi-label classifier, where each class can be trained separately.

Experiments and Experiment Series

An *experiment* in the notion of SPEACLAP is a script that generates a list of jobs according to a specified configuration. It usually combines all stages of a pattern recognition task and ends with the creation of several report documents. Instead of running feature extraction, classification, and evaluation manually one after the other, the experiment takes care of this automatically. It can also configure special dependencies between scripts that are not automatically detected by the IDE, thus allowing certain jobs to run in parallel. Experiments are quite useful when multiple parameter configurations should be run with otherwise identical conditions. By using an experiment script, the chance of mistakes by selecting the wrong items or running jobs in the wrong order is considerably reduced.

Experiments can easily be set up using a wizard-style interface called the *New Experiment Wizard* (see Figure 5.8), which presents a selection of experiments, a list of experiment param-

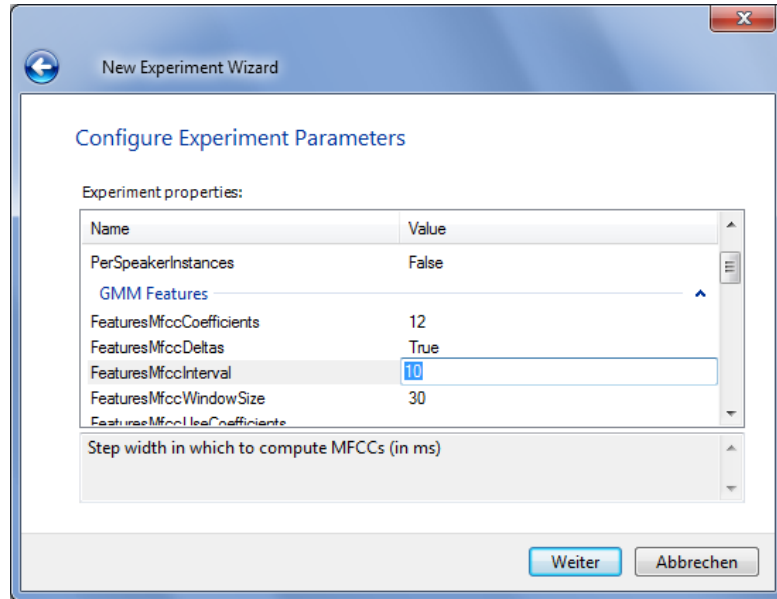


Figure 5.8: The *New Experiment Wizard* allows quick execution of existing experiments with different parameter configurations. This screenshot shows the configuration of the FRISC main experiment series.

eters, and allows title and description for the report to be specified. It also allows a previous experiment to be re-run. If a large number of experiments has to be run, even this comfortable method can become tedious at some point. Therefore, the concept of an experiment series was additionally introduced. Experiment series are special scripts that run experiments with multiple configurations in a specific order. The order can be adjusted dynamically during the experiment. An example is a parameter space hill-climbing implementation: For any number of experiment parameters, such as the MFCC extraction step width, the settings to be examined can be configured. Like in the main experiment series, all parameters will then be explored consecutively. The configuration for the following parameter depends on the best-performing setting of the current one. The criterion, e.g. error rate or accuracy, can be chosen freely.

5.3.2 Corpus Management

Before any analysis can be done, the speech needs to be available. A speech corpus is a database of related wave samples with annotations. Each sample in the corpus is related to exactly one audio file and has a unique file ID. There is no technical definition as to when two files should be in the same corpus (except that they must have the same audio format); it is up to the user and can be freely set up whichever way is most purposeful for the project. For example, one might create corpora based on the origin of the data or the recording conditions.

Audio Formats

One nuisance that arose several times prior to the introduction of SPEACLAP was the absence of a common file format for both the speech corpora as well as for the feature extraction tools, so that the audio files often had to be converted manually between different formats multiple times. To avoid this, SPEACLAP requires a copy of the audio data in raw uncompressed linear PCM format. It can import and convert the files from several popular formats such as WAVE, AU, a-law, μ -law, and NIST Sphere. Also, when a tool requires audio data in a different format, the corpus system will handle the conversion into the required format automatically and keep a cached copy of the files in this format until the disk space is needed. The corpus data can be stored in a shared network location to be usable from multiple projects.

Each corpus has a specific “native” audio format. This format is made up of sampling frequency, bit rate, and number of channels in the sound files. The native format is supposed to be the original recording format and that with the best quality, so this format is always kept. As with the file format, the speech samples can be converted on-the-fly to almost every audio format if requested. This enables the classifier designer for instance to build speaker classifiers for both a local as well as a phone-based scenario with a low quality voice channel by only changing a single setting.

Meta Data

Meta data is the information associated with a single speech sample beyond the actual digitized LPCM signal. There are three types of meta data: Basic data, static data, and dynamic data. Basic meta data are fields like a unique ID, file name, timestamp, and length. This information is present for all corpus files. Static meta data are the annotations or class labels that are provided by an external database and that do never change. Dynamic meta data differs from static meta data only in the way it is computed, which is using a script function that may be based on one or more other fields. For example, the static field *BirthDate* may contain the exact date of birth of the speaker and *RecordingDate* the date when the sample was recorded. To retrieve a representation that is more suitable for analysis and classification however, the dynamic meta field *Age* is created and computed from the other two fields using a simple script. Further dynamic meta data might map the integer *Age* to a nominal *AgeClass*. Values of dynamic meta data fields are cached, but can be updated when the script changes. The dialog that is used to manage corpus meta data is shown in Figure 5.9.

Filtering Corpus Files

Corpus file filters based on meta data represent a convenient way of selecting subsets of corpus files, e.g. for feature extraction or classification. They are implemented as human-readable filter rules that restrict the choice of corpus files by certain conditions.

For example, in FRISC, we often want to consider only some age groups, thus, as an alternative to creating an *AgeClass* meta field, a filter rule to select just adults between 25

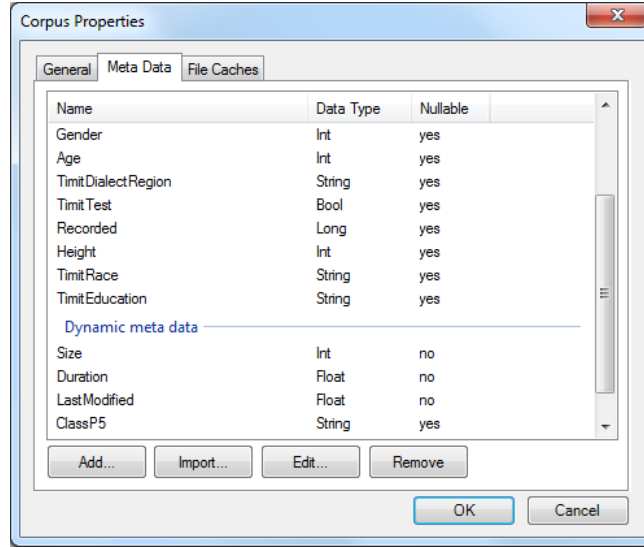


Figure 5.9: *Corpus Properties* dialog. The active tab displays a list of static and dynamic meta properties and their data types, including commands to add more.

and 60 years could be written as:

age > 24 AND age < 61

Corpus Reports

Besides the basic functions for adding, deleting, and configuring corpora, the corpus management view provides some reports. One of them displays the complete meta data table for a corpus. A second report computes statistics for each meta feature, such as the available labels and their share (for nominal data), or range, mean and standard deviation (for cardinal features). It also allows quick listening to any sample in the corpus.

5.3.3 Feature Extraction and Management

Features are the pieces of information on the basis of which a classifier makes its decisions, hence choosing the right features is a key in creating a good classifier. The computation process is called feature extraction. They are usually computed by applying signal processing algorithms to the speech data, but in theory, anything can be used as features, including the output scores of other classifiers. If dynamic meta data is used for classification, it is used as a feature in this sense as well.

Feature Tables

The basic storage unit for features both in memory and on disk is called a *feature table*. It is basically similar to database table, but with a more constrained structure and additional meta

information. A feature table has one column for the primary key and one column for each feature that it contains. The feature columns can be of any of the supported feature types: nominal (*string* or *bool*), cardinal (*int*, *long*, or *float*), or binary. Every row (or record) in the table needs to have a unique primary key. The type of primary key may vary depending on the semantics of the features extracted. In many cases, when one feature is extracted per corpus file, the primary key will be the ID of the corpus file. If features are computed for segments within corpus files, the segment ID must also be part of the primary key. Feature tables also support *NULL* values in their records, which denote a missing feature value. Further, feature tables can have arbitrary meta properties, but a set of defined default properties is always supported, such as their creation time, source audio format, or the script by which they were extracted.

Extraction Scripts

The feature extraction is done by running scripts (see Figure 5.10). A feature extraction script defines where the program or code used for computation is found and how it is executed. SPEACLAP can run executables, integrated and custom scripts. In the future, the addition of further interfaces like PHP scripts and Web Services is possible. The script can be configured in several ways that allow for the quick integration of most existing tools for feature extraction. For executables, command line parameters can be defined, which may also be feature values, script-generated values, temporary files, input files, or output files. Small scripts written within the IDE can be applied to feature values before they are sent to the tool or after they are received again, so that the values can be formatted to the tool's needs. For executables, an output parser can be specified, which reads and interprets the information returned by the tool – either as a text file or on the console – into a table. A default CSV parser is included. Custom scripts implement the `IScriptFeatureExtractor` interface, which has an `Extract` method that returns features already in table format, so there is no parsing needed.

Feature extraction scripts also can have different call semantics, which are determined by the workings of the script and must be specified by the user. The most common pattern is that a script is called for every single record and it outputs a single record. Some scripts however may perform time-consuming start-up operations, but can handle multiple records in one call. In this case, SPEACLAP can also provide all records in one call. A special case is per-class computation, where the script receives all instances belonging to a specific class in each call. Apart from the call pattern, the output semantics can also be different from the input settings. For example, the script that processes all records from one class in a call may also return one result for each record, or it can compute a single record for the whole class. In SPEACLAP, the setting *output grouping* controls the semantics of the output and can be set to *None*, *Corpus file*, *Class*, and *Custom*. As long as the output table is grouped by corpus file, the original file meta data remains linked to the features.

In general, the commands that need to be executed to extract the features are first generated as a single batch and then executed locally or on different machines. A special functionality is

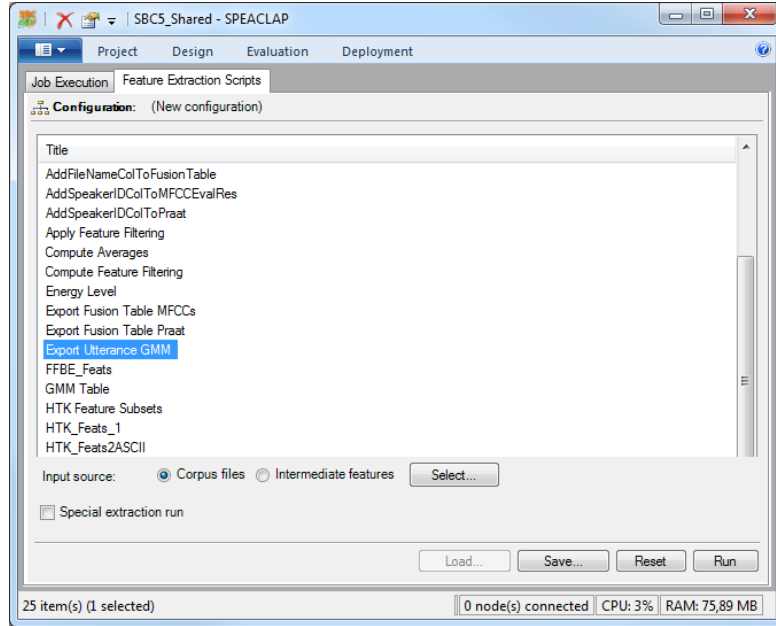


Figure 5.10: Assortment of feature extraction scripts in a project.

delayed input data collection. If one of the input features requires a large amount of memory and cannot be written to a batch file in beforehand, a placeholder is instead created and the feature is only extracted when the command is finally executed. This function works in conjunction with storage providers, which must supply so-called “getters”, i.e. function pointers that can be called to retrieve a single feature value, and which will only occupy the space needed for the pointer in memory.

To perform a feature extraction, a feature extraction configuration is created. A configuration consists of one or more predefined feature extraction scripts and parameters for each script. Parameters can include custom script parameters or settings for command line arguments to the executable. There are also two default parameters for every script, which are input and output. The *input* to a script can either be a set of audio files on disk (i.e. a reference to corpus files), an existing feature table, or the output of another script. When using files on disk, the file format, audio format, meta data-based filter, and additional waveform transformations can be specified. For existing features, it is not only possible to select a feature table present in one of the storage providers, but custom tables can also be composed on-the-fly, which is one of the more powerful capabilities of the framework (see the sections below). Output from scripts is kept in a special *temporary feature store* when it is needed as input to other extractors. Otherwise, a storage provider can be chosen by the user. The order in which scripts are executed is derived from dependencies between scripts. Scripts that are independent can be executed in any order. It is possible to save feature extraction configurations and restore them later. Each table “remembers” the complete configuration that produced it, so that in the future, it will be possible to automatically re-extract features when needed.

Storage Providers

As there can be many files in a corpus and a lot of interesting features to be investigated, the amount of information that has to be processed and stored can become very large. This is especially true when features are computed for small segments of audio data. To handle this volume, SPEACLAP supports multiple feature storage providers. A storage provider can handle and optimize the following aspects of feature access:

- Storage of feature table descriptions (meta information)
- Storage of the actual feature data
- Listing and retrieval of feature tables
- Caching of feature data.

There are several reasons why having multiple storage providers is useful. First and foremost, the performance of a storage provider can vary a lot depending on the structure and amount of feature data and the physical platform used for storage. For example, some storage engines work well with many features (columns) but are not suitable for storing a lot of records (rows). Another example is a database engine like *MySQL*, which works very fast when many small records are stored, but may become rather slow if the records or table size as a whole get too large. To counteract this, SPEACLAP includes a storage provider that writes records to individual binary files and is also suitable for large records. Also, some feature extractions such as n-gram builders do not produce a fixed vector, so they cannot easily be stored in a normal database table. Again, the IDE has a custom provider for this that can handle dynamic feature vectors. Table 5.2 lists all providers that are currently implemented. Another motivation is that storage providers can provide direct interoperability with other utilities by accessing data stored in custom formats. For instance, one of the integrated providers can read and write feature data from and to a CSV¹ file, so it is possible to share this data with other tools that support that format without explicit conversion. Lastly, using the generic interface that is exposed by all storage providers, it is easy to make any kind of other information available to classifiers in the same way as features. Consider corpus file meta information: The **CorpusMetaStorage** provider exposes all meta information as feature tables, so the classifier can use audio duration and other data like any other feature. This and similar providers however do only support reading existing tables, not creating new ones.

Feature Retrieval

Large feature tables do not only pose challenges to storage, but also to retrieval of their contents. Some popular machine learning frameworks like *WEKA* require all features of a table to be kept in memory for processing. This can limit the type of experiments that

¹CSV = comma-separated values. A CSV file is a text file with one record per line. The actual format can vary within certain bounds w.r.t. field separation, quotation, decimal places, and others.

Provider Name	Description
MySQL	Stores features in a MySQL database using one database table for each feature table.
ESENT	Uses the Extensible Storage Engine, which is a native part of Windows.
Binary File Store	Uses a custom fixed-length binary record for storing data. Multiple records can be stored in a single file based on a secondary key to improve performance.
Multi File Store	Stores one record per file, making it suitable for large but few records. Additionally, each file can have its own fields in order to support dynamic feature vectors.
CSV	Stores data in a simple .csv file for maximum interoperability with other tools at the cost of performance.

Table 5.2: Several data storage providers are already integrated into the platform.

can be performed with these tools considerably. Nevertheless, it usually is the fastest way of accessing features. To combine both aspects, feature tables in SPEACLAP support two ways of data retrieval: table-based and enumerator-based. The table-based method reads all records into a single table structure that is compatible with the `System.Data.DataTable` class used throughout the .NET Framework. Because physical access to the data is only needed for a single call, traversal of the table rows is extremely fast, especially because data is kept in RAM (if available). The enumerator-based approach maintains a pointer, which starts at the first record and can be advanced to the next record until all data has been enumerated. As only the current record is kept in memory, this method does not suffer from any memory problems. It is however up to several times slower because the data structures need to be updated for each record, and because multiple physical accesses to the same file may be needed. This is a place where an enumerator can be optimized. For instance, the MySQL storage enumerator always queries 1000 rows in each SQL `SELECT` statements instead of just one. Obviously, the more caching is applied, the more memory is needed again. Another disadvantage of enumeration is that there is no row count available, i.e. the total number of rows is not known beforehand, which may require different formulas for some filtering or feature computation algorithms. One more difference between the two versions is that the enumerator is uni-directional, i.e. records can be traversed only in one direction, while tables allow instant access to any row by its index. Again, this can affect the way scripts work with the data and can also result in a performance decrease.

Each storage provider is responsible for implementing the data retrieval methods. Only the enumerator-based method is mandatory because it is guaranteed to work independently of the table size. Table-based access can be emulated by simply collecting the records returned by the enumerator, although this may not exhibit the best performance possible.

Joining and Filtering Feature Tables

When working with features, it is often not sufficient to use the tables in their original form. Feature tables from physical storages have a fixed layout, which is typically dictated by the tool with which it was created. For example, a feature table created for FRISC using the *Praat* tool would contain common *Praat*-based features such *pitch_mean*, *pitch_max* etc.

Furthermore, the features may be distributed across multiple tables for different corpora or subsets of audio files, because not all files are extracted in one run due to time considerations, for the same reason that not all features may have been extracted for all files. And finally, there often exist multiple versions of the same basic features but extracted with different parameters.

Feature tables are used by several components such analysis tools, exporters, other feature extraction scripts, and classifiers. Consider for example a classifier: It is created by selecting a set of features, which may stem from different features tables. For this purpose, there is a special *compound feature table*, which is accessed like any other feature table, but is dynamically created from other tables. It does not contain any actual data, but rather retrieves and parses the data from the underlying tables on-the-fly while it is being enumerated. It is also possible to build compound tables on top of other compound tables, which is similar to the concept of operator chaining known from *YALE* (Ritthoff, Klinkenberg, Fischer, Mierswa, & Felske, 2001).

SPEACLAP supports horizontal (or feature) joining and vertical (or instance) joining of feature tables (see Figure 5.11). *Horizontal* joining is performed when multiple tables contain different feature of the same instances, while *vertical* joining occurs when several tables store identical features, but differ in primary keys. It is also possible to cope with advanced combinations of both, which may require specification of some rules for conflict resolution. For example, features with identical names may be different or not. Keys for which not all features can be gathered from all tables can be dropped or be complemented with *NULL* values. Joining of feature tables is done incrementally when using the enumerator-based access and with complexity $O(n)$. This is possible because all feature tables are already sorted by their primary keys.

Feature Versioning and Archival

A topic that is sometimes disregarded in other tools is that of data archival. Prior to the IDE, working with features required manual data management and archival, which translates into giving descriptive names to files and database tables, and creating a folder structure. Doing this manually is however both error-prone and time-consuming, and if neglected during times when many new results arrive and need to be processed, some results may even be lost. But even if done carefully, each experiment is subject to so many different parameters that a single label, such as the filename, cannot hold it all. As a consequence, the researcher often has to look at the actual data to determine – or sometimes guess – the original context. Moreover, it is very hard to sort and filter the information to quickly find what is needed for a particular experiment, something that has been a cause of nuisance in earlier work.

Because long-term data management is considered an important subject, SPEACLAP has ample built-in support for it. First of all, when a feature table is created, an array of meta information, e.g. creation date, source corpora and files, audio format, and feature extraction configuration, is stored with it. To browse features, the *feature browser* depicted in Figure 5.12

Horizontal join

	F1	F2	F3			F4	F5	F6			F1	F2	F3	F4	F5	F6
A	2.44	5.45	23.76	+	A	143	62.59	4.14	=	A	2.44	5.45	23.76	143	62.59	4.14
B	2.76	4.77	22.58		B	146	55.48	4.77		B	2.76	4.77	22.58	146	55.48	4.77
C	2.37	4.99	23.81		C	118	54.21	4.06		C	2.37	4.99	23.81	118	54.21	4.06

Vertical join

	F1	F2	F3			F1	F2	F3			F1	F2	F3
A	2.44	5.45	23.76	+	D	2.60	5.12	21.46	=	A	2.44	5.45	23.76
B	2.76	4.77	22.58		E	2.95	4.78	22.44		B	2.76	4.77	22.58
C	2.37	4.99	23.81		F	2.63	5.06	22.38		C	2.37	4.99	23.81
										D	2.60	5.12	21.46
										E	2.95	4.78	22.44
										F	2.63	5.06	22.38

Combined join (with missing values)

	F1	F2	F3			F2	F3	F4			F1	F2	F3	F4
A	2.44	5.45	23.76	+	B	4.77	22.58	146	=	A	2.44	5.45	23.76	NULL
B	2.76	4.77	22.58		C	4.99	23.81	118		B	2.76	4.77	22.58	146
C	2.37	4.99	23.81		D	5.12	21.46	131		C	2.37	4.99	23.81	118
										D	NULL	5.12	21.46	131

Figure 5.11: The three modes of joining feature tables supported by SPEACLAP. If features or record IDs are duplicate in the source tables, they may produce missing (*NULL*) values.

provides an explorer-like user interface made up of a tree view and a sortable list view, where features can be scanned and selected. There is also a search function for looking for features with specific criteria across all storage providers. From the same place, the feature table can be displayed and analyzed using the table analysis tools integrated into the IDE. It can also be exported to CSV, Excel, and several other formats.

To conserve space and simplify management, there is a default feature table for each configuration of source files and feature extraction script. If the features are extracted multiple times, possibly with slightly different parameters, the old features are overwritten unless the user specifies otherwise.

5.3.4 Classifier Design

Based on the features extracted in a prior step, classifiers can be trained and evaluated from within the GUI. SPEACLAP distinguishes between design-time classifiers and runtime classifiers. *Design-time* classifiers are only part of the development framework and are implemented as .NET scripts, while *runtime* classifiers are the ones that are embedded into the final application as C++ code and controlled via their API. Runtime classifiers cannot currently be trained, but are created directly from existing design-time classifiers and use the stored

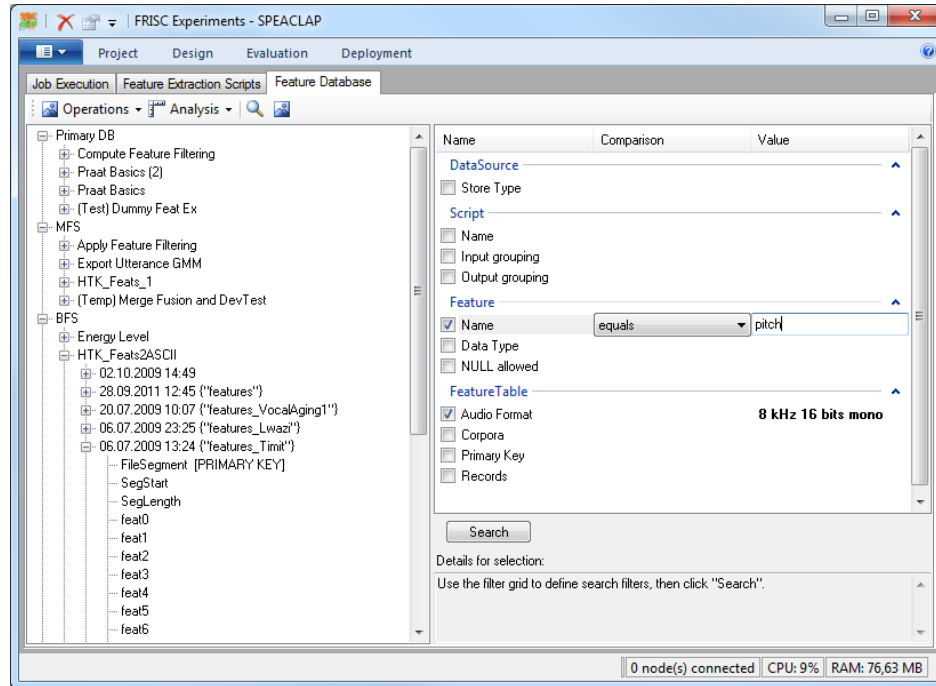


Figure 5.12: The feature browser enables the user to browse all data stores, list individual tables and also filter for particular features.

Script Name	Description
RandomSubsetFilter	Randomly selects a range of files (e.g. 0% - 90%). Can be initialized with a static seed to allow evaluation to work on the remaining part.
AbsoluteNumberFilter	Selects n files, either randomly or not.
IDListFilter	Selects a user-defined set of files as defined by an externally provided list of IDs.
ClassBalancingFilter	Reduces the number of files per class until each class contains equally many files.

Table 5.3: List of file filter scripts already included in SPEACLAP.

model. Design-time classifiers support a more sophisticated interface, including serialization and conversion into runtime classifiers.

For every classification problem, a design-time classifier is created, thus a project usually consists of several classifiers. Before it can be trained, the features have to be selected from all available data sources using the feature browser. It is also possible to apply dynamic post-processing steps such as normalization and custom scripts, and to filter files based on corpus and meta properties. There are also some filters which work purely on the file list and are commonly used to select files for training and for evaluation. Some important file filters are listed in Table 5.3. Then, the class property is picked from the list of corpus meta properties. Finally, the classification algorithm is chosen.

In the current version of SPEACLAP, only the classifiers used as part of FRISC have been implemented, i.e. GMMs with K-means initialization and MAP adaptation (using a custom

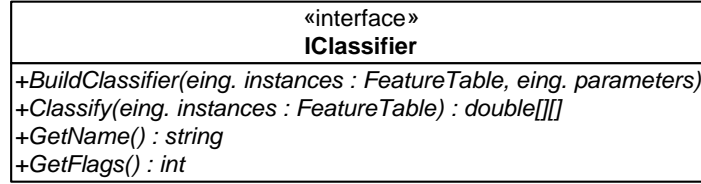


Figure 5.13: Interface which is implemented by all classification algorithms (UML class diagram). *eing.* stands for *input* arguments.

implementation), and SVMs using either *svm-lite* (Joachims, 1999) or *LibSVM* (Chang & Lin, 2011). Both implementations have been optimized with respect to parallel and incremental processing and size limits. For example, the GMM can train multiple classes in parallel with only a single source data loop, which reduces the bottleneck otherwise introduced by the feature table iteration. The MAP adaptation algorithm uses an implementation that closely follows Reynolds et al. (2000), where re-estimation of an already adapted model is straightforward by adjusting what they call the “sufficient statistics” and using the previous Gaussians. This quality is beneficial for the incremental processing of frames in arbitrary intervals. For the SVM, the external application *svm-lite* has been ported to 64 bits to allow for training data and models larger than 2 GB. Even though these are only two examples of classification algorithms, it is straightforward to wrap other existing classifiers for use in the framework, especially when provided as .NET libraries (such as the *AForge* machine learning library²), but also in other languages (such as the *WEKA* classifiers) or as executables (like in the case of *svm-lite*).

Interface

SPEACLAP allows custom classification algorithms to be written in the integrated editor. There are two types of classifiers: binary classifiers, which can only decide on the class membership of an instance for a single class label, and multi-label classifiers, which decide between multiple labels of class. The difference between the two is mostly a semantic one, but nonetheless important, especially for evaluation and visualization. Here, generative classifiers are treated as binary classifiers with a special flag indicating that they do not produce a class decision. This may not do the nature of these models full justice, but is considered an appropriate technical design choice. The interface for a classifier is depicted in Figure 5.13. As can be seen from this UML chart, the classification method returns an array of arrays of floating-point numbers. The inner array stores scores for the individual classes, with binary classifiers only using a single element, and the outer array encompasses the test samples for use with batch classification.

As we have seen in Section 2.5.5, classifiers produce very different kinds of results. In SPEACLAP, we distinguish between them using the following terms:

²<http://www.aforogenet.com/>

Multi-class decision. This is the index of a class which the classifier assigns to an instance. Each multi-label classifier produces this type of result.

Score. The generic term for a numeric classification result where “greater means better”. Scores are neither normalized, nor do they imply class decisions.

Probability. Technically, a probability is a value between 0 and 1 that does not imply a decision. It is produced by some single-class classifiers, such as the naive Bayes classifier.

Likelihood. The likelihood is actually a type of probability and can be used interchangeably with that term in the context of classifier outputs. A special case is the *log likelihood*, which has a different scale. Yet, neither imply a decision. In practice, not all likelihood functions are normalized.

Likelihood ratio. The (log) likelihood ratio is a binary classification result, which has a natural threshold at 1 (0 for LLR). Values ≥ 1 can be interpreted as acceptance, while values < 1 can be interpreted as rejection. By shifting accordingly, the threshold can be moved to 0, which is the default used by SPEACLAP when scores are interpreted.

Training

When training is started, a compound table is created for the source features. It contains the features and a column with the class label. When training a binary classifier, a multi-label wrapper is automatically created, which internally consists of one binary classifier per class, and the training table is modified for each classifier to have Boolean class labels instead of text labels. The algorithm being trained can decide whether to access feature data using the enumerator or the table-based approach. A future version might also choose the version depending on whether there is enough memory available for the whole table. After training is complete, the model is serialized to a binary file.

Feature Normalization

Certain classifiers are sensitive to the input range of a feature. Numeric input features can be automatically normalized by applying a normalization script, which maps the feature values to range of $[0; 1]$. Normalization will be performed on the same data that is used to compute the normalization criteria, unless specified otherwise. The following types of normalization have been implemented and used in the FRISC experiments:

- **Linear mapping normalization** can be used to map the smallest value of a series to -1 , the largest value to $+1$, and the values in between linearly.
- **Mean-variance normalization** is only slightly more complex and probably the most commonly used type of normalization, which first computes the mean μ and variance σ^2 on the data and then normalizes a value x according the formula $x' = (x - \mu)/\sigma$.

- **Rank normalization** requires more memory to store all individual samples. Based on the samples, an inverse quantile is computed. I.e. for a value x , the value $x' = q/100$ for which x is the $q\%$ quantile in the normalization set. If x is not in the set, it has to be interpolated from the closest values using linear or a more complex type of interpolation function.

5.3.5 Classifier Evaluation

The IDE features a wide set of evaluation functions for classifiers. Evaluation in the context of SPEACLAP really refers to the technical evaluation process as defined in Section 2.5.7, although it can be used for component evaluation as well. There is one default evaluation for each classifier, but custom evaluations with deviant parameters can also be created. Both design-time classifiers as well as runtime classifiers can be evaluated. The latter is especially useful because it enables the user to benchmark the classification performance of the final application. Files that are used for evaluation can be filtered in the same way as for training, e.g. with a custom training set. If training was performed using a `RandomSubsetFilter` with $n\%$ of the files, there is an option turned on by default which will then use the remaining $(100 - n)\%$ for evaluation.

Results Data Format

The results of an evaluation have a specific format. Like in other areas of the framework, the propagation of a fixed structure is supposed to help in making tools and scripts widely applicable to all kinds of data. Therefore, all evaluations are based on the same format; however, some parts of the data can be omitted from archival in individual evaluations, e.g. to save space. An evaluation result consists of the following parts:

- **Core Meta Data.** This basic information can be used to reconstruct most vital information about the evaluation, such as the configuration in which it was executed, the evaluation set and input features, the class property and class labels, the classifier that was evaluated (including its parameters such as the number of mixtures in a GMM), and some core statistics for quick access (e.g. accuracy and error rates).
- **Scores.** The scores are the most important outcome of the evaluation. They are stored in a structure referred to as *scoretable*, based on its format. To allow running and saving numerous evaluations, the structure stores merely the most critical data and is designed in an efficient way. It consists only of numeric information and (virtually) no strings. More details are given below.
- **Extended Results.** This data goes beyond simple meta data; it can be any kind of complex data structure computed by a script in conjunction with the evaluation. Examples are score normalization data or the thresholds used for score optimization.

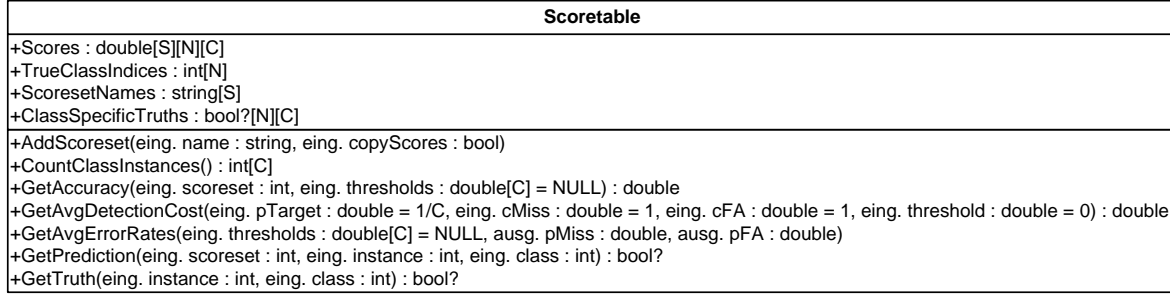


Figure 5.14: UML class diagram of the *Scoretable* data structure, including some of the auxiliary methods. The array boundaries denote the number of instances (N), classes (C), and scoresets (S).

- **File IDs.** To save resources, the original corpus file IDs are normally not saved with the evaluation results, especially since they often do not change. However, sometimes it is desirable to keep them separately. They can for instance be used for manual investigation of outliers, e.g. by listening to samples which have provided an unexpected score for a class.
- **Classifier.** The classifier which was evaluated can optionally be included in the archived results.

The *scoretable* data structure is motivated by the fact that almost all relevant classification performance metrics, both in terms of detection and identification task, can be constructed from a table or matrix of scores where each target has been tested with each sample. Accuracy, confusion matrices, error rates, DET plots, etc. can all be derived from this tabular representation. Figure 5.14 shows the structure of a scoretable. The main array **Scores** contains the scores indexed by instances and targets. In addition, there can be multiple score sets of the same dimension $N \times C$. Each scoreset can be given a name. New scoresets are created when score modifications are applied, which might for instance be additive decision thresholds or normalization. The previous scores are always kept for reference in such a case. In addition to the scores, the truths for each class are stored in the **TrueClassIndices** array. In some classification scenarios, multiple truths can be valid for a sample (e.g. overlapping speakers in speaker diarization). In this case, the truth is specified per class in the array **ClassSpecificTruths** instead. However, this case is beyond what is needed for typical Speaker Classification problems.

In addition to evaluating a classifier trained in the IDE, results such as scoretables can also be imported from an external source (e.g. in CSV format) to use the reporting features of the IDE. This corresponds to one of the usage scenarios of the SPEACLAP framework. It can be helpful for instance to quickly create DET plots of experimental data.

The evaluation results can also be written to a new feature table. This offers the possibility to create n -th order meta-classifiers, which are trained on the results of other classifiers. For

example, to support the gender-dependent aging concept known from AGENDER, one could create two classifiers for age, one for gender, and then train a third “final” age classifier on the evaluation results of the other three classifiers, using the true age as class label. All of this works completely transparently because of the unified feature table structure. Of course, evaluation result feature tables can also be analyzed and exported like other feature tables.

Reports

Just as with feature data, evaluation results are archived. Based on the standardized results, multiple reports in different formats (e.g. text and HTML) can be created. The default report contains some basic classifier information, a detailed confusion matrix, and measures like precision, recall, and error rate. A further report can be used for drawing ROC curves for all classes. The graph-based reports use *Gnuplot*³ internally and can be configured to output EPS, PNG, or another supported graphics format. Again, the collection of reports can be expanded with custom scripts. There is also a special evaluation differences report, which compares the classifier parameters from multiple evaluations.

The following scripts have been developed for SPEACLAP and were also used to create the graphics shown in Section 4.3 and other places within this document:

- **BasicEvalReportGenerator:** Summarizes the general (meta) information about an evaluation, including information about the features and the classifier that was evaluated.
- **TargetCentricEvalReportGenerator:** Reports the error rates for each target class, including nontarget-specific false alarm rates. A custom DCF can be specified for computation of C_{Det} . An example of this report can be seen in Figure 2.21 on page 67.
- **DETCurvesEvalReportGenerator:** Produces detection error tradeoff curves. One chart per class is generated, one chart containing all classes, and one chart containing the merged curve of all trials. All aspects of the DCF can be customized in the script parameters. The charts use the same formatting used by *NIST* in their original proposal. In fact, the computation script is based on the *NIST* Language ID evaluation script. For an example, see Figure 2.22 on page 68 and Figure 2.23 on page 69.
- **ConfusionMatrixEvalReportGenerator:** Generates the type of confusion matrix that is popular with identification tasks. In addition to the identification matrix (see Figure 2.24 on page 71), matrices for all binary sub-problems are created, if the classifier is a wrapper around a set of C binary classifiers. The matrix is generated in HTML format and can easily be copied to spreadsheet applications.
- **ConfusionMatrixBubbleChart:** This chart represents the same core information as the confusion matrix, but is graphically enhanced for better viewing. An example is the chart in Figure 4.11 on page 133.

³<http://www.gnuplot.info>

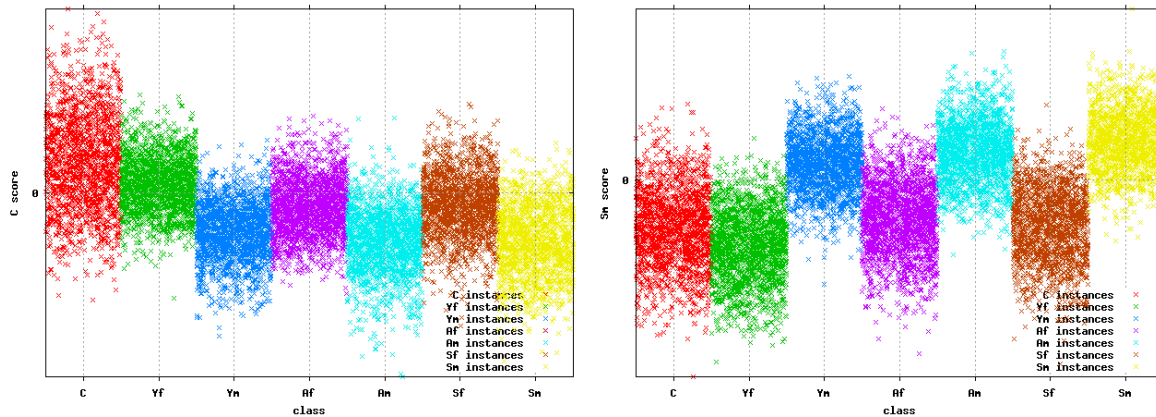


Figure 5.15: Score scatterplot. Each chart shows the scores produced by seven target classifiers from the FRISC main experiment series on test samples of a single class. This chart was generated for scores optimized towards minimizing the DCF. *Left image:* Scores from *Children* samples. *Right image:* scores from *Senior male* instances. A clearly visible effect in both cases are the considerably higher scores for targets trained on the same gender.

- **ROCCurvesEvalReportGenerator:** This script generates receiver operating characteristics curves such as the one shown in Figure 2.25 on page 72. It is based on the optimized algorithm by Fawcett (2006).
- **ScoreScatterplot:** Visualizes the raw scores by class. When looking at the performance of individual classes, these charts may provide more insight than the condensed view of a confusion matrix or DET plot. They might reveal information about variance, patterns, and outliers in the score space. Two sample scatterplots are depicted in Figure 5.15. Note that the x value has no meaning except to indicate the target; within a target, the values are randomly distributed to make the plot more readable.
- **ScoresetEvalReportGenerator:** Returns some general numerical and statistical information about the scores in the result set. This may be helpful when comparing different scoresets, e.g. to study the effect of normalization or a different score modification script.
- **ComparativeImprovementReport:** Plots a chart of the improvement in multiple evaluations over time. Accuracy or an error metric can be used for comparison.

Score Modification

Technically speaking, calibration or score modification means that a special data structure is computed from the output scores of a classifier according to some logic. This data structure can be considered as a transformation function applied to scores. It is stored separately

from the classifier, and can be applied to all scores produced by that classifier later on by requesting SPEACLAP in the system evaluation step to apply the score modification data produced earlier. The major reasons for performing score modification are *optimization*, *normalization*, and *aggregation*. The former two are also often summarized as *calibration*.

The modification of classifier scores is not part of the training, but performed during the evaluation of the classifier, since the classifier has to be applied to a full data set to complete the process. In an experiment, it is common to set up the evaluation of a classifier with a special development test set right after the training to produce the score modification data structures.

There are arbitrary ways to modify scores. The score modification structure can in theory be just another classifier model that works on a scoretable. In practice however, these models are rather simple and most commonly consist of a global or class-specific decision threshold, which is a constant added to the classifier output score. The score optimization is implemented by a score modification script, which knows how to compute the modification structure based on existing scores, and how to apply it to new scores. Apart from that, the script is characterized by the cost function by which it computes the structure (see Section 2.5.1). SPEACLAP implements two optimization scripts: One for the detection task, which can optimize by P_{Miss} , P_{FA} , P_{Err} , or C_{Det} , and one for the identification task, which optimizes by accuracy.

Many classifiers do not produce normalized scores (see Section 5.3.4). During evaluation, a so-called *score normalization* can be performed. This is useful when classifier scores need to be post-processed in an interoperable manner, i.e. using the normalized range $[-1; 1]$. With score normalization, SPEACLAP will apply a normalization method to the scores encountered in the evaluation results and stores the resulting normalization data (e.g. mean and variance for each feature). This data structure will then automatically be used to normalize scores. The same scripts that can be employed for feature normalization (see Section 5.3.4) are available for score normalization. This process works technically almost identical to optimization. A special type of score normalization that can be applied is *test normalization* or short *T-Norm*, which is frequently used in speaker verification to compensate the inter-session variability (Barras & Gauvain, 2003).

Score aggregation is used to simulate the addition of temporal information into otherwise separate scores. While the number of output scores stays the same as the number of input scores (at least in this implementation), the output scores are actually aggregated over several input scores using a sliding window with this method. For example, the output instance score with the index 3 could be computed by averaging the per-class scores of input samples x_1, \dots, x_5 . Alternatively, a multi-class majority voting could take place. Aggregation is very useful when the instances are frames of an audio file. Since in practice, decisions are rarely generated on a per-frame level, it is purposeful to merge several frames into one sample. For this purpose, SPEACLAP offers the `OverlappedScoreAggregation` script.

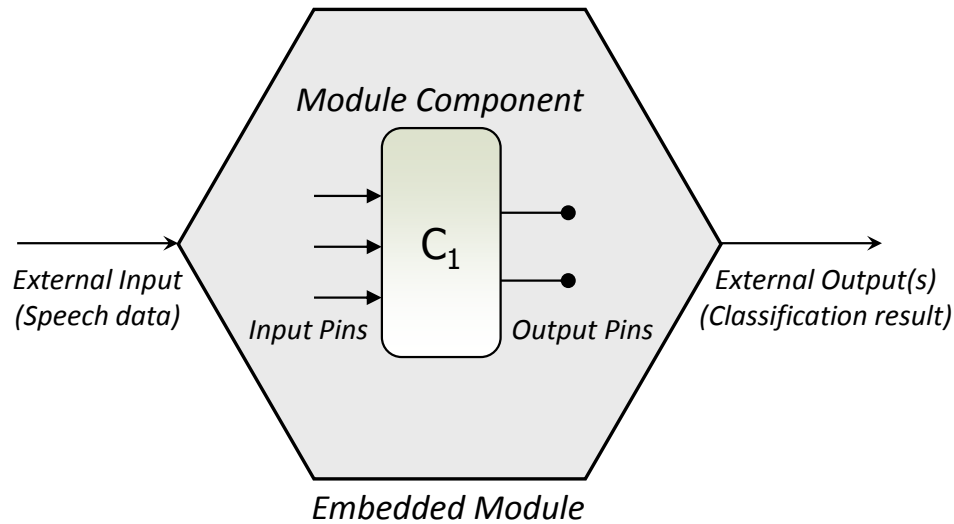


Figure 5.16: Schematic presentation of the basic layout of an embedded module.

5.4 Embedded Modules

The idea of embedded classification modules was introduced in [Feld \(2008\)](#). It is born from the need for a fast, compact, portable, and customizable solution for Speaker Classification. An embedded module is a highly optimized black-box classification pipeline that can be automatically compiled from a classification module template for a specific platform and configuration, and which already contains the interface that allows it to be integrated into any kind of application.

In this context, the term “embedded” has multiple meanings: First, the modules are pieces of software which become *embedded* into other applications (i.e. embedded in the sense of *integrated*). Next, all relevant models and logic are *embedded* into a single file, which represents the module. And finally, they are also optimized for use in *embedded* scenarios such as mobile devices and automotive scenarios. SPEACLAP was designed with an easy transition from development versions of classifiers to mature production-quality modules in mind.

Classification modules support several interfaces, which are mostly optional. The “core” interface, which is the most compact version of the code, can be statically compiled into C or C++ programs and is fully integrated into the host application. The DLL interface is made up of a single .dll file that can be called from C, C++, or any other language supporting dynamic library calls. In addition, wrappers for .NET and Java are available.

5.4.1 Basic Structure

The structure of an embedded module is schematized in [Figure 5.16](#). A module consists of components, which are created and plugged together in the IDE. Components can be classifiers, feature extractors, pre- and post-processing components, and so on. In the essence,

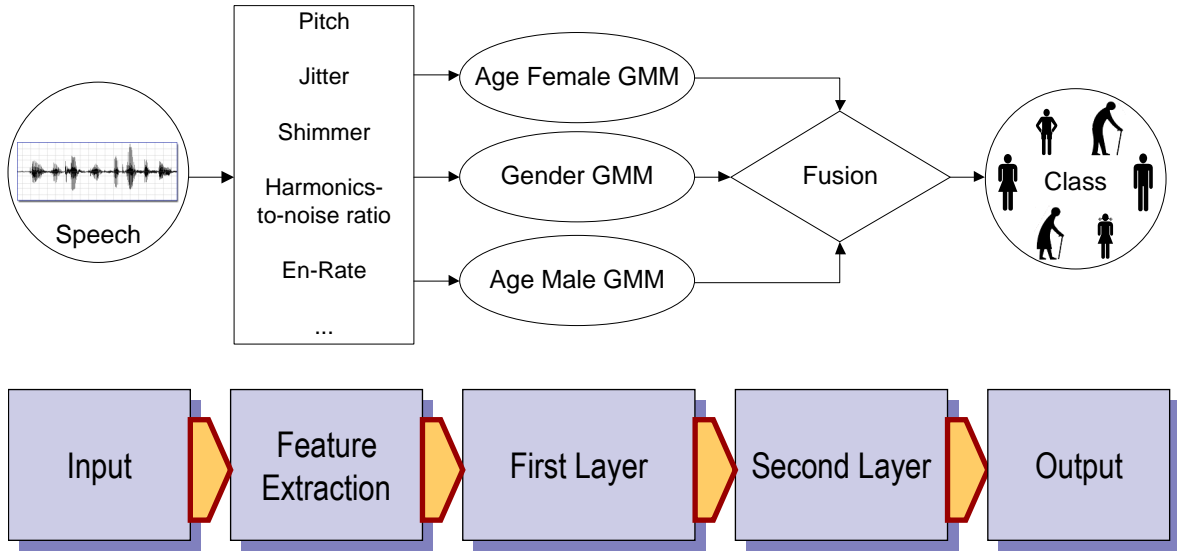


Figure 5.17: A sample pattern classification system implementing recognition of speaker age and gender. There is one classifier for gender and two gender-specific classifiers for age. Their results are combined on the second layer.

each component has one or more inputs and outputs, and applies some transformation to the data in between. The dependencies that are introduced by connecting the modules will determine how the final classification pipeline looks. Once the module design is in place, a fully automated build process generates and compiles the code and the desired interfaces. The scripts for which module components are created must support code generation. For instance, a GMM classifier must be able to provide a C++ implementation in addition to the .NET design-time implementation to be eligible for integration into an embedded module – there is no automatic process for generating C++ code from .NET assemblies. Module configurations can be saved and loaded again later for quick recompilation if parameters of the scenario change, e.g. the classes, or if more source data becomes available.

To get a better understanding of the different elements that make up a classification module, we will consider a simple, very traditional classification pipeline. The example is illustrated in Figure 5.17. We assume that the speech data is already present in raw LPCM format. The first step is the feature extraction, which computes a number of features using tools such as *Praat* and custom algorithms. Then each of the available classifiers is run with an input vector composed of these features. In the final step, the results from the individual classifiers are merged and post-processed to form a likelihood for the speaker classes the user is interested in.

As can be seen from this description, the classification process models a mostly linear data flow with different steps of data transformation. Consequently, each of the steps is modeled as an individual component in the module with inputs and outputs, also called *pins*. The majority of components can be categorized into *pre-processors*, *feature extractors*, *classifiers*,

and *post-processors* to reflect the structure of a pattern classification system as defined by Duda et al. (2000), but technically, all components share the same generic i/o interface. From the application view, the classification module is a “black box” that takes speech data as the input and returns class decisions as the output.

The data flow within the module is created by adding connections between pins. Connections always go from an output pin to an input pin. An input pin can have only one associated connection, but an output pin can be connected to multiple input pins. There are some special pins to facilitate the external interface of the module, which is the speech data as the external input pin, and likelihood output pins, which convey the results returned to the application. The connections introduce dependencies between components and thus define the order in which they are executed.

A pin has an index, a direction (in/out), a data type, and some optional flags. The data type can be one of the basic types *int*, *long*, *float*, *double*, *byte*, and *boolean*, or an array of any of the former. It is also possible to pass custom structures as a generic pointer without further type checking. A component is *ready*, i.e. can be executed, when there is data available for each of its input pins. Normally, data is queued if multiple components supply data to the same pin, and each of the inputs is handled in a separate run of the component. In some cases, however, it may be desirable to process all data in the same run. For this purpose, there is a special type of pin called a *multi-field*, which works similar to a list in such manner that new data can be appended and the component can traverse the whole list each time it executes.

In a number of cases, multiple instances of the same component representing different classes are part of the classification pipeline. The most common example for this are binary classifiers, where there is one classification component for each class. A special logic is applied to components marked as multi-class: Each class is created as a separate component creating different outputs, but they are treated as a unit and share the same input pins. As far as the outputs are concerned, the individual class results of type *T* are either combined to form an array output pin of type *T*, or they are stored in a multi-field (list) type where they can be processed sequentially or – if the input pin to which they are forwarded is also a multi-field – in parallel.

The classification module runs all components which are ready in the order defined by the connections until no more output is produced in one execution cycle. Components can be set to run either as soon as input data is available (“greedy”), or as late as possible. The latter is sometimes useful when a component incorporates the results of an arbitrary number of other components. To make this work, at least one input pin must be a multi-field.

The core module is compiled to a C++ static library, which can be natively linked to C or C++ programs. To use the module from different programming languages, one or more external interfaces can be built. One such interface is the DLL interface, which encapsulates the module in a .dll (*dynamic link library*) file. DLL calls are supported by most programming languages. Yet, to accomplish the goal of a very easy integration, there are also specialized interfaces available for *Java* and for *.NET*. The *Java* interface contains a class file that

internally uses *JNI* (*Java Native Interfaces*) to call the functions in the module DLL.

In order to produce an embedded module, a module template needs to be created and compiled. The following two sections will elaborate on both processes.

5.4.2 Module Design

The design phase of a classification module consists of two steps. First, the components that are needed in the classification process must be created. Second, the module *template* is assembled from these components. Both tasks are supported from within the SPEACLAP application.

Feature extractors and some processing components can be created by selecting a pre-defined script. A script contains the logic to produce the component's code and can be customized through parameters. For example, the *Praat Feature Extractor* script produces code that computes a vector of features from an audio sample. A classifier is also based on a script, but can be pre-trained and evaluated in the development environment. There are several generic scripts available that perform data type conversions, which may come in handy when designing complex modules.

External interfaces, though not part of the actual module, can be selected and configured using the same user interface. They are built together with the module, usually in both a *Debug* and *Release* configuration. There are dependencies between some of the interfaces, e.g. the *.NET* interface needs the DLL file built by the DLL interface.

Figure 5.18 displays the embedded module design window. Components can be added, removed and configured. The configuration of a component as shown in Figure 5.19 determines the pins it provides. Some components may for example provide a switch that determines whether a vector result is output as one value per output pin or as one output pin with an array-type value. This allows for dynamic array sizes and flexible combination of modules. There are also global module settings such as platform, caching, tracing and benchmarking options. In most cases, they have an immediate impact on the build process.

In the common case that some variables of a Speaker Classification scenario change (e.g. the age class boundaries), a previous module design can be loaded and the corresponding adjustments can be made. The module can then simply be recompiled to replace the previous one.

Let us again consider the example pattern classification system depicted in Figure 5.17, which consists of feature extraction, classification, and post-processing components. This theoretical concept can be converted into an actual Speaker Classification module in a few steps: After creating a new module in SPEACLAP, the *Praat Feature Extractor* component is added. It has output pins for a large number of speech features, but can optimize the feature computation by checking what features are actually used, which is done at compile time. Next, three GMM classifiers are created in the IDE's *Classifiers* tab, trained on speech corpora, and added to the module as three separate classification components. Their inputs, which are different for every classifier, are linked to the corresponding outputs of the *Praat* component

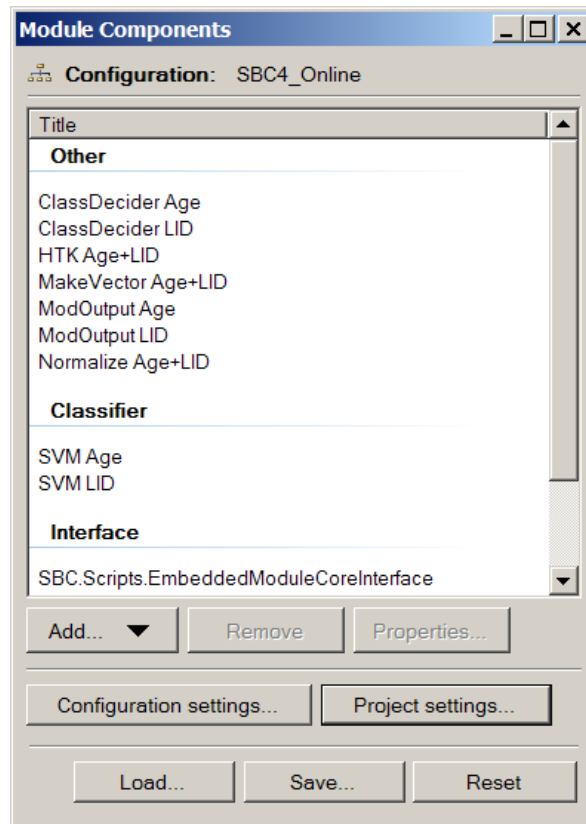


Figure 5.18: The embedded module template design window allows adding and configuring all components of the classification pipeline.

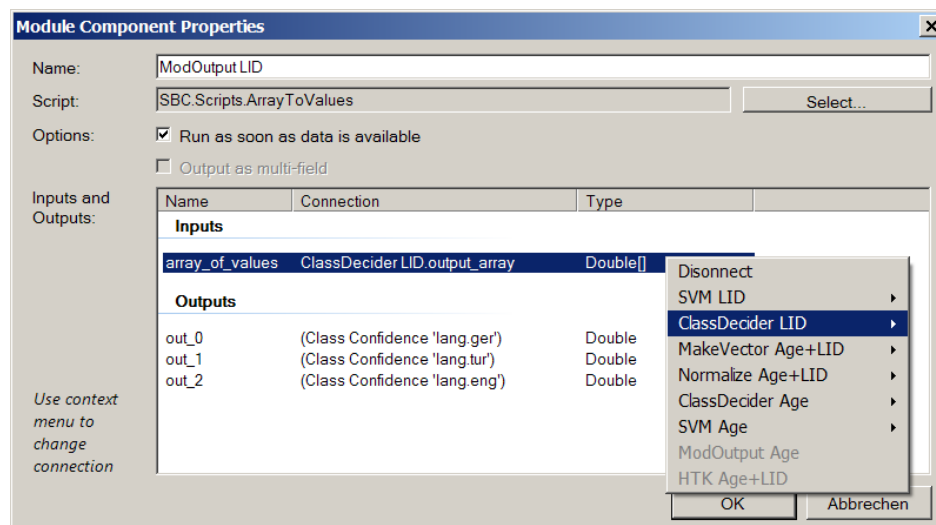


Figure 5.19: Embedded module component property dialog. The screenshot illustrates how pins can be connected.

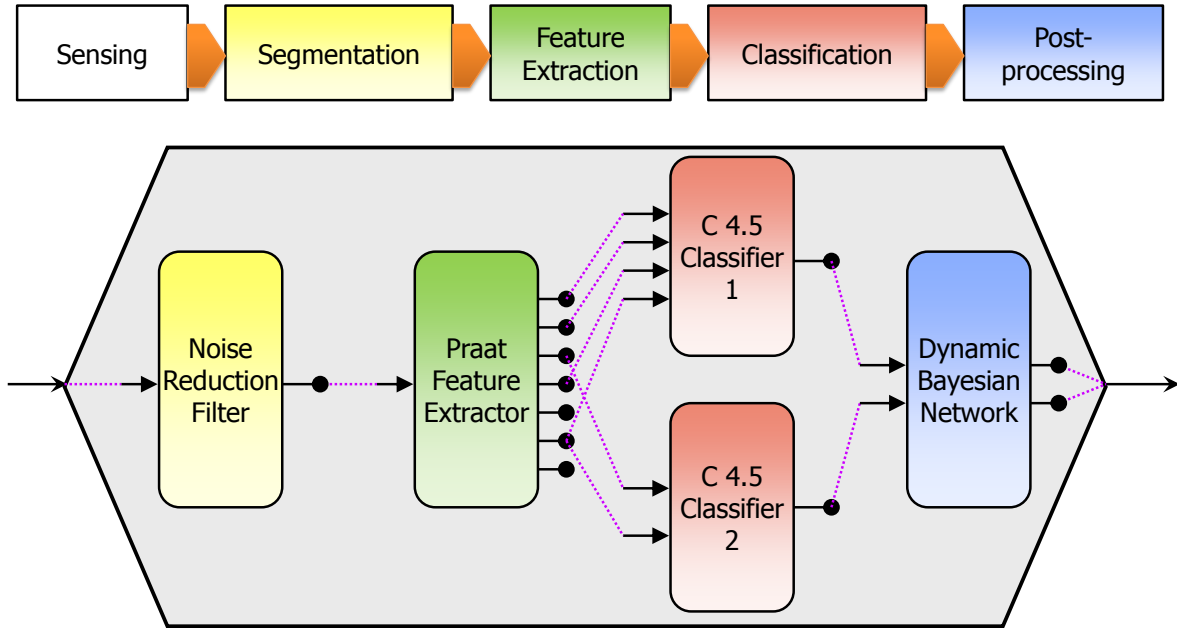


Figure 5.20: Example of an embedded module implementing a basic classification pipeline. It mostly matches the example from Figure 5.17, except that a pre-processing component was added and one classifier was left out for better readability.

using the module designer UI. The result of the *gender* classifier is directly forwarded to an output pin *gender*. For *age*, we add another component, which receives the results of both *age* classifiers as input, as well as the result of the *gender* classifier. Its single output is the result of one of the *age* classifiers, depending on what the *gender* classifier says. This structure reflects the fact that normally, the performance of a *gender* classifier is considerably higher than that of the *age* classifiers, and combined with gender-dependent aging (see Section 2.1.3), a more reliable score for *age* will be the result. This output is set to be the *age* result of the module. The resulting module might look similar to the schematic shown in Figure 5.20.

5.4.3 Compiling Modules

The process of compiling an embedded module template into an actual module is called *build process* and is a vital part of the concept. It is fully automated in a way that a single click in the GUI designer initiates an operation that produces the module library and external interfaces. This is very useful for example when the training data for a module changes, as all existing applications can be updated without reconfiguration. Additionally, embedded modules can be versioned and are automatically archived.

Each interface is built using a *module interface script*. The script that generates the static library in its most native form is the *Core Interface* script. Other scripts include the *DLL Interface* script and the *.NET Assembly* script. There is also a *Test Application* script, which generates an executable that is statically linked to the core interface and that can be used to

Element	Description
Project file	File that aggregates all source code files, and that can be built using a command line tool.
Dynamic source code files	Source code files that should be included in the project (if they are not already defined statically in the project file).
Include directories	Directories that should be browsed by the C/C++ compiler for include files.
Library directories	Directories that should be browsed by the C/C++ linker for libraries.
Libraries	Libraries that should be linked to the project.
Definitions	Preprocessor constants that are included on compilation.
Injections	Pre-compilation text injections into source files. This can be used to compile configuration variables (e.g. module version) into the output. An encoding can be specified for each injection.
Output files	Files that will represent the final project output (i.e. no temporary files). These files are moved to the output folder.

Table 5.4: Information generated by a module interface script.

classify files in .wav or raw LPCM format on the command line.

A module interface script does not normally launch the actual build process. Instead, it generates a fixed set of information (see Table 5.4) that describes the source code and project file that can be built using one of the standard build tools (e.g. the Microsoft Visual C++ Compiler for C++ code or the C# Compiler for C# code). Hence, the build process first invokes all interface scripts to collect the source code and meta information, post-processes it to complete the project file, and then invokes the build tool for each project. Most of the tools have been encapsulated by a wrapper called *MSBuild*⁴, which works similar to a *Makefile* (Feldman, 1979) by describing build targets and default commands to build these targets. The source code that is generated by the script must be output to a predefined temporary directory and either be referenced statically from within the project file or dynamically from the *Dynamic source code files* section.

The *Core Interface* script is considerably more complex than the other scripts. One of its tasks is to determine the order in which components need to be executed to satisfy their dependencies. This also includes deciding which components can be executed in parallel without causing side effects. To accomplish this, the whole execution is split into *processing blocks*, and modules that can run concurrently are assigned to the same block. After the order has been defined, each component that was added to the module template is asked through a script to generate its code, include files, binary resources, and type references. The code is enclosed in a class frame and written to an implementation file while the call to the class is added to the corresponding processing block. The most challenging part of processing block code generation is to facilitate the data marshalling between components. The reason for this complexity is that the input and output interfaces may be very different in type and structure. For example, some pins may implement value types, some pointers

⁴[http://msdn.microsoft.com/en-us/library/wea2sca5\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/wea2sca5(v=VS.90).aspx)

to values, and some objects, which also have to be destroyed later in a garbage collection cycle. Also, conversions between simple fields, arrays, and multi-fields can occur, and they work differently for single components and multi-class components. Although the generated code is very difficult to read by humans, it is quite fast and most likely saves the application developer much effort that he would otherwise have to invest in hard-wiring his pattern recognition components manually.

Most scripts generate the output in both a *Debug* and a *Release* version. The *Debug* version does not only include the debugger information created by the compiler, but it also integrates a *tracing* engine that will write all steps in the classification process into a log file when run. This has been particularly helpful during the development of classifiers, as intermediate values can be checked for plausibility or be manually recalculated if necessary at all stages. There are some default trace messages, like the names of components being executed and their pin values, but component scripts can also include custom trace output. For the DLL and .NET interfaces, the *Debug* and *Release* version can be switched without having to recompile the application, which is a valuable aid in tracking bugs after the application has moved to a production environment. Versions are not limited to *Debug* and *Release*; in fact a module interface script can produce any number of independent versions to allow for different scenarios.

5.4.4 Module Implementation

From an implementation perspective, the embedded module core interface, which represents the actual Speaker Classification module, is made up of a couple of static C++ files that are either included in a module or not depending on the configuration, and a typically larger number of component implementation files. Static files include the library header, the module component base class, the processing pipeline, tracing functions, benchmarking functions, and some custom implementations for basic data types like arrays, lists, and matrices.

The module execution revolves around the processing pipeline class. Each object of this class can handle one Speaker Classification task at a time. The two main public functions are classification request and result retrieval. Each component is implemented by a class derived from the abstract **ModComponent** base class. One instance of each component is cached statically (i.e. application-wide) in the processing pipeline class. This class uses reference counting to determine when the static instances should be created and destroyed. An application that uses only a single pipeline at a time may create a “dummy” pipeline during its lifetime to prevent the reference counter from loading and unloading the components repeatedly.

At the beginning of each processing request, local data structures like arrays and lists are initialized. They are used to capture the outputs of components, including a flag indicating whether output was updated, and also queue the input elements for some components. The main part of the function is organized in a loop, which executes all processing blocks (see Section 5.4.3) in sequence and runs as long as any output is generated by some component.

Within a block, modules can be run in parallel. Before a component is executed, it is first checked whether the input queue contains any items. If this is the case, the input pins are set accordingly and the component is run. On the first run of a component, a custom initialization procedure is triggered, which might for example load a speaker class model. When the method returns, the output pins of the module are immediately copied to the input queues of components using them. Objects that need to be destroyed later are registered with the garbage collector. Clean-up tasks like garbage collection are performed when the processing request is completed.

There are three methods for including stored classifier models and other resource data in embedded modules: The first method is to compile the data to source code. For example, decision trees and Bayesian networks can very well be converted into code. This is the recommended choice if data is not too large and such a representation is possible, as it is fast both for loading as well as for execution, and it provides the tightest integration. Another way is to store formatted data in code or embedded resource files. This also provides a good integration, because the data is still part of the module, but loading and execution are often slower because they require additional parsing and interpretation efforts respectively. The third alternative is to store the model in an external file. However, this partially forfeits the advantage of having a truly embedded module. For small models, there can also be a decrease in runtime performance as a result of file i/o, but if the model is really large, including it in the main module may not be an option due to system restrictions or exorbitant load times. Figure 5.21 illustrates the difference between these three methods using the example of a decision tree classifier.

Sometimes, module components require calls to external processes. While this should generally be avoided for performance and platform compatibility reasons, it may be the only option in case of a required application for which no source code is available. Here, the engine implements helper functions that can be used to include the application file in the module as described in the previous section, and spawn it at runtime on request to a temporary file. Temporary files are automatically cleaned up when the module is no longer used, i.e. when the last processing pipeline is destroyed, hence the behavior is completely transparent to the user and the embedded module deployment integrity remains intact.

5.4.5 Summary

This section has served as an introduction to how embedded modules in the SPEACLAP framework function and how they can be used. They offer a modular and flexible design for integrating it into different applications on various platforms, are optimized for speed and scalability, and can be compiled in a fully automated build process.

Moving forward, there are still a number of ideas that could not be realized for the existing version yet due to time constraints, and they provide some interesting starting points for further research. One such plan is to provide even more customization options for building embedded modules and include more advanced components as pre-defined scripts. Examples

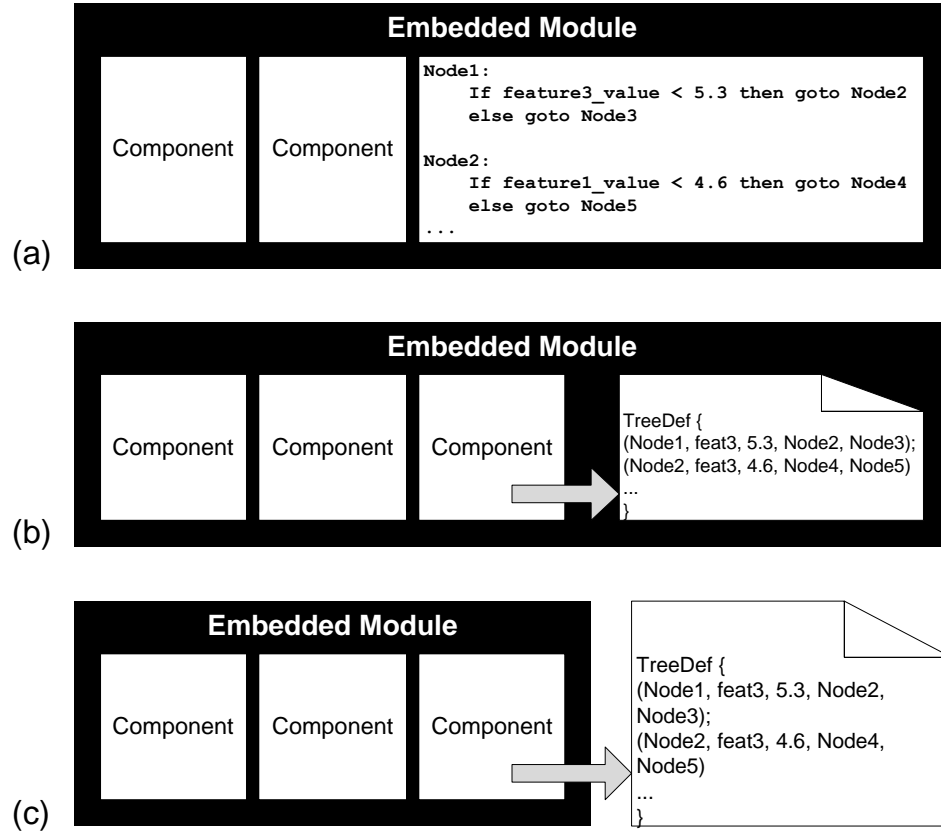


Figure 5.21: Different ways of embedding a decision tree model. (a) Tree classifier compiled to code, (b) tree data embedded as a module resource, (c) external tree data file.

for such components are a multi-purpose cache manager, networking features for server-based classification, user profile management and persistence functions, voice recording, audio conversion and preprocessing, and others. We envision the module designer being able to select these system-layer components from a feature catalog similar to the way it can be done for embedded platforms (e.g. *Platform Builder*⁵).

5.5 Example: FRISC Experiment Set-up

This section serves two purposes: First, it entails further details on the FRISC approach as it was implemented, such as the exact way of data set balancing. Second, it demonstrates the use of the SPEACLAP framework on the example of an actual classification system.

The FRISC experiments have been implemented in their own project. They consist of several types of objects that were configured in the IDE, plus an experiment script that uses the platform's scripting API described earlier. It would have been possible to run all components

⁵Microsoft Windows CE 5.0 Platform Builder, <http://msdn2.microsoft.com/en-us/library/aa448756.aspx>

Listing 2 Example *HTK* configuration similar to the one used in the described experiments. The final configuration depends on the selected parameters.

```

SOURCEKIND = WAVEFORM
SOURCEFORMAT = WAV
SOURCERATE = 1250
TARGETKIND = MFCC_D
TARGETRATE = 100000
WINDOWSIZE = 300000
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 20
CEPLIFTER = 22
NUMCEPS = 12

```

manually through the UI by adding jobs and configuring the parameters. However, for a series of experiments, it is much more comfortable and robust to automate this procedure in a script. Since the experiment script orchestrates the experiment flow and starts all other components, the following description will be oriented at the control flow in the script. This control flow is illustrated in Figure 5.22 with its dependencies.

As part of the preparation, a corpus was set up containing the *SpeechDat-II* training files. During import, files were converted from RAW into WAVE audio format. Labels were imported from a textual annotations file, and ages were converted from integer notation into one of the seven target classes by using a `DynamicMetaDataGenerator` with the FRISC class mapping. Furthermore, several data stores to hold the feature data were added. These stores are explained later on.

5.5.1 Feature Extraction

The first step is the feature extraction using *HTK*. For this purpose, a feature extraction script (*HTK_Feats_1*) has been set up. It runs the external tool *hcopy* on each source file. Arguments are the input WAVE file containing the utterance, an output file, and a configuration file. The configuration file depends on the parameters selected for the step width, window size, number of coefficients, and deltas. An example of such a configuration file is shown in Listing 2. Since the tool outputs binary data, the `BinaryOutputFileParser` is used, and the results are stored to the `MultiFileStore`, which works best with unstructured binary data.

As a successive step, the binary data was converted into a textual format for further processing using the *hlist* tool, which is also part of *HTK*. This conversion was configured by creating another feature extraction script *HTK_Feats2ASCII* that runs the tool, but uses a reference to the file where the binary data is stored as argument to the tool. To make

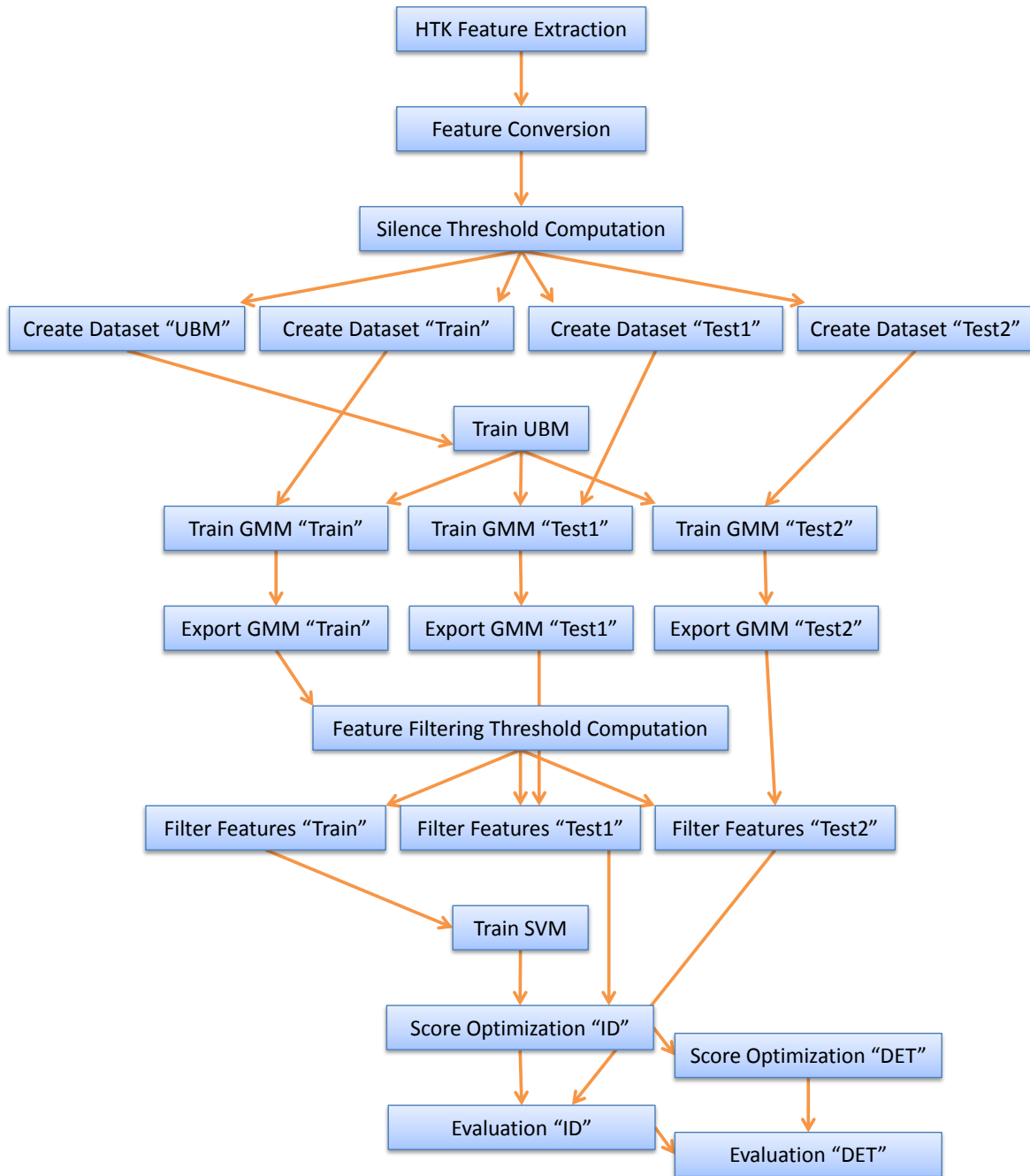


Figure 5.22: SPEACLAP jobs that make up an experiment in the FRISC main series. A component can only start when all jobs connected to incoming arrows, which represent the dependencies, have completed. The order of execution is fully specified through these arrows. Given these constraints, several jobs can run in parallel.

this work, the input to the tool is set to the `MultiFileStore` feature table where the output from the `HTK_Feats_1` script is stored, and a *variable* command argument with the text `features(BinData)` has to be added. Also, the checkbox *Write argument to temporary file* has to be selected in the argument configuration (see Figure 5.23). This tells the execution engine that for each row in the source feature table, the *BinData* column, which contains the binary MFCCs, should be written to file, and the file name should be added to the command line of *hlist*. This script writes its output to the console, therefore, the *CatchConsoleOutput* option in the `ExecutableFeatEx` script that runs the command has to be set to *True*. This causes the output to be used as base for the new features. The actual features are generated by setting the output parser to the special `HTKListSampleParser` script, which parses the MFCCs from the text output of *hlist*. This script has a further parameter *CaptureFrames* that causes multiple samples returned from *HTK* to be stored as multiple frames (instead of multiple records), which is our intention. This time, we use a `BinaryFileStore` data source to store the features. The advantage of this data source is that it can store and query records with a secondary key. We will use this to store the data by using the frame ID (which we more generally call file segment) as primary key, and the corpus file ID as a secondary key. Therefore, the *Output grouping* for the `HTK_Feats2ASCII` feature script has to be set to *FileSegment*, which causes the *FileSegment* column generated by the `HTKListSampleParser` script to be used as the new and unique primary key in the feature table (the file ID is not unique for frame-based features). The *FileSegment* is actually a concatenation of corpus ID, file ID, and frame index. After running these initial two scripts, the output is a feature table containing one row per frame, and columns for the file ID, frame ID, and the MFCC coefficients.

5.5.2 Silence Filtering

The next step, if enabled, is setting the silence threshold. This is performed with yet another feature extraction script we name `Energy Level`. The input is set to the frame table that was created in the previous step. This time, the actual extraction is performed through the integrated .NET script `AudioFeatEx`. It does not use the numeric features stored in the source table, but rather looks up the frame it refers to using the primary key, and performs audio feature computation using any script with the interface `IScriptAudioFeatureExtraction` on it. This way, we run the script `EnergyLevelAudioFeatureExtractor` for each frame in the corpus, which computes the energy according to Section 4.1.5. The result is stored in a `BinaryFileStore` table. To determine the threshold for silence filtering, an event handler is registered from within the experiment script to be called when the feature extraction script `Energy Level` finishes. Within this event handler, the new table is processed and the threshold is determined so that $n\%$ of frames are below. The absolute threshold is logged and stored in a temporary variable.

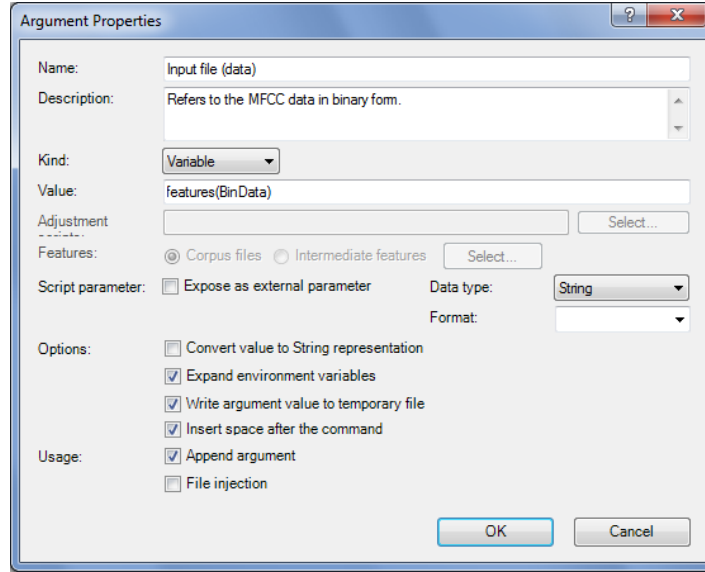


Figure 5.23: Arguments supplied to external scripts have several properties that can be configured. In this case, note that the argument kind was set to *Variable* and the checkbox *Write argument to temporary file* is set.

5.5.3 Data Set Generation

The following step is the data set creation. The final frames for each data set are copied to their own tables, respectively. It would also be possible to perform the filtering on-line during enumeration; however, doing it only once will speed up the process when multiple iterations with the same data set definitions are needed later on. Since data set creation technically involves feature generation, it is realized with yet another feature extraction script, here called **HTK Feature Subsets**. The extractor script we use here is simply called **IdentityFeatEx**, and as the name suggests, it does not modify the input features, but merely copies them. The interesting part and reason why we use this feature extraction script is the filtering that we set up in conjunction with the input features from our MFCC frames table. We set up two types of filters: An *item list filter* based on the reference files, and a *live item filter* based on the frame ID (see Table 5.1 on page 167 for the difference between the two).

The item list filter consists of two cascaded scripts for which the order is important. The first is called **ClassFileBalancingFilter**, and its purpose is to ensure that approximately the same length of data is used for each speaker. It essentially randomizes and balances file IDs by speaker IDs. The parameters for this script are configured as shown in Table 5.5. The second is a **ClassGroupBalancingFilter** script, which works similar, but groups by a third property: It balances file IDs grouped according to the speaker ID by their target class (so it actually balances the speakers). The parameters are also indicated in Table 5.5 (note the additional *GroupProperty* parameter). Distinction into different data sets is performed as part of this second item list filter. Here, a so-called *subset filter* is applied, which will cause

only a certain partition of the corpus to be considered for the data of each class; the rest will be discarded by the balancing script. In this case, a `RandomSubsetFilter` script is used as the subset filter, but a different filtering script could be used as well. The `RandomSubsetFilter` is configured to produce the first 40% for the *train* and *ubm* sets, the following 30% for *test1* and the last 30% for the *test2* set (see Section 4.1.1). By enabling logging, the final list of generated data sets can be cross-checked to ensure that no errors have been made. The aforementioned balancing and subset selection filters all have a *RandomSeed* parameter, which is used to initialize the random number generator that performs the shuffling of data. By specifying the same seed constant between experiments, we can be certain to produce the same output sets for the same input data.

After the item list filter, which is only applied to the reference files (i.e. the original corpus files denoted by the file ID in the feature table), a live item filter is applied to all frames during the enumeration. This filter is used to apply the silence threshold computed earlier. The live item filter used here is a `FileSegmentBasicAudioStatLiveFilter` script which does exactly that: It computes the audio features using the same script as before (`EnergyLevelAudioFeatureExtractor`) and compares the scalar output value with a pre-defined threshold. Only frames with a value above the threshold will pass the filter. If the silence filtering is disabled in the experiment configuration, this step is skipped. The output tables are also stored in the `BinaryFileStore` and receive a unique name for each set, i.e. *ubm*, *train*, etc. Note that the data generation steps can be executed in parallel.

5.5.4 GMM-UBM Training and Export

Once the data sets are in place, training of the UBM can commence. Although it is not evaluated, the UBM is designed as a classifier object in SPEACLAP. A class property is not set, which causes only a single class to be trained on all instances. The classification algorithm GMM is configured with the selected number of Gaussians, EM iterations, and initialization function. Further, the input features are set to a subset of features from the *ubm* feature table created in the previous step. A regular expression filter is applied by configuring a `RegexFilter` for the feature selection. This is needed to filter out the MFCC coefficients which should not be included, e.g. when only MFCCs 12-19 are selected in the experiment configuration. When the training job is complete, the classifier will automatically be stored.

After the UBM training, the utterance GMMs are trained in parallel. One training job is generated by the experiment script for each of the three data sets *train*, *test1*, and *test2*. In the IDE, each of these classifiers is represented by a separate classifier object, which ensures that they will be written to different files, although their settings are identical except for the input table. The input features correspond to those of the UBM, including the `RegexFilter`, only that a different table name is specified depending on the data set. The GMM classifier is configured to MAP-adapt from the UBM trained earlier, and the relevance factor is set to the preferred value. It is also specified that only the means should be adapted. This time, the class property for the classifier is set to *FileID*, so that one class (one GMM) is trained

Parameter Name	Description	Value (1)	Value (2)
ClassProperty	The field or nominal meta property to be balanced by. The same number of items (or a different score property) for each value in this field/property will be the result.	SpeakerID	Target Class (Age-Gender)
GroupProperty	The field or nominal meta property to be grouped by. The field is retrieved following the same rules as <i>ClassProperty</i> . Note: This parameter is only supported by the <i>ClassGroupBalancingFilter</i> .	n/a	SpeakerID
RandomSeed	Seed for initializing the random number generator. 0 will use a const. timer-based seed.		const.
DropThreshold	The minimum score that needs to be present for a class to be considered. If a label contains fewer items, it is discarded. This must be lower than or equal to the <i>BalancingThreshold</i> . Increasing this number generally results in <i>less</i> material.	0	0
BalancingThreshold	The minimum score that will be used as a threshold. Classes with lower scores will not be taken into account for balancing level determination, but may still be included. If no class has a higher score than this value, the value becomes the threshold. Increasing this number generally results in <i>more</i> material.	250	0 / infinite *
BalancingThresholdMax	The maximum score that will be used as a threshold. Only used if > 0. Classes with higher scores will not be taken into account for balancing level determination, but may still be included. If no class has a lower score than this value, the value becomes the threshold.	0	0
ScoreProvider	A script that is used to compute scores for each item. If none is specified, each item gets a score of 1. Here, <i>Duration</i> means that the score is based on the duration of the underlying corpus file.	Duration	---
PerClassFilterEarly	A script that is used to filter the list for each class <i>before</i> threshold computation.	---	RandomSubsetFilter
PerClassFilterLate	A script that is used to filter the list for each class <i>after</i> file selection.	---	---
DetailedReport	If <i>true</i> , a balancing report will be sent to the logfile including a list of all classes.	true	true
DetailedInstanceList	If <i>true</i> , the names of all selected items will be written to the logfile.	false	true

Table 5.5: Parameters of the item list filter scripts *ClassFileBalancingFilter* and *ClassGroupBalancingFilter* used for shuffling and balancing of material. Each *Value* column indicates the configuration for the first and second script, respectively. The * denotes a value that depends on the data set.

for all feature records (frames) belonging to the same file (utterance)⁶.

When a GMM was trained, it can be exported. Export is once again facilitated through a feature extraction script named **Export Utterance GMM**, which is run for each data set, although this is a slightly special case, since there is no input feature table or corpus file list. The extraction script is **ClassifierFeatureTableExportFeatEx**. It is meant to produce a feature table containing a representation of a multi-label classifier, with one row for each single class. The actual values depend on the classifier being exported. Any other input specified to the script is ignored. The script has three parameters: The classifier, which is set to the multi-class GMM for the respective data set, the name of the new column that will receive the class label being exported (we use the default *FileID*), and the script that generates the actual representation. The latter is set to a script particularly tailored to GMMs that can produce the supervectors, i.e. the stacked means. It could also add weights and variances to the vector if requested. The output vectors are saved to a **MultiFileStore** table.

5.5.5 Feature Space Reduction

Before the SVM can be trained, the feature filtering step is performed. Since we store the feature list in a persistent feature list cache, we introduce an intermediate step at this point, which clears this cache. We have seen in Section 4.1.6 that the feature filtering is based on the threshold of metric called *variance ratio* for each feature. To obtain the threshold for each feature, the metric has to be computed for each feature similar to the silence threshold. The feature extraction object we created to do this is **Compute Feature Filtering**. It employs the **VarianceFilteringFeatEx** extraction script, which in turn computes the variance ratio for each input sample and produces an output table containing pairs of feature ID and variance. This script has a special property: since it is both a feature extractor (through the **IScriptFeatureExtractor** interface), as well as an item list filter (through **IScriptItemListFilter**), i.e. it can also be used to dynamically filter a list of features. The input source for this feature extraction script is the *train* data set of exported supervectors. Again, an event handler is registered to be called when the extraction has finished, where the threshold is obtained and saved.

The feature removal is performed in the following step. Here, the feature extraction script **Apply Feature Filtering** is applied to each supervector table. This component makes use of the **FeatureFilteringFeatEx**, which is a generic script used to filter the features in a feature table; it is essentially a column reduction performed on a table. This column reduction has one parameter that points to the filter to be applied. In this case, we can exploit the dual-interface property of the **VarianceFilteringFeatEx** script from above, and specify it as a filter script. We set its *Threshold* parameter to the variance ratio threshold computed in the previous step. The feature filtering script has a second parameter that refers to a feature list cache, which is basically a list of feature names. If a name is specified for this parameter,

⁶If speaker-specific models are to be used, this could be changed to the *SpeakerID* property, which will automatically be obtained from the meta data in the corpus.

the list of filtered features is saved in this cache and will be used in all subsequent runs of the script, instead of computing the filter again each time. Hence, the time-consuming part of the filtering is needed to execute only for one of the data sets. If feature space optimization is not used in an experiment configuration, this step can be skipped.

5.5.6 SVM Training

The following step is the training of the target SVM(s). For representation of the SVM, another classifier object is added to the project. In this case, the `SVMlite` script that connects to the external *svm-lite* tool was used as classification algorithm. It has several parameters, but only the kernel function was set. The target-vs-nontarget cost factor (or *j*-factor in *svm-lite* terms) is computed automatically based on the data. The output models are automatically converted from a textual representation to a more efficient binary format. Input to the SVM are all feature-filtered supervectors from the *train* set. A rank normalization is applied as post-processing script to these input features. The class property is set to the final age/gender class. The SVM training script generates a multi-label wrapper around the binary SVM classes, and allows multiple classes to be trained in parallel (this will spawn one child job for each class during the experiment execution).

5.5.7 Score-Level Optimization and Evaluation

The final training-related step is the score-level optimization. A single evaluation object has been created in SPEACLAP for this purpose and was named `DevTest`. It is configured to evaluate the target SVM classifier and archive all results (except for the classifier). In addition, the input features are overridden, otherwise the evaluation would occur with the same *train* set used for the classifier generation. However, the configuration is slightly different for this and the next step. These changes are part of the experiment script. For score optimization, the script is run twice: once for the identification task optimization and once for detection task optimization. In both iterations, the input features are set to the *test1* supervectors. However, for the first run, the score modification script is set to `ComputeIdentificationDecisionThresholds`, while the second run employs the `ComputeDCFThresholds` script. In either case, the evaluation produces only the original scoreset, but stores score modification data as “extended results” together with the basic evaluation results such as the scoretable (see Section 5.3.5). An optimization that has been applied for the second run is to specify the `UseClassificationResults` property of the evaluation configuration to point to the scores from the first run. Since the original scores are obviously the same for both tasks, they do not need to be obtained twice.

The evaluation of the *test2* data set using the score optimization applied is the last step in the experiment. Like in the optimization step, the SVM evaluation is run twice. This time, we set the parameter `UseScoreModifications` to the previous evaluation results, which means that the score modification structures generated in these runs will be applied to produce a new scoreset. In the end, there are two evaluation results *ID* and *DET*, which can be

analyzed with the reporting functions presented in Section 5.3.5. Several of the charts that are produced have been included in Section 4.3.

5.6 Evaluation

Much of the motivation for the development of SPEACLAP originates from drawbacks or limitations of other utilities with respect to performance and scalability. The previous sections have already mentioned some of these limitations and how they were technically solved. In this section, we want to present how the platform deals with a particularly challenging and data-intensive task.

We will report the performance metrics on the same experiments that were performed during the FRISC parameter exploration. For a description of the setup and hardware, see Section 4.3.1. During the execution of the experiments, timing information for all tasks were accurately recorded through the logging features of SPEACLAP. Using this information, we are interested in two metrics: (1) the time consumed by the whole experiment, i.e. including training, and (2) the speed of the actual classification process in relation to real-time.

For simplicity, all results in this evaluation have been obtained using the design-time implementations of the classifiers within SPEACLAP. For the runtime performance in a live system, this means, that the numbers reported here represent a lower boundary of what can be expected. However, the results and processing on the *test2* data set technically correspond to the workflow that the on-line system would follow.

Table 5.6 lists all results from these measurements. The table layout reflects that of the results table in the main experiment section. The column *Training & Eval* contains the full experiment duration, i.e. the processing of all data sets. The columns under *Evaluation* refer only to the processing tasks of the *test2* set, on which the actual numbers were reported. These processes can be limited to MFCC feature extraction using *HTK*, MAP adaptation of the GMM, and the classification of the supervector using the SVM. All of these are identical for the *test1* set, and the first two for the *train* set. The percentages of these sub-tasks are computed relative to the whole evaluation time. Auxiliary jobs in SPEACLAP, such as the copying of the stacked means from the GMM, have been left out of this calculations since they do not exert a significant impact in the tests, and would do so even less in a live system. The real-time metric provided in the last column is obtained by relating the evaluation time to the length of the material in the *test2* set, which is approximately 216 minutes.

As can be seen in the table, a full training run takes between 5 and 30 hours, and the duration depends very much on the chosen parameters. With a few exceptions, the training times and evaluation times correlate. The most visible exception is the k-Means condition, which is reasonable since k-Means takes a considerable amount of extra time that occurs only during training. The feature extraction time is largely constant, although it slightly increases with higher frame rates. In average, feature extraction makes only 8% of the whole pattern recognition task. GMM adaptation obviously works faster when less frames are used for adaptation, for instance in the case of increased silence filtering. It generally corresponds

Experiment			Training & Eval	Evaluation							
				Feature Extraction		GMM Adaptation		SVM Classification		Total	
Position in System	Experiment # / Parameter	Value	Duration (minutes)	Dur.	% Total	Dur.	% Total	Dur.	% Total	Dur.	% Real- time
Front-end	(1) Step Width	5ms	766	8	5%	110	63%	57	33%	175	81%
		10ms	428	7	9%	55	71%	15	19%	77	36%
		25ms	528	6	14%	22	52%	14	33%	42	19%
		50ms	288	6	12%	30	59%	15	29%	51	24%
	(2) Window Size	1x step	738	7	5%	111	85%	13	10%	131	61%
		3x step	766	8	6%	110	76%	27	19%	145	67%
		6x step	709	8	6%	110	83%	14	11%	132	61%
	(3) MFCC Coefficients	12 (1-12)	709	8	6%	110	83%	14	11%	132	61%
		19 (1-19)	925	8	5%	123	77%	29	18%	160	74%
		8 (12-19)	657	8	7%	107	87%	8	7%	123	57%
(4) Delta Coefficients	No deltas	709	8	6%	110	83%	14	11%	132	61%	
	With deltas	1162	8	4%	146	75%	41	21%	195	90%	
Frame Filter	(5) Intensity- based Removal	0%	709	8	6%	110	83%	14	11%	132	61%
		20%	679	8	7%	89	79%	15	13%	112	52%
		40%	543	8	9%	65	76%	13	15%	86	40%
		60%	443	8	13%	43	67%	13	20%	64	30%
GMM Training	(6) Number of Mixtures	64	589	8	7%	105	89%	5	4%	118	55%
		128	709	8	6%	110	83%	14	11%	132	61%
		256	1064	8	5%	122	70%	44	25%	174	81%
		512	1764	8	3%	138	49%	135	48%	281	130%
	(7) MAP Relevance Factor	0.1	754	8	11%	53	72%	13	18%	74	34%
		2	753	8	11%	53	71%	14	19%	75	35%
		16	709	8	6%	110	83%	14	11%	132	61%
		100	748	8	11%	53	72%	13	18%	74	34%
	(8) Initialization	Random	754	8	11%	53	72%	13	18%	74	34%
K-Means		981	8	11%	51	71%	13	18%	72	33%	
NVC	(9) Feature Removal	0%	754	8	11%	53	72%	13	18%	74	34%
		10%	706	8	6%	112	85%	11	8%	131	61%
		30%	691	8	6%	112	88%	8	6%	128	59%
		65%	680	8	7%	112	91%	3	2%	123	57%
SVM Training	(10) Kernel Type	linear	754	8	11%	53	72%	13	18%	74	34%
		poly. (2nd)	825	8	5%	112	72%	35	23%	155	72%
		poly. (3rd)	882	8	5%	112	67%	47	28%	167	77%
		radial basis	1105	8	7%	112	91%	3	2%	123	57%

Table 5.6: Table containing the time performance of SPEACLAP during the FRISC main experiment series. The rows correspond to those in Table 4.4 on page 126. The row highlighted in green corresponds to the best parameter configuration. Times are given in minutes (rounded). The last column shows the percentage of real-time (e.g. 80% = 8 seconds for a 10 second utterance), which is further visualized by the color coding.

very much with the overall duration, since it represents the largest part (76% average). The number of mixtures also affects the GMM training time, but not as much as it impacts the SVM. The SVM is apparently very much dependent on the size of the supervector. In the “worst” case of 512 mixtures, the time share of the SVM, which is at 17% on average, rises to 48%, which is almost equal to that of the GMM in this case. Likewise, feature space reduction by 65% reduces the SVM share to only 2% of the eval time, which is clearly an over-proportional influence. Also, the higher-order polynomials require significantly more time during classification.

Overall, the classification performance is faster than real-time (and therefore acceptable) in all cases except the 512 mixtures condition. The best performance measured was 19% real-time, and 34% real-time for the best parameter configuration, which is clearly below the goal formulated in the research questions. There does not seem to be a general correlation between speed and the error rate. Considering that these results were obtained on a platform not optimized for on-line throughput, substantially better numbers are to be expected for embedded implementations. Such implementations are not only freed from much of the overhead of the development platform, such as logging and data set traversal, but they can also be optimized speed-wise: In [Feld \(2005\)](#) for instance, the author has suggested an optimization that has shown to reduce the runtime of feature extraction by a factor of 9. This and similar optimizations ensure that not only will Speaker Classification become faster, but also that less powerful platforms can run the technology as well.

5.7 Summary

The previous sections have given the reader an idea of the capabilities and workings of SPEACLAP and the rationale behind. Many of the features that are essential to modern machine learning environments can be found in the framework. Like with any other software of that type, no claim is made that it is by any means complete or can be applied to every problem in the field of audio-based classification. Through its extensible nature, however, the groundwork has been laid to provide a good starting point for those situations that some feature is not natively supported in the IDE. In the course of the FRISC experiments and other related research projects, the environment has grown and was steadily enriched with functionality. Over time, in order to try out new ideas and parameters, the amount of work needed to implement the process has steadily decreased, which can be seen as confirmation that many common tasks are now readily available without further investments.

Since the scope of this framework has extended considerably beyond the initial demand of being simply an implementation of the FRISC pattern recognition system, a lot of effort has been put into prospective architecturing and solid engineering of the codebase. Consequently, upgrading existing and adding new features in the future is a straightforward procedure and more than likely to happen as new, related research questions receive attention.

The SPEACLAP workbench is freely available on the author's website⁷. The ongoing optimization of the utility is an aspiration of the author and is believed to benefit the community as a whole, by reducing the overhead associated with experimentation, providing new insights on data, and speeding up development of solutions in general. Especially the ability to build embedded modules optimized for use in real-world applications is a distinguishing factor rarely found in other tools. The current state of the IDE code base is not quite production-quality, since that would require a broadened user base to regularly use the tool, but certainly fit for serious research work.

⁷<http://www.dfki.de/~mfeld/>

6 Achieving Personalization using Speaker Classification: Applications in the Automotive Context and other Domains

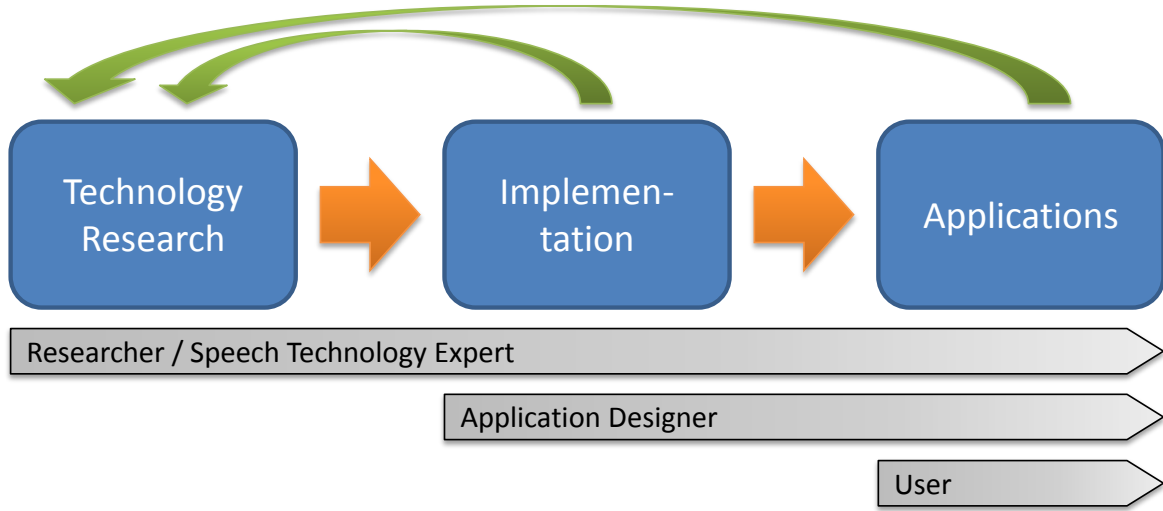


Figure 6.1: The field of Speaker Classification covers aspects of theoretic research, implementation, and application, which have mutual influences. Researchers, application designers, and users are involved at different stages.

In the previous chapters, we have developed the foundation for Speaker Classification as a technology that allows us to non-intrusively determine age, gender, and other properties of a speaker. However, it does not end there. The technology does not exist in a vacuum, but it is meant to serve as part of an application or service to the user. This work puts a strong emphasis on considering the full loop (see Figure 6.1), spanning from development of the technology, over its implementation, and its use in concrete domains. While research without a concrete practical application of the outcome might still yield useful results, a more aimed strategy is supposed to have a much higher expected impact. The reason for this is that, as seen in Figure 6.1, there is a mutual influence of technology and application, i.e. the application can have a decisive impact on the direction into which the technology develops during all phases of this process. Additionally, the applications serve as a further evaluation of the foundation: Given that we are able to successfully create functions that use the technology, it can be considered to have reached a certain level of maturity.

Chapters 2 to 4 have presented the general theoretical approach followed in this work, guided by a researcher’s point of view. In Chapter 5, we have moved more towards practical considerations by discussing how the approach can be realized to account for performance, scalability, resource-adaptivity, and generalizability. Here, in addition to the obvious omnipresence of research aspects, the view of an application designer was leading the descriptions, who might ask a question like “How can I compose a Speaker Classification system for ready use in application?” This can be considered a first step towards applications, although there was no tying of the framework to a *particular* domain or function yet. In this chapter finally, we further add the perspective of an end user working with a Speaker Classification-

enabled application.

From all application categories of Speaker Classification (see Section 2.3.1), personalization has the greatest variety of applications and most likely the biggest impact on daily activities. This is one of the reasons why the portrayals in this chapter are mostly located in this area. The chapter follows a two-step scheme: It first introduces a domain, and then presents a number of more concrete applications in that domain. The domains which were chosen are the automotive field, telephone-based services, and mobile end-user devices. The reasons for choosing these domains were numerous:

- They are very current topics with comprehensive research going on.
- Speech plays an important role in each domain.
- Each of these domains has a strong potential for UI personalization (in particular through the user – device association).
- More specifically, there are scenarios where age and gender knowledge can be used to improve the service.

One of these domains, the automotive field, was selected for a more in-depth consideration of various aspects and for the implementation of example applications employing and demonstrating the technology. Driving scenarios, through their very strict safety-dictated requirements, are especially dependent on the use of all possible knowledge sources to mitigate the distracting impact that complex information presentation has on the driver. There are numerous motivations for personalization, but most of its advantages fall into one of two categories. The first is what is still the task of most technology in a vehicle: to improve the *safety* of the driver. This can be an immediate improvement, e.g. through the output of warnings, or an indirect effect, e.g. through the reduction of workload on other tasks. The second is improved *comfort*. This could for example be comfort in the literal sense of an individually adjusted seat, or it could be a more suitable suggestion of a parking space. A more detailed analysis of this scenario is conducted in the corresponding section.

All attempts to study personalization – its goals, methods, and challenges – inevitably lead to certain recurring concepts. One of them is the creation of a model of the user. This and other basic concepts are summarized in the introductory Section 6.1. The subsequent sections will present each of the aforementioned domains separately.

6.1 Theoretical Foundation of User Adaptation and Personalization

Due to their relative importance, this section contains a review of some basic concepts needed to define and understand the application-related goal of personalization and the method of user adaptation.

6.1.1 User-Centered Design

In the early days of computing, human-computer interaction was focused entirely on the machine. The machine determined almost solely how the user (i.e. the person working with the machine) would interact with it. This was partially due to missing experience in the area, resulting in the view of the creator of an application to be adopted “by default”, but also due to the very strict requirements and limited flexibility of the underlying hardware. For instance, before graphical user interfaces become widely available, there was just no other option than to interact with the system using text commands, or even earlier, instructions hard-coded onto ticker tape. Such a view is however in most cases counter-intuitive, as people normally tend to think about *what* they want to do instead of what means might be available to accomplish that. This made it difficult especially for novices to learn how to use new systems.

Later, that focus shifted towards the task at hand, i.e. the interfaces were centering on the outcome of an action and transparently arranged the underlying functions accordingly. For example, operations related to customizing screen appearance are collected under the task of *screen customization* instead of being distributed over technically motivated categories of *screen hardware settings* (e.g. resolution), *color settings*, *text and font rendering settings* etc. Or, as another example, media players moved from the device-specific view (record vs. cassette vs. compact disc) of traditional stereos to a task oriented in terms of choosing between playing a particular album, playing songs with certain ratings or mood, synchronizing music with a portable player etc. This considerably improved the usability and learning curve. The concept of task-oriented HMI design is something that has been rediscovered by the industry over and over, and there are many guidelines explaining how to reach that goal best.

User-centered design is a complementary philosophy that is predominant in many areas today. It places a stronger emphasis on how tasks are perceived by the user, trying to understand and reproduce their needs and thinking. Being task-oriented alone is not a guarantee for a good design, because it is easy to set wrong priorities, e.g. to come up with task descriptions that sound useful but are rarely used in practice. Following this principle requires a thorough analysis of usage scenarios before the sketching of the actual HMI can start, which is accompanied by user studies to validate the findings.

One example where both task-oriented and user-centric aspects play a role is the *Useware Engineering* process (Zühlke, 2004; Görlich, 2009), in which experts from multiple disciplines contribute their knowledge and skills. The core of this engineering process can be considered the four-phase model depicted in Figure 6.2. It is based on the classical model of domain analysis, design, and evaluation, with an additional structuring phase. The processes also partially overlap, and are accompanied by a continuous evaluation process that returns feedback from users into the design processes. In the *analysis* phase, the requirements of the users and their tasks are studied, e.g. through interviews. The *structuring* phase formalizes these tasks into a concept where tasks are split up into usage building blocks, the so-called *elementary usage objects*, including user aspects in form of annotations. In the *design* phase,

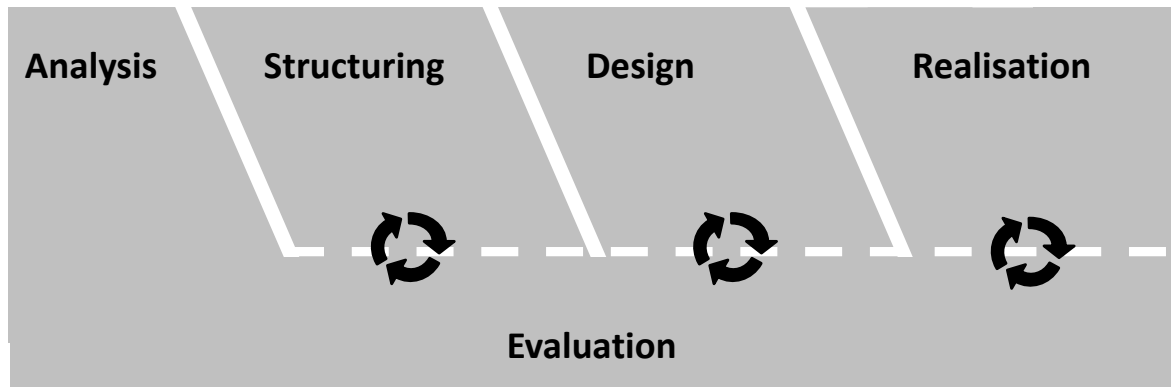


Figure 6.2: Phases of the Useware Engineering Process that has been applied in many different research and industrial projects. (Source: Görlich, 2009, p. 6)

these tasks are transformed into the user interfaces that form the system. This is done on multiple abstraction layers, another characteristic of Useware Engineering. The iterative evaluation process collectively describes different methods of testing designs and interfaces with users. This includes general usability metrics such as questionnaires, but also icon tests or other more specialized methods. Another vital component of Useware Engineering is the availability of a tool chain that supports the modeling efforts (Meixner, Seissler, & Breiner, 2011). For instance, the structuring phase is supported by tools for modeling in *useML*, a description language created to model task descriptions. This description is then converted into an *abstract UI* specified e.g. in the *Dialog and Interface Specification Language* (DISL), and then into a *concrete UI* language such as the *User Interface Modeling Language* (UIML). Only the last step, the realization, will render this into a final UI such as *Java* or *Adobe Flash*.

Finally, personalized or user-adaptive design is a special case of a user-centered design, and this is where our efforts should finally lead us: It does not only consider the demands of users in general to produce a well-designed but overall immutable interface, but rather tries to match the *current*, individual user as good as possible, thereby changing the interface if necessary according to some rules, possibly in real-time. This will be described further in Section 6.1.5.

To give a better idea of the difference between the philosophies mentioned in this section, Figure 6.3 shows a comparison between them in form of an example scenario involving usage of the in-car navigation system. Typical tasks in this scenario might include entering a destination address, following the map view, locating gas stations when the fuel is low, and possibly finding shopping opportunities. The first example shows a traditional menu that has low hardware requirements. Its structure is motivated by the architectural design of the system, which makes it quite non-intuitive for users and may require switching between menus and other views very often. The second case has a cleaner menu structure, which is based on tasks. Users can now find all navigation-related tasks on the same screen in a contextual menu and do not have to browse the whole menu structure. Although the system

might be theoretically effective, it does not much consider human factors. The third example therefore replaces menu-based navigation with touch-based input, and uses font sizes, icons, and a layout that is supposed to perform better according to various studies¹. Additional information might be added to support the user when relevant, such as the fuel indicator. The last example finally shows how the HMI can be personalized by being user-adaptive. Here, the visuals have been increased to counter the short-sightedness of the driver. Other tasks, which the user would otherwise have to initiate manually, are automatically performed by the system.

It is important to note that this progress is not simply a result of technological advance. Even today, one constantly encounters applications and devices that reveal an interaction behavior that was obviously designed with neither the task nor the user in mind, but simply reflecting a more or less ordered collection of possible operations. It is the duty of the HMI designer to ensure that concepts for a well-designed HMI are actually applied, and to constantly verify their efforts.

6.1.2 Personalization and Customization

The term *personalization* can be broadly defined. When applied to HMI, it generally means that a system can appear and behave differently depending on one or more user-related variables, including *who* uses the system, in what *state* the person is, *when* he uses it, what his *preferences* are, and so on. The term is also commonly associated with the web and adaptive hypermedia. Here, one of most well-known examples is the introduction of personal recommendations by the retailer Amazon by evaluating browsing histories and product ratings across users. Blom (2000) defines personalization as “a process that changes the functionality, interface, information content, or distinctiveness of a system to increase its personal relevance to an individual”. Further, according to Mackay (1991), it is also important that the behavior is preserved over time and across sessions.

Used in close conjunction with personalization is the term *customization*. Unfortunately, there is no common understanding in literature of the difference between these terms, except that both refer to the change of a system. Some authors make no difference between them at all, however most do. One possible definition is that customization is based on data explicitly given by the user (e.g. ratings), whereas personalization is done on implicit information (e.g. sales). However, this definition is difficult to apply in practice for two reasons: First, the origin of the information is sometimes unknown, in which case neither term could be used. And second, it generates borderline cases, such as a system performing an implicit action based on explicit input. For example, the user stating “I like green” might cause the system to change the background color to green, which would be difficult to classify using this scheme.

A more common view is to use the two terms depending on the initiator of the change: If

¹Note that this example merely shows aspects that *might* be changed for the better; it is not necessarily better in this particular design. Also, depending on the input modality available, a menu could also still be the better choice.

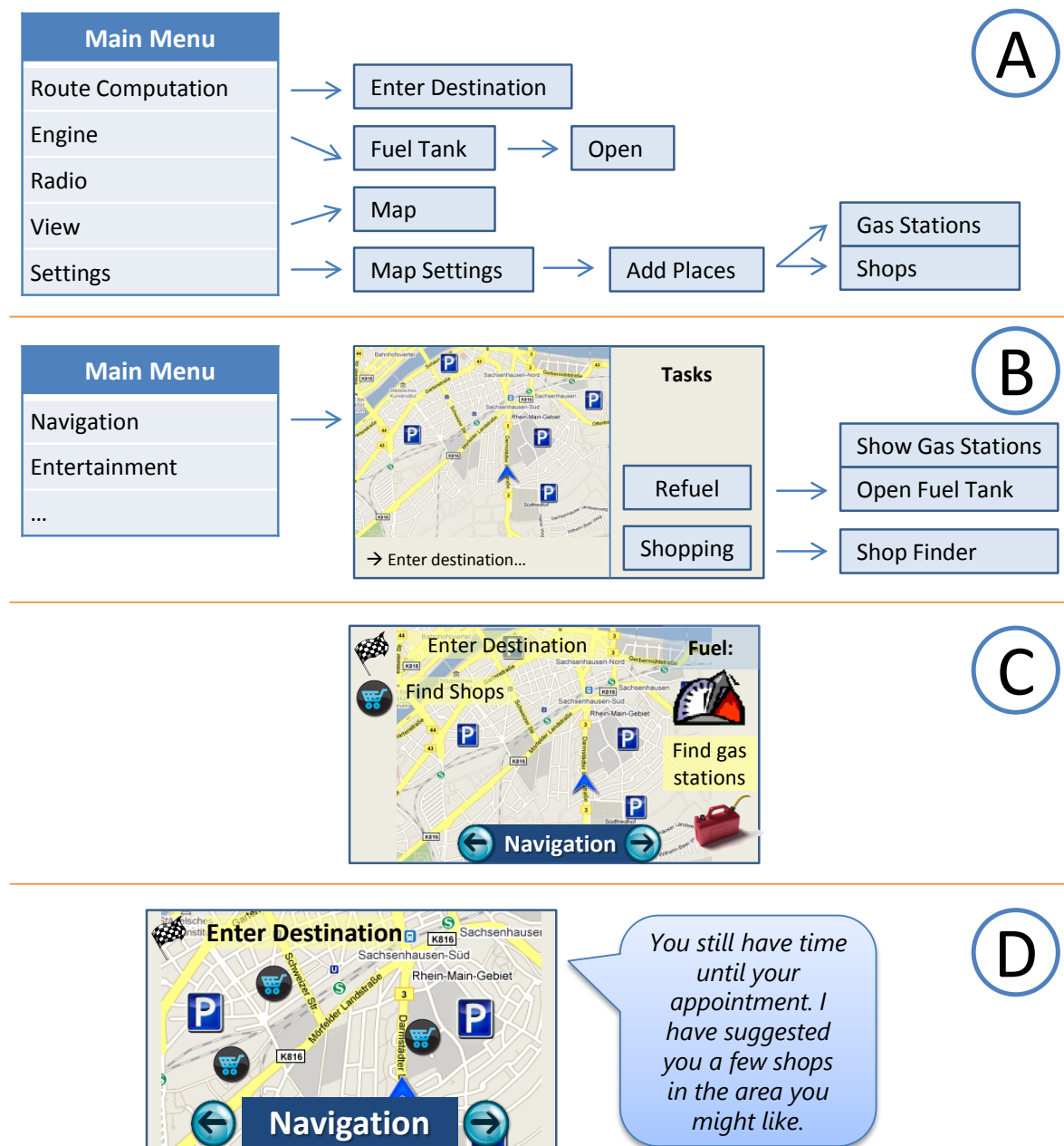


Figure 6.3: An in-car HMI designed following different design principles. (A) Traditional, system-focused design. (B) Task-oriented design. (C) User- and task-oriented design. (D) Personalized/user-adaptive (and task-oriented) design.

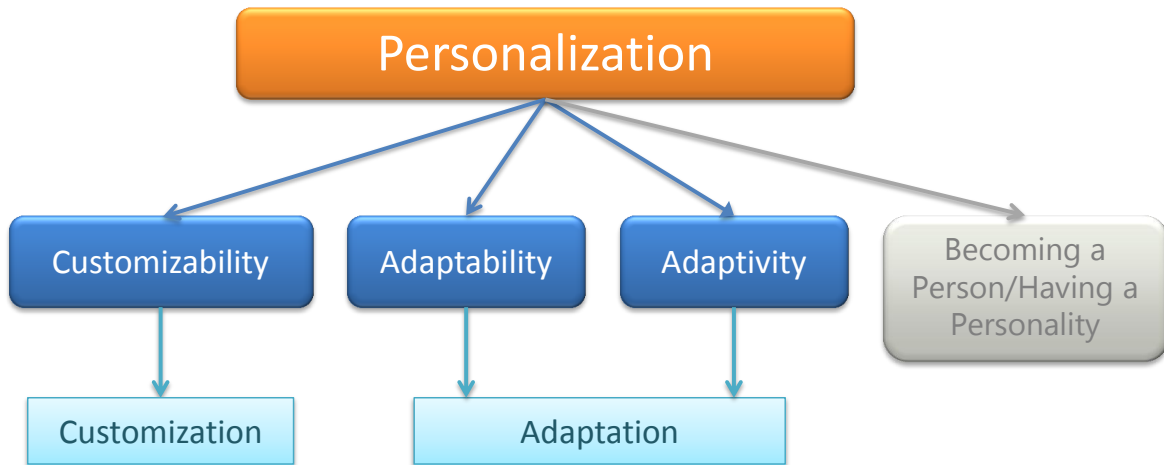


Figure 6.4: Scheme of four different system properties that enable different ways of personalization and two methods to achieve them. (Source: Endres et al., 2010, p. 2, extended)

the change is system-initiated, it is called personalization, whereas if it is user-initiated, the term customization is used (Sundar & Marathe, 2010). Personalization can also be considered the superset, as in Blom (2000): “Personalization can be system or user-initiated, the former often being described as customization.”

Another possible understanding is that customization is aimed rather at larger user groups than personalization. Tseng, Jiao, and Wang (2010) state that “in customization, customers are classified into different market segments based on different known customer needs identified by mark[et]ing analysis”, whereas “personalization is to achieve satisfying each customer as an individual”.

A similar position is taken by (Endres et al., 2010) when they say that customization or “being tailored to” is “an object or system [...] built or modified to fit the needs of one particular user or a particular user group”, so it is usually more final, or at least more difficult to change the customization once in place. However, they treat customization as one of several meanings of personalization, as depicted in Figure 6.4. The remaining three meanings are based on a scheme by Jameson (2001) and refer to adaptable systems, adaptive systems, and infusing personality. This view is generally adopted in this thesis. An *adaptable* system can be changed by the user himself, and requires no special knowledge or skills. It corresponds largely to the “user-initiated” aspect other authors have called customization (see above). Likewise, an *adaptive* system performs the adjustments without explicit user intervention, such as in the “system-initiated” case from above. The fourth meaning is rather special and has little relevance for most applications; it refers to the system taking over the personality of a character or (imaginary) becoming a person. It is important to note the difference between adaptable/adaptive systems and the adaptation process itself. As used throughout this thesis, the actual adaptation *process* (see Section 6.1.5) will be identical both

in the adaptable and adaptive case. However, all application examples presented herein will refer to adaptive systems, which corresponds to the case where information obtained through Speaker Classification is used.

The motivations behind personalization are as manifold as its manifestations (see e.g. Blom, 2000). They include providing more comfort, saving the user time, reducing memory load, enhancing safety (especially in the mobile context), allowing the user to explore new features, helping people gain access to information, accommodating individual differences, etc.

Personalization of the HMI requires a couple of ingredients to work: One of them is the data (or rather knowledge) upon which our decisions will be based, which is also known as the user model and is introduced in the following section, in conjunction with ways to acquire and represent it. In the most simple case of customization, these are the settings the user wants to apply, such as the background color. The other is the logic used to apply the model to our interface in order to customize or adapt it.

6.1.3 User Modeling

In order to build user-adaptive applications, one needs to have knowledge about the user – collectively termed a *user model* or *user profile*². In the most general case, such a model contains anything associated with that person that may aid the application in making decisions with respect to what to present and how to present it. Some common examples are identifying, demographical, and physiological information about the user (e.g. name, age, height, address, etc.). Advanced properties might describe preferences, skills, limitations, and needs (e.g. favorite music, driver's license, swimming skills, shortsightedness, etc.). Knowing that a person has bad sight could for example justify the selection of a larger default font size.

Even if not particularly mentioned, user models are inseparably linked to personalization. Every application that allows the user to choose settings and preferences implicitly builds a model of that user. The more powerful and complex user models however have emerged from artificial intelligence, most prominently from dialog systems (Wahlster & Kobsa, 1989) and are nowadays used in a plethora of other fields. In this regard, instead of using different user models for each application, a conceptually more promising approach can be seen in a single user profile that contains all the data and is shared by all applications. The method is called *ubiquitous user modeling*, since it is very closely related to the idea of *ubiquitous computing* as envisioned by Weiser (1991). Heckmann (2005, p. 5) defines it as the “ongoing modeling and exploitation of user behavior with a variety of systems that share their user models”. The knowledge management component presented later will serve a similar purpose, but is limited to the in-car ecosystem. However, there are obviously also reasons (technical or social) that might oppose the use of a global user model in a given case.

Being a subset of an application's knowledge base, the user model is usually accompanied by other models, such as a vehicle model, a physical environment model, models of other users etc. These models will be collectively referred to as the *context model* in this chapter.

²Some material refers to the *user profile* as the persisted form of the user model.

The *context* is defined as containing those application-relevant aspects which are dependent on but not directly related to the user at a given instant. This of course still leaves the boundary between user and context fairly blurry, but this goes in conformity with how the term is generally used.

User and context models are different in contents and usage, but they are created using the same techniques of *knowledge modeling* and *knowledge representation*. In short, they deal with the question of how knowledge can be gathered and formalized so that it can be used efficiently applied in knowledge management systems, e.g. for inferencing. One essential difference between a data storage system and a knowledge management system lies in the way that complex relations between instances can be described. While databases can have tables with different layouts, the relations between tables are very limited (usually only certain key constraints can be defined). Also, these systems are not designed for a very large number of different table layouts paired with potentially only few instances in each table. However, due to inheritance and other class relationships, the structure of instances can become very dynamic. A knowledge management system therefore has to be flexible enough to accommodate any such structure in an efficient way. Thereby, the concept of separating the structure from instance data has proven successful. The domain-specific structure is called the ontology. It describes the layout of the instances (in this context also called *concepts*) and all possible relations between them, creating a complex semantic construct.

Additionally, user knowledge representation deals with the syntactical aspects of the representation, such as a particular language. The user modeling work by Heckmann (2005) for instance is based on the popular standards *Resource Definition Framework* (RDF), *Web Ontology Language* (OWL), and constructs a specialized *user model exchange language* (UserML) extension from it.

The aspects of user modeling that the major part of this thesis deals with is that of *knowledge acquisition*, i.e. the entering of new facts into the knowledge base. There are several ways for a system to obtain knowledge updates, for instance

- through a user-triggered explicit input (e.g. if he adjusts the color theme of a graphical UI)
- through a system-triggered explicit request for feedback (e.g. if the system asks the user for her name)
- by a non-interactive, but invasive form of sensing (e.g. using biosensors)
- by a non-intrusive form of sensing, which can still be interactive (e.g. using cameras or Speaker Classification)
- from an external, possibly contextual knowledge source (e.g. emails, traffic information, expert knowledge)
- by inferring new facts from existing knowledge (e.g. concluding from high humidity and turned-on wipers to rain).

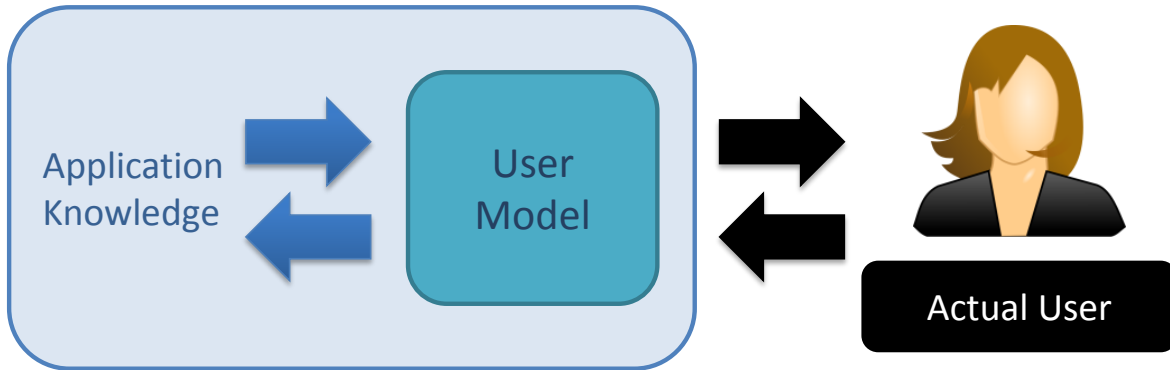


Figure 6.5: Relationships between application knowledge, user model, and the user to which the model points.

As we have seen, the human voice is a particularly rich source of such information about the user. It is especially valuable because it can be exploited without explicitly requesting attendance from the user: Any utterance targeted at whatever recipient, be it another individual or a voice control interface, is sufficient to serve as a basis for the analysis of paralinguistic features. Hence, it is also called a *non-intrusive* source of information. In general, non-intrusive knowledge acquisition is clearly preferred because it does not cost the user any time or cognitive resources. The user age and gender are two selected properties that are part of the demographic knowledge contained in the user model. Continuing on the path taken in previous chapters, these two features will receive particular attention when we turn to concrete applications.

6.1.4 Facets of User Identity for Applications

In this thesis, we consider user properties that are either part of the user's identity (age and gender) or, in case of one of the research questions, refer to the identity itself. Since the information from either source can be complementary, substituting, superseding, or conflicting, it is worth to have a look at how these aspects of identity play together as part of the user model.

The user model takes the role of the central container for all aspects related to the identity, properties, and state of a particular user. Not all knowledge that is learned relates to a user, but with respect to personalization, user-related knowledge certainly has a special status. One the one hand, it is only possible to extend the user model if the new knowledge can be matched to a model. On the other hand, a user model needs to refer to a real entity to be of use (see Figure 6.5).

In an idealized world, each user profile would have a 1:1 mapping to an actual person, and there would be a unique identifier (user id) for each profile to enable the association. In reality, things are more complicated. Users can have more than one profile, either intentionally to be more in control of information management or unintentionally because of a lack of application

functionality for sharing profiles, with information distributed between them. This fact will be neglected for this thesis since (1) we assume all applications utilize the user profile functions of the corresponding system component, which is designed as extensible as possible to make any additional profile creation for technical reasons unnecessary, and (2) with advanced privacy management features, users are assumed to be able to achieve full control over information sharing with a single profile.

Yet, even then, the 1:1 user profile mapping is not sufficient. Consider the following example from the automotive domain: If someone enters a car and starts giving voice commands to the system, she is identified using her voice, and her existing user profile is loaded if it is known to the car. If another person enters the car without speaking or using any other means of identification, but can be detected e.g. through weight sensors or interaction with the window controls, it is still better to have a new “anonymous” profile that contains only the weight and seat position than no profile, even if the person does have an existing profile stored in the car. Similarly, if the person starts speaking but cannot be identified, this very basic profile can at least be extended by the age and gender estimation, to offer personalization to the extent possible using the given evidence. If seats are not equipped with weight sensors or not all seats have their own microphone, it is also possible that different pieces of information (e.g. age from speech and temperature preference from the usage of buttons in the center back console) cannot be associated with the same person for certain, so that a person may have multiple profiles and there is not a clear indication as to the number of people inside the car. As new observations are made and knowledge becomes available, these user / profile associations can be updated.

This illustrates clearly that there are several levels of identity/identification that a user can pass through. Since they correspond directly to what the system *believes* to know about the identity of the user based on its sensors (i.e. there is some uncertainty involved as well), they can also be called *belief states* (see Figure 6.6). The presence of an estimated age and gender of a user is an example of a such a belief state. In this case, the system has a stereotypic belief about the user, such as being an *adult male*. This is more than just knowing the user’s existence but less than knowing his or her identity. The classified speech signal is the *evidence* which causes the user to shift between belief states. The actual analysis of observations has to be done by an evidence fusion component.

Without any evidence, a user’s state is *undetected*, when the system neither has any sort of user profile nor is aware of his existence. This changes when at least some evidence for the existence of a user is observed. If it cannot be associated with an existing profile, a new user profile is created, and it contains nothing but the new evidence. At this state, the identification level corresponds to *detected*, since the system knows of the user’s existence. It still does not know yet whether the user has another existing profile with the car or not. As an intermediate level, *classified* is entered when the user has been classified according to a particular criterion, e.g. age. The more specialized the class, the higher the identification level. This enables basic personalization effects, which corresponds to having a user-profile that is pre-loaded with information for users of this age class. This type of bootstrapping

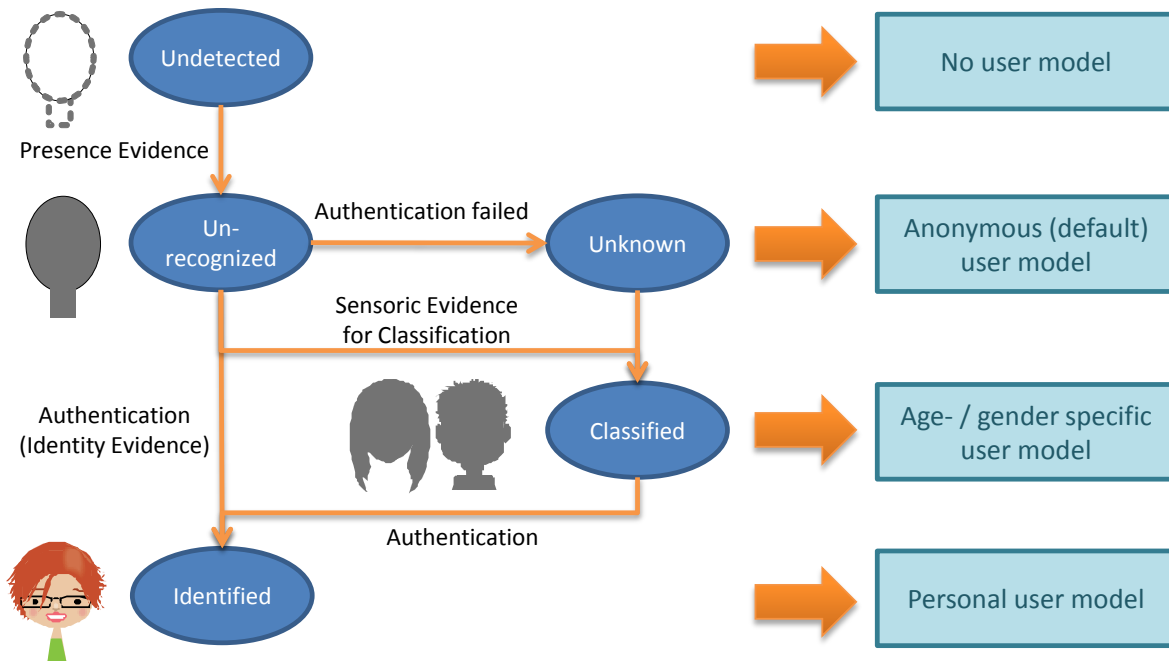


Figure 6.6: Schematic diagram of how evidence from different types of knowledge sources changes the system's perception of the user and activates specific or generic profiles. The circles represent belief states, the arrows evidence flow and the boxes user profiles.

is best implemented by using inference or adaptation rules based on the age. When some observation is made that contains sufficient evidence to positively match with an existing user profile, this profile is loaded and the identification level changes to *identified*. If on the contrary, the sensor information leads to the conclusion that the user most likely does not match any of the known profiles, this leads to the *unknown* state instead, which still conveys more knowledge than *detected* (namely the knowledge that the user is not known). The highest level of identification, *authenticated*, does not add new knowledge, but rather confirms the existing knowledge (e.g. by increasing its confidence). Since many identification methods, in particular the non-intrusive ones, are not perfectly reliable, the identification is subject to some uncertainty, and any application needs to be aware of this. For many personalization effects, this is not critical: When the seat is configured to a wrong position and state, this may be unfortunate for the user, but it can easily be corrected. (If we are talking about a very unreliable method of identification, that fails often, even that may be too inconvenient.) If however the user's reminders for appointments are displayed, or access to even more sensitive information such as payment options is given, it is reasonable not to rely on estimations or ambiguous sensor data to conclude on the user's identity. Rather, the application should await authentication information that cannot be misunderstood. An example are log-in credentials that are entered manually into a log-in form. This still leaves

room for non-intrusive solutions, such as a mobile authentication token carried on a radio chip.

Associating new observations such as incoming sensor data with a user profile is more straightforward in the vehicle than in other domains. Resulting from the constrained space, the rigidly installed seating options and safety regulations, the concept of the 1:1 relationship between occupied seats and users appears like a good choice. It is not difficult to come up with conditions where this is not the case (e.g. a passenger lying across all seats or a baby in the arms of her mother), but they are rare enough to set them aside for this work. In many cases, sensors can be associated with a particular seat. At the same time, user profiles are also associated with seats. Any knowledge acquired from a device that is specific to a seat can thus immediately be related to the user profile for the seat. If no user profile is assigned to the seat, the observation is stored for further processing, which can result in a new “anonymous” profile to be created, or the knowledge could finally be merged with an existing profile.

6.1.5 User Adaptation

Adaptation (sometimes called *adaption*) is the last ingredient needed to achieve personalization. Based on the available knowledge, decisions are made which can result in changes that are visible to the user. Personalization and adaptation are sometimes confused, yet the former merely describes a goal, and the latter a method (or rather a family of methods). Adaptation can also serve other purposes. A software could for instance also adapt to the platform it is running on, which is clearly not personalization. There are different ways of performing adaptation. They can be categorized according to several criteria:

Source. The source of the adaptation describes the knowledge upon which the adaptation is based. In this work, we limit ourselves to the case that the user model is part of the source, since we focus on properties obtained through Speaker Classification. In this case, we also speak of *user adaptation*. Typically, we can name a specific property that is responsible for it. A different case would be context adaptation, e.g. the case that the car behaves differently based on the weather. Further, adaptation can involve multiple sources.

Target. The target of the adaptation is where the effect will be visible, i.e. the part of the system that is changed. There are two main types to distinguish here: Adaptation of the *function* and adaptation of the *form*. Adapting the form includes “cosmetic” changes, such as of the graphical user interface (e.g. color or layout), the dialog system (e.g. the way errors can be corrected), and other input/output modalities. It can also affect the selection of modalities and their basic parameters. In other words, the content remains unchanged while the mode of presentation is altered. This type of adaptation is often designed at an abstract level independent from a specific function and therefore easier to generalize. Compared to this, adaptation of the function involves a greater variety of possible effects on the inner workings

of a system. They affect the *generation* of content rather than the presentation, hence more detailed knowledge of the function is needed, and which is why generic adaptation is less common in this case. Examples are the modification of filters that determine the selection of items, or the algorithm to be used in a formula (e.g. selection of a different route).

Initiation. This refers to the component which decides that an adaptation should occur, either by specifying adaptation rules or by committing a change. A rough distinction can be made between user-triggered adaptation (i.e. the user is adapting the system) or system-initiated adaptation (i.e. the system adapts itself). Within system-initiated adaptation, a finer distinction is possible with respect to the involved component. Certain types of adaptation can be undertaken largely independent from the function, such as improving readability for certain people. Such adaptation is said to be more general. On the contrary, when a concrete application or function initiates the effect, it is more specific to the task at hand. For example, when a warning is to be displayed to the user, a generic adaptation might increase the font size of the warning text for an elderly driver, but the warning function itself could override this when it decides that the warning is not relevant to the user, and hide it altogether. From the application's point of view, one can also speak of *active* vs. *passive* adaptation: In the active case, the application controls what should be adapted, while in the passive case, the application's requests to some presentation framework (e.g. the GUI) are interrupted by a proxy component (an adaptation framework) and transformed into adapted requests without additional input from the application.

Parameters. In some cases, the adaptation is constrained to being applied or not (e.g. an extra warning that can be displayed), while with others, there may be additional parameters that can be adjusted (e.g. the speed to which the TTS is changed).

6.2 Speaker Classification in the Automotive Domain

The automotive field is a very active area in terms of user interface research. This coincides with the rather recent introduction of many functions in vehicles that go far beyond the innovations of the previous decades, which were usually limited to low-level electronics such as airbags and ABS. Starting with the production-line introduction of navigation systems, both the pace and the intelligence contained in these systems has been continuously increasing, and with upcoming technologies such as electronic vehicles, it is going to stay like that for the next years. Functions like warning of obstacles, electronic displays of road signs, alternate road finders, parking guidance, trip planning systems, ecologic driving assistance, in-car messaging, entertainment and “navitainment” services, car-to-car communities, and many more all offer great benefits in terms of comfort and safety to the driver and passengers in the car. Further, they all have a strong potential for personalization, which can increase their value even more.

The modern car is also quite rich in terms of knowledge, which is the basis for personalization and adding intelligence. It is equipped with numerous sensors in the inside and outside, such as for speed, video, sound, passenger weight, proximity, temperature, weather, ice on the road, etc. It has profound geographical and infrastructural knowledge through its maps, which are regularly updated. It receives further information from the traffic management center, from OEMs, and from roadside units. It can communicate with other vehicles in the vicinity through car-to-car communication, and with the home or office over the Internet. Since the car accompanies the driver on many journeys, it can also “learn” much about her habits, e.g. the route to work, preferred types of road, driving style, typical shopping hours, and so on. In addition, the passengers bring their personal mobile devices into the vehicle, which would allow them to connect to its system and share up-to-date emails, calendar, and other information. Summarizing this, the vehicle can be considered an excellent *information hub* due to wealth of knowledge present in a single location. The more knowledge is available, the more powerful the application scenarios that can be realized on top of it.

But the challenge for HMI researchers does not only lie in the individual functions and services themselves, but also in keeping the cockpit a safe and manageable place for all passengers. If too many functions demand the driver’s attention or interfere with her tasks simultaneously, the sum of several positive functions can also become negative, i.e. functions supposed to reduce the workload might actually cause more distraction than before. Work is going on to accommodate this in terms of less distracting input modalities, better use of cognitive resources, coordination of output channels, and so on. It is unquestionable that many of these factors depend on the individual and the current situation, therefore, personalization can be of equally high value here too. As lead topic of the *Second Workshop on Multimodal Interfaces for Automotive Applications (MIAA)* (Feld, Müller, & Schwartz, 2010), personalization in conjunction with multimodality has been investigated as a possible strategy to avoid overloading the cockpit with hardware knobs or resorting to difficult-to-use menus.

In the automotive industry, as far as market-readiness is concerned, personalization ideas are still not very widespread. This is different from other areas where computer technology and artificial intelligence are found, such as personal computers, mobile phones and other portable devices, gaming platforms and further consumer electronics products. One reason is certainly a technical one: It is much more difficult to integrate technology into cars, due to electrical, space or weight issues. Related to this is that a common infrastructure, which enables the easy integration of components, such as a common operating system or a hard disk, is not present. There are little common interfaces between vehicle manufacturers, which makes the development of software for it a lengthy and complex process. Then, for good reasons, safety is a topic of prime importance in vehicles. Until recently, many functions that did not directly relate to safety or driver assistance were not even considered because of the potential impact on safe driving. This does not apply to personalization in general, but to some advanced functions which are candidates for it.

For equally good reasons, this is slowly changing. People themselves are one of the driving

forces here, because for younger generations, the so-called “digital natives”, mobile phones, Twitter and Facebook are services as naturally used as a radio by previous generations. Being used to staying on-line all the time, banning these services from the car is not easily accepted. Because the adoption of new services has been faster than their maturing, many countries have reacted by bans as the short-term reaction to the increased safety risk. However, with more advances in the safety-related aspects of the technology, combined with the increasing demand, a solution towards better integration is likely to prevail in the long run. Moreover, with fully computer-controlled cars already being possible and legal in some parts of the world, we are also going to see a reduction of the overall workload on the driver in the near future that will support and accelerate this process even more.

In this section, we are looking at a broad yet special domain. A considerable amount of background information would be necessary to learn about all the characteristics, research methods, common issues, etc. for someone who does not work in this area. The following sections attempt to give an introduction and highlight many points worth knowing, as well as give an overview of relevant work in the field. As far as completeness goes, the focus will rather be on selected scenarios that support and exemplify the ideas centered around speech-based personalization.

6.2.1 Overview and Categorization of In-car Functions

Having grown out of its mechanical origins, the majority of systems found in a car today are electronic systems and functions that fall into several categories (see Figure 6.7). In theory, all of them can have aspects that could be personalized to the user. For many of them though, this is less likely to happen, be it for practical or safety reasons. For instance, the function of ABS or an airbag is so hard-wired to the vehicle that it probably would not expose parameters that are user-specific. It also does not keep the user in the loop – these systems are designed to work autonomously and be ready under all circumstances. Further, these examples do not have a complex user interface that could be enhanced or simplified by a more personalized approach, which is one of the main goals of personalization. For these reasons, the following categories of (generally lower-level) vehicle electronics have been excluded from this survey: engine electronics (e.g. ignition system or throttle control), transmission electronics (e.g. automatic gear shift), chassis electronics (e.g. ABS, ESP), and most active safety (e.g. air bag)³. This leaves the following system categories for consideration with respect to personalization (note that some of them overlap, so a system can belong to more than one category):

- (Advanced) Driver Assistance Systems (ADAS)
- In-vehicle Information Systems (IVIS)
- Infotainment/Entertainment Systems (IES)

³see http://en.wikipedia.org/wiki/Automotive_electronics

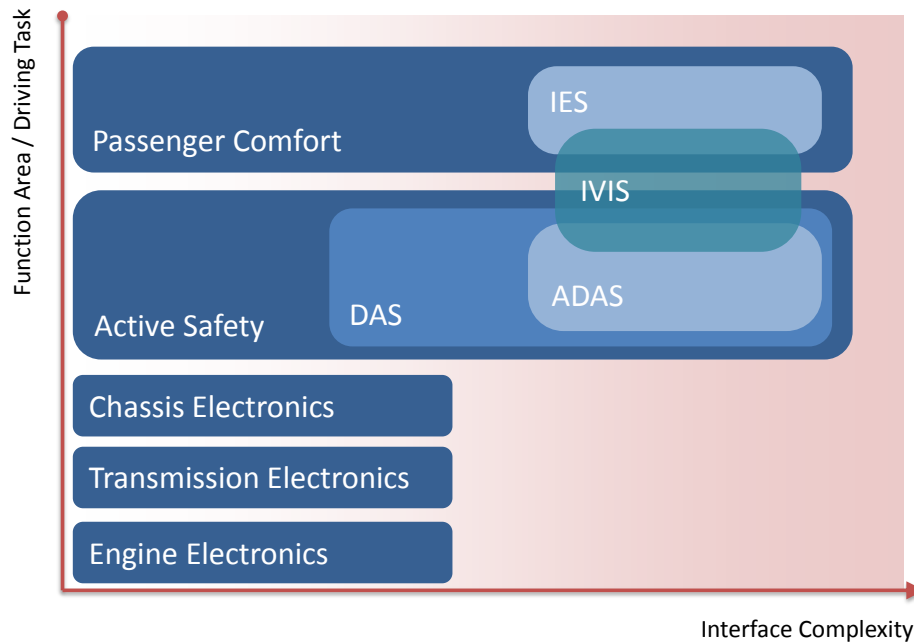


Figure 6.7: A possible categorization for electronic systems in vehicles. Systems on the right side generally offer a greater potential for personalization.

- Passenger comfort functions (include IVIS and IES, and also add more “basic” functions such as automatic climate control or electronic seat adjustment).

Development of technologies for vehicles was traditionally always split across three areas: Those supporting the actual driving process, those that are directly related to driving by adding safety to the process or traffic in general, and the remaining ones which are not directly related to driving. These three tasks, which are supported by various mechanics, technology and software systems, have been known in the automotive community for some time as the primary, secondary and tertiary task. Bubb (2003) defines the *primary (driving) task* as positional (longitudinal and lateral) control of the vehicle, i.e. mainly steering, stabilizing, accelerating, and braking. These operations are absolutely required in order to handle the vehicle and arrive at the goal of reaching the destination. The *secondary task* thereby refers to tasks that are caused by or needed in conjunction with the primary task to maintain safety, but do not relate to steering. Examples are controlling the turn signal, horn, wipers, cruise control, etc. The *tertiary task* finally describes all other tasks, which are not directly related to driving, but rather to information acquisition or comfort⁴. Examples are controlling the radio, air conditioning, telephone, or other services. According to Bubb (2003), navigation in the sense of route selection is also part of the primary task. Other sources consider navigation part of the tertiary task (Amditis, Andreone, Polychronopoulos, & Engström, 2005; Ussat,

⁴Some sources such as Schwarz et al. (2006) do not make the distinction between primary and secondary task, but rather subdivide the primary task into a stabilization, maneuvering, and navigation level.

2010) or do not count it explicitly as part of the primary task (Schwarz et al., 2006). One problem is that the term can be very broadly defined, and it is believed that much of the interaction with navigation system, such as the initial programming (which usually takes place before even the engine is started) or the exploration of points of interest or scenic routes, are essential neither for steering nor reaching the destination. Additionally, using a navigation system can involve arbitrarily complex interaction. Therefore, we consider navigation split across driving tasks, with cognitive and operational aspects of route planning focused on the primary task.

Even in the earliest series production vehicles, all of these aspects can be found. This has not changed with today's cars. However, the balance has shifted. Until a few decades ago, driving features had played such a distinguished role that the other areas only received basic attention. Later, because of technological advances and the urge to address the high accident rate, more resources were dedicated to safety, and resulted in systems like ABS, ESP, traction control etc. being built into cars. With respect to non-driving related functions, there has not been much change until recent years, limiting the accomplishments to amenities such as car stereo, air conditioning, seat heating, and electrical window openers. Today, we witness equally many or possibly even more breakthroughs in the category of systems supporting the tertiary tasks, such as the examples mentioned in the introduction to this chapter.

A particularly important type of system is the *advanced driver assistance system* (ADAS). Driver assistance systems by definition always support the primary task (Schwarz et al., 2006; Bubb, 2003) and do therefore always have a safety-related function. Their contribution can be in form of information and feedback, workload reduction, or active stabilizing measures. ADAS are additionally characterized over DAS by an extensive utilization of knowledge, e.g. from vehicle sensors, road infrastructure, or environmental data, and a more complex implementation and signal processing. Further, they allow direct interaction between the driver and the system (Schwarz et al., 2006). Common examples of ADAS are adaptive cruise control (ACC), collision warning/avoidance/mitigation systems (CWS/CAS), lane departure warning (LDW), lane change assistance, driver drowsiness detection, pedestrian recognition, automatic emergency brakes, traffic sign detection, adaptive headlights, and night vision (see e.g. Gründl, 2005). For ADAS, obviously, the more advanced a system is, the more potential for adding intelligence.

An *in-vehicle information system* (IVIS) is generally characterized by providing the driver with information, but not actively taking control of the vehicle. Schwarz et al. (2006) explicitly consider it to support the navigation layer of the primary driving task (which would make it a sub-category of ADAS). This restriction is not shared by all sources; according to (Vashitz, Shinar, & Blum, 2008), “they can also reduce boredom and drowsiness imposed by monotony”. The information provided by IVIS might for example refer to alternate routes, traffic congestion, obstacles, accidents, road conditions, weather, hazards, or technical problems with the vehicle. While IVIS are generally intended to improve safety, the opposite effect might occur if they are badly designed, e.g. when they present too much information at once, are hard to understand, or in the case that they fail.

Infotainment/entertainment systems (IES) are exclusively related to the tertiary task, and quite often targeted at rear seat passengers. Such systems might provide background information (e.g. history) about cities or landmarks, suggest places to visit, display enhanced maps, or provide traditional entertainment functions such as music, video, or games. If the latter are related to the current traveling context, e.g. a quiz on cities visited or a car-spotting game, they are also referred to as *navitainment*.

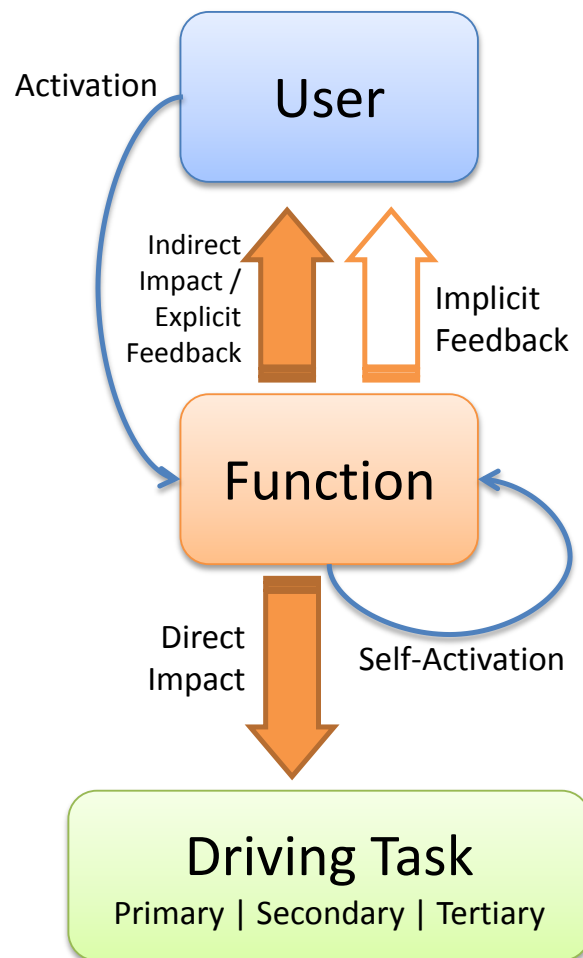


Figure 6.8: HMI-related aspects of in-car functions.

In addition to falling into the aforementioned categories and driving tasks, each function can expose a different degree of personalization and interaction as shown in Figure 6.8. The most common case today is that no personalization happens whatsoever. Functions in the car work in the same way independent from the person who uses them. Examples are the turn light and the braking pedal behaving no different for two individuals. A first step into the area of personalization is the ability to customize or adapt functions. However, even of the functions in cars that exhibit a certain degree of adjustability, most are not really

personalizable since they will not associate the changed setting with a user. The seat heating level or speaker volume are some adjustable, yet not personalized functions. An example for a personalized function are the seats in several new cars that remember the position based who enters the car, so different users can have their own configuration. Recalling settings is usually done based on the key used to unlock the doors. Even though the user model is very simple and not used for any other purpose, it exemplifies the basic concepts. Modern technology enables a more advanced level of personalization, which is characterized by a broader range of possible adaptation effects, a more complex relationship between knowledge and production of these effects, and an automated, i.e. non-intrusive, acquisition of knowledge, e.g. through Speaker Classification. Only very few systems in today's cars already implement this degree of personalization.

A different aspect of personalization is its visibility, which depends on the target of the adaptation types that occur. Some functions provide a clear user feedback communicating their presence. Other types of adaptation happen without the user noticing it, which can be positive or negative depending on the situation. Strategies that adapt the interface (or more generally the interaction), e.g. by adding items or changing the layout, are more likely to be noticed by the user, while strategies adapting the behavior of a function, e.g. suggesting a different route, can easily go unnoticed unless an extra piece of information is presented in one of the available output modalities. Some examples are on the borderline of both: If the steering is affected by an adaptive lane keeping assistance function, it will affect the internal workings of that assistance system, but it can still be noticed to some extent through the implicit feedback given by the moving of the steering wheel as an output modality in this case.

Lastly, we can distinguish personalization scenarios based on whether they are used with concrete systems or functions, such as a visual representation of an emergency vehicle's route for persons with a bad hearing, or whether they are rather general ideas that work with an array of functions, for instance the adjustment of message timing parameters (delay and duration) according to age and cognitive state of the user (which can be used in conjunction with all warning functions and many others) or a resizing of screen elements for users with reduced sight (which can be used with most functions that have any graphical output).

6.2.2 Characteristics of Personalization in the Automotive Domain

Looking more closely at the characteristics of the in-car scenario reveals several points that should be noted. What makes the car an almost ideal surrounding for personalization is the aggregation of information that takes place on a very constrained space. Through its primary purpose, the car already knows where the user currently is and where he or she often goes. It can derive information about the user's current state through the interaction with the vehicle as part of the driving task. As a consequence of the car taking increasingly over the role of a mobile office, all work-related items are present, which includes emails, appointments, contact data etc. External vehicle sensors, wireless communication with other

cars (car-to-car communication), and feedback from roadside units, such as traffic lights and “intelligent” traffic signs (car-to-x communication) provide a good overview of the current situation or external context. In addition, due to the physical conditions, instrumentation with sensors and other equipment is much more easily possible than in other spaces like an apartment or public areas. As [Brouwer, Hoedemaeker, and Neerincx \(2009\)](#) state, “the human is in a constrained (relatively fixed, ‘indoor’) position, sitting in an environment that can be relatively easily enriched with driver-state sensing technology”.

The cockpit layout does not only have advantages, but also drawbacks for hardware design. It is not possible to fill the vehicle up with devices like one could do at home, since the space is limited and every additional weight costs more energy to move. In terms of screen space, the requirements are similarly strict as with mobile devices: Only a small amount of information can fit on the screen at any time, which requires a careful selection of what is important. Although the screen space might be larger than on mobile phones, the driving context also requires a larger presentation that mostly cancels this advantage. Furthermore, the range of preferred modalities and available interactions is quite different from other mobile scenarios. Because of the primary driving task the driver has to attend to, she does not have her hands available for manual system interaction all the time. Apart from that, the location of the screen makes it difficult for many people to reach it for touch input at all. Speech, on the other hand, is a “hands-free eyes-free” channel, therefore it is much preferred to maximize safety.

The distinguished role that safety aspects play in the design of all systems which are put to use inside an automotive environment is also very characteristic for the domain. It is crucial that at any time, the driver can dedicate as much attention as needed to the primary driving task. Because the cognitive resources available to a human for the accomplishment of this task are naturally limited, it is easy to dissipate too large a share of these resources on less critical tasks to maintain safe driving. If the system was not evaluated for the effects it has on driver distraction, this effect and the consequences are completely unpredictable. Speaker Classification may aid us in estimating the available cognitive resources, which vary by person and situation. Particularly the car HMI is subject to much stricter rules and undergoes a more rigorous testing than interfaces in other areas. A common topic of discussion are moving images, i.e. animation or video. While they can be a very appropriate means to improve the look and feel of an interface on home systems, or convey the personalization notion of face-to-face conversation using virtual agents, they are not suitable for many situations in the car. Again, one has to carefully analyze the exact situation to find a resilient answer. For instance, a vehicle that is not moving because it is waiting at a traffic light when there is still plenty of time before it turns green again is a fundamentally different case from a high-volume traffic situation on the freeway. Here, the interruptibility is a criterion that applies specifically to tasks performed in the car: If the driving situation requires it, every function that is performed simultaneously with driving must be suspended immediately. Functions which discourage this, such as scrolling text or phone calls, are potentially more dangerous. It should be noted that most arguments with respect to distraction apply far less to passengers

and even less for backseat passengers, as long as there is no impact on the driver.

Driver distraction as a indicator of an overload of the cognitive resources can be measured. This is usually done by measuring the effect of a secondary task on the performance of the driving. An established driving task commonly used in distraction studies is the *lane-change task* (LCT, see Mattes (2003) and the ISO norm *Road vehicles – Ergonomic aspects of transport information and control systems – Simulated lane change test to assess in-vehicle secondary task demand*, 2010). The primary task consists of a driving simulator set-up, in which a test subject is required to switch between three lanes according to the road signs that appear every few hundred meters while driving with a constant speed. Simultaneously, the secondary task is performed following given instructions resembling the envisioned usage scenario. The standardized performance metric that is applied is the mean deviation from a normative model, i.e. an “ideal” line, which accounts for both lateral deviation as well as reaction time. Adaptive behavior presents a special challenge for a user study: Since it is based on the user’s actions and state, it is difficult to follow a fixed protocol. More complex situations are possible in the experiment and new procedures have to be developed to incorporate these aspects.

Looking at these aspects, particularly the safety aspects, it also becomes clear that when dealing with personalization inside the car, we are rarely just looking at the user, but we need to take into account the current context as well. Context could for instance be other passengers in the car, the state of different vehicular controls, or the traffic outside. Moreover, personalization takes into account the relevant information at a certain point of time or for a certain interval of time. This combination of user knowledge, context knowledge, and time is also called a *situation*. A lot of work deals with the detection of special types of situations (see e.g. Dötzer, Kosch, & Strassberger, 2005; Krishnaswamy, Loke, Rakotonirainy, Horovitz, & Gaber, 2005), most importantly those that may become dangerous. Common examples are the detection of drowsy driver situations, impending crash situations, or slippery road conditions. The detection of basic maneuvers of the driver (e.g. passing, left/right turn, left/right lane change, starting, stopping; see Oliver & Pentland, 2000) is part of that aspect as well. Understanding a situation and its meaning for the current and future instants is the next step, and is known as *situation awareness*. It describes a set of processes that are performed continuously, but to varying degree, in order to support decision making. In psychology, this concept is mainly attributed to humans. In the car, a situation-aware driver has a better understanding of her own status, her surroundings, and is capable of recognizing threats early and act accordingly. Hence, situation-awareness is relevant for personalization too: First, knowing the driver’s level of situation awareness can help predict difficult situations and act pro-actively. Second, an adaptive system can affect the driver’s situation awareness through suitable means.

The concept of situation awareness can even be taken further by projecting it onto the vehicle itself (or its systems): Since driver assistance systems perform similar operations to an actual driver, the same concepts with regard to the perception and understanding of a situation apply to them as well. Making the vehicle situation-aware involves a high degree of

Role	Implications (examples)
Driver	<ul style="list-style-type: none"> - Has full access to all car functions - Input and output modalities optimized for "hands-free, eyes-free" usage - Distracting and low-priority functions are suppressed - Information that enhances situation awareness is favored
Codriver	<ul style="list-style-type: none"> - Has access to most car functions, except for those intruding the driver's domain - Should be kept up to date with traffic-relevant information in favor of entertainment - Is encouraged to use modalities that favor precision over simplicity and other aspects
Rear seat passenger	<ul style="list-style-type: none"> - Defaults to access only to car functions affecting his/her own seat or the rear area - Comfort is usually the primary concern
Child	<ul style="list-style-type: none"> - Very limited to no access to car functions - Restricted access to infotainment/entertainment content
Chauffeur	<ul style="list-style-type: none"> - Promoted access to additional functions referring to the fare's seat and comfort - More extensive usage of navigation services
Fare	<ul style="list-style-type: none"> - Is offered special comfort services and infotainment features (local information) - Might be offered value-added services, such as ticket booking
Coworker	<ul style="list-style-type: none"> - Might be able to take over certain bussiness messaging task from the driver - Collective document editing scenarios enabled
Driver of a rented car	<ul style="list-style-type: none"> - Additional help offered for locating controls

Table 6.1: Common passenger roles and possible implications for personalization.

situation evaluation, coordination, and semantic information sharing, something that modern agent-based systems strive for. In the end however, the full potential of this information gain can only come into effect when it can be synchronized with that of the driver in order to complement her shortcomings and predict when intervention will be necessary. Reaching this point can be considered a goal in the long run of personalization technology in the vehicle.

Another characteristic of the in-car domain that is relevant for personalization are the roles that different passengers hold. The co-driver for instance plays a special role among passengers, since he is usually considered a “back-up” for the driver, following the traffic situation and being able to advise, warn, or even interfere to some extent if the situation demands it. Table 6.1 summarizes several roles, of which some are related to particular seat or location inside the vehicle, others to the identity or status of the user, and even others to a combination of both (for example that of the passenger in a cab). Speaker Classification can help to detect several of these role concepts, such as the child on the back seat. Roles affect various aspects of the interaction: For mostly safety-related but also technical reasons, not all input and output modalities are available for all roles. An adaptive function might decide to present the same content differently for the driver than for a backseat passenger. Roles also have a security aspect: The driver obviously has access to more functions than the other passengers. Furthermore, in a chauffeur scenario (e.g. cab ride), the passenger would possibly have even less or differently mannered access than a backseat family member on a holiday trip.

6.2.3 Speech in the Vehicle Context

Since we are particularly interested in using speech as the source of personalization, we also need to look at those characteristics of the domain that affect this modality in particular.

Instrumentation

Working with audio data requires sensors in the form of recording devices to be installed. The vehicle has some special characteristics with respect to the instrumentation with such devices. Because it is clear in advance where the speakers are going to sit, one microphone can be installed for each seat and the position and orientation can be chosen to fit the head's location properly for most people. It is also possible to use directional microphones since the head will usually move only within a small range, and the benefit is an improved separation of voices from different speakers.

The former point is quite relevant for multi-speaker classification in general: The number of microphones does not affect the classification accuracy for a single voice in any significant way, but with fewer microphones than seats, it becomes more difficult to assign the result to different speakers (and even more to locate those speakers physically, if that is desired). If four people are recorded using the same microphone, the system first needs to do a segmentation to determine which part of the speech belongs to a different person. It could do so by detecting pauses (short silence periods) and processing all contiguous speech frames as potentially different speakers, thereby losing the advantage of time fusion (see Section 2.5.6). There are also techniques that can be applied to distinguish speakers by their position using fewer microphones than speakers, such as beamforming. In spite of the general success of these methods in open spaces, the car environment does not allow a clear line-of-sight to most passengers from a single location, e.g. due to occlusion of the back seat by front seat passengers, which can inhibit these methods (Hu, Cheng, & Liu, 2006).

If one microphone is used per speaker and its position is carefully chosen, it is a rather reliable process to find out which person is speaking at any instant, even though some algorithmic effort is still required because of the simultaneous recording of the same voice with multiple microphones, which cannot be fully avoided even with highly directional microphones. The most straightforward option is to compare the volume of speech in each microphone, and only take the input from the microphone with the highest level, assuming the speaker is the one to which the microphone is assigned. This works as long as only one person is speaking at a time. For overlapping speech, a possible option is to use a mutual canceling technique that cancels those parts of the signal that occur with the highest level of energy in one device in all other devices.

When do passengers speak?

When speech is used as an (explicit) input modality, the information becomes naturally available whenever users decide to give input. This is different if speech is used as a non-

intrusive input source. In this case, there is no guarantee that speech will be available when needed. Whether the strategy works at all depends on the scenario. In the telephone scenario for example, speech can be provoked by asking the caller questions in an automated prompt.

In the car context, it is generally to be expected that speed is absent. However, there are several situations where speech can be involved:

In-car Conversations. If more than one person sits in a car, we can almost be certain that some conversation between them is going to take place. It is hard to predict any earliest point when this occurs or a minimum amount of speech per speaker and per kilometer, since the variance is considerable, but it is fair to assume that speech will be available rather soon during the trip when multiple persons are present.

Voice Commands. Modern car HMI provides the option to control car functions with voice commands or with multimodal interaction (see e.g. [Castronovo, Mahr, Pentcheva, & Müller, 2010](#)). The drawback is that the commands are sometimes short, which may result in a somewhat lower recognition performance. Considering that the result improves when more commands are spoken, this is still one of the best options for picking up speech, also because it does not depend on other passengers to be present.

Remote Voice Communication. Although not possible in all traffic situations, phone calls while the vehicle is standing are possible and can deliver a large amount of voice data when they are made. However, their likelihood is generally relatively low. Writing emails and sending instant messages by free-text dictation belongs into this category as well.

Reflection. People sometimes “think aloud”, which may not always be semantically useful data, but is perfectly sufficient for speech-based classification.

Outside conversations. Sometimes, a driver stops and lowers the window to ask for instructions or talks to the neighboring vehicle’s passenger on the parking. This might be an opportunity to record speech, but it is less optimal, since the speech is not directed to the car interior, and may also be mixed with the voices of non-passengers.

Background Noise

Another characteristic of the car environment is the background noise that is produced by the machinery in it, most notably the engine, the aerodynamics, and the sound of the wheels on the ground. In general, there are many ways of dealing with noise in digital signals, e.g. frequency (low-pass, high-pass, band-pass) filtering, adaptive noise cancellation, channel compensation, etc.

It is advantageous that the main type of noise is very constant, at least for a given velocity, so it is possible to dynamically adapt to it and consequently suppress it. For the engine, it

would even be possible to train a profile in advance for each car model, with electric cars obviously requiring the least aggressive approach. Finally, the topic can also be approached physically by suppressing vibration, shielding the microphone, or using highly directed microphones.

Audio Context

Speech is not the only thing that can be discovered in the audio signal. Other events that are relevant to the description of the user's current status or refer to occurrences in the environment are often audible as well, many of which are closely related to Speaker Classification and are detected using very similar techniques. They manifest themselves in the nonverbal part of the audio signal, which is otherwise mostly considered a nuisance factor. Examples include non-verbal utterances of passengers such as laughing, coughing, sneezing, whistling, crying, clapping, knocking, vehicle sounds like honking, sirens, door slamming, radio music, as well as environmental noise from rain, wind, and loose gravel. Many of these have already been successfully recognized in empirical studies (Müller, Biel, Kim, & Rosario, 2008). Some of them are difficult to determine using conventional hardware-based sensors, and for others, audio represents an economic alternative. Even if there is already a hardware sensor, e.g. for rain, it could have a delay, be inaccurate, or defective, so the added redundancy will improve the reliability of the system. The hardware requirements for the detection of non-speech events are even lower than for speech: Only a single microphone is needed, which is inexpensive and already present in many cars.

Parallel and Off-board processing

Like any mobile scenarios, hardware in vehicles is subject to strict resource limitations. There are several ways to cope with this. One option is to optimize the software for this environment. How this can be done has been described in Section 5.4 for embedded classification modules. Here, platform-specific modules are created that do e.g. use different cache sizes or performance trade-offs depending on the platform. Another common example is GPGPU. Since the importance of visually appealing graphical displays in cars is increasing quicker than that of other tasks, powerful graphics boards are given preference over good CPUs by many car manufacturers. As the graphics board essentially also contains one or multiple processor cores, the concept of GPGPU was introduced to utilize their power as backup processors for other operations. The major overhead of this is handled by frameworks, but because the GPUs are parallel in nature, it additionally requires a rewriting of the algorithms. In speech technologies, this is not easily possible for all processes.

If an adjustment of the software is not sufficient to resolve the issues, or if the penalty in the trade-off is too high, there is another possible option, which is to move the processing out of the vehicle and onto a back-end machine located in a computing center. This is also called off-board processing and requires a connection for the transmission of data. Whether this connection needs to have a high or low bandwidth, delay, and availability depends on the task.

Off-board processing is especially attractive for tasks that have high computational intensity, but a low data volume. With speech data, one might initially think of high bandwidth audio data that is more difficult to transfer, but actually, there are methods for compressing speech data which work rather efficiently. In addition, for speech processing including speech recognition and Speaker Classification, low-fidelity sound (e.g. 8 kHz sampling rate mono) can give good results too. Another improvement would be to extract features in the front-end and only do the classification on the back-end side, since the feature data is usually smaller and thus easier to transfer than the source signal. This is precisely the idea of a voiceprint (see Section 6.2.6). On the other hand, in the future, broadband connections such as UMTS will increasingly be available in vehicles, so that there is even more incentive to move to off-board processing. Still, a constant connection cannot be expected, so a dependency on it can turn out to be more problematic, for example when the result of a speech recognition request is needed. In case of Speaker Classification or speaker recognition, a delay in processing is much more passable.

6.2.4 Major Related Projects

The dissertation at hand was created in the context of two large and some smaller automotive-related projects. There is a bilateral influence that can be observed between both entities: As part of the thesis, certain theoretic questions in the automotive field were raised, which were going to be answered in a series of studies, experiments, and practical solutions. The projects defined related research questions in need of a practical answer, so that the thesis results immediately influenced the project outcome. On the other hand, projects presented a formal frame for this research and, through numerous other contributors, also offered new ideas, views, and connections. This section introduces the two larger projects, CARMINA and SIM^{TD}, and highlights the aspects which are most relevant to personalization.

CARMINA

CARMINA is an acronym for **CAR**-oriented **Multimodal IN**terface **AR**chitecture. In short, the project investigates how vehicle-related interaction can be improved on three levels: interaction between passengers, interaction between passenger and vehicle, and interaction between passenger and the outside environment (possibly via the vehicle) or mobile Internet services (see Figure 6.9). Multimodality is a key technique in ensuring that this multi-layered interaction will be safe and intuitive with respect to the special requirements and restrictions of the automotive domain. Its particular challenge in this project is to mitigate the effect of the increasing count of devices, functions, and controls that are introduced in new vehicles, and that have been shown to interfere with intuitive and efficient usage due to their great number. Hence, CARMINA deals with the design of dialog systems, which involves the development and evaluation of new interface concepts. Dialog systems are traditionally knowledge-intensive, and require a basic understanding of the environment (here: the vehicle) and the actors involved. One of the research questions in this context deals with adaptation of the system

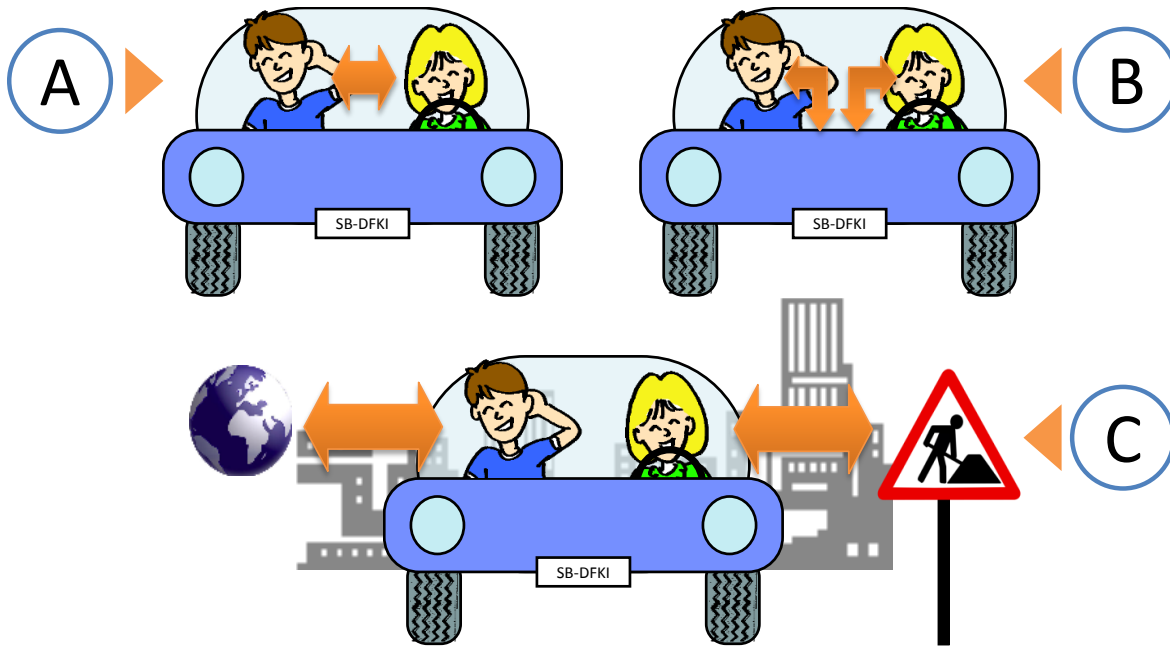


Figure 6.9: CARMINA investigates interaction between passengers (A), interaction between passenger and vehicle (B), and interaction between passenger and the environment (C). (Source: project proposal)

behavior to the rapidly changing physical and spatial environment, both of them being information categories of the in-car domain. An example would be a Twitter application which suggests possible topics the user might post, taking into account the current situation, including user state and environment. A second question attempts to generalize multimodality in terms of multi-party interaction, which plays a role for various forms of collaboration among passengers and multi-party dialogs. Then, issues related to multimodal fission need to be considered with respect to the output modes available in cars, including different “zones” (e.g. front and rear) and actuators (screens, LEDs, audio). Finally, role modalities are investigated. This covers differences between passengers (e.g. driver vs. co-driver), but also further concepts associated with a role, such as that of parents vs. children or chauffeur vs. fare.

In the light of these research questions, Speaker Classification and personalization are two more key aspects of CARMINA: spoken and multimodal input, change of context, multi-party-interaction, and user roles – all of these topics have strong user-related aspects that inevitably lead to questions that are covered in this work, such as: What is known about the person using the system? What are ways to acquire new information? How can we maintain, combine, and extend the knowledge? And finally: What is the appropriate way to make use of the facts in order to improve the system through adaptation?

As depicted in Figure 6.10, CARMINA also delivers a technology platform: On the software side, the four most visible components are the Multimodal Dialog Manager, Knowledge

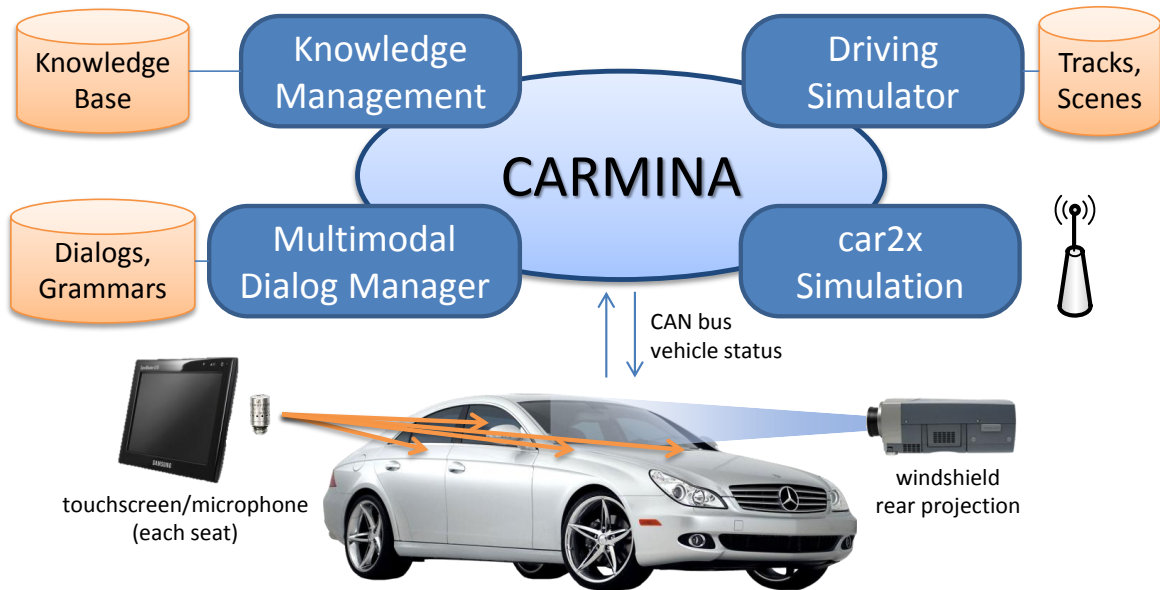


Figure 6.10: Core components and technology architecture of CARMINA.

Management (including adaptation), Driving Simulator, and Car-to-X Simulator. The ODP⁵-based (Pfleger & Schehl, 2006) dialog manager can work with speech, touch, eye gaze, and other input modalities, including multimodal combinations. A special type of input modality that is still under active development are electro-statically detectable gestures (Endres et al., 2011). The ASR component uses complex grammars created for the automotive domain. The output is sent to the screens or read to the user by a TTS voice. The knowledge management component is used by many applications in CARMINA as a way to store and obtain knowledge about the user and the situation. It is what this chapter will focus on for the largest part. On the hardware level, a number of scenarios require access to the car's on-board electronics, such as gas and break pedals, steering wheel, electric windows, air conditioning etc., through the vehicle's CAN bus, which has been connected to the CARMINA system. To communicate with other vehicles, car-to-car communication devices are necessary. Using a simulation component, certain aspects of this communication, such as conspicuity enhancement (Castronovo, Endres, Del Fabro, Schnabel, & Müller, 2011), can be demonstrated and investigated. Important metrics like driver distraction and driving performance can only be measured in a driving simulator set-up, therefore a simulator environment with front windshield projection was integrated as well. It can be used both for simple and strict test courses, as well as for more realistic and visually appealing city drives in large 3D worlds. A more fine-grained view of the CARMINA modules separated into input, output, applications, and support functions, is shown in Figure 6.11.

CARMINA is a three-year government-funded research project that was started in 2009. It is fair to state that CARMINA was the most influential source for the automotive-domain

⁵ *Ontology-based Dialogue Platform*

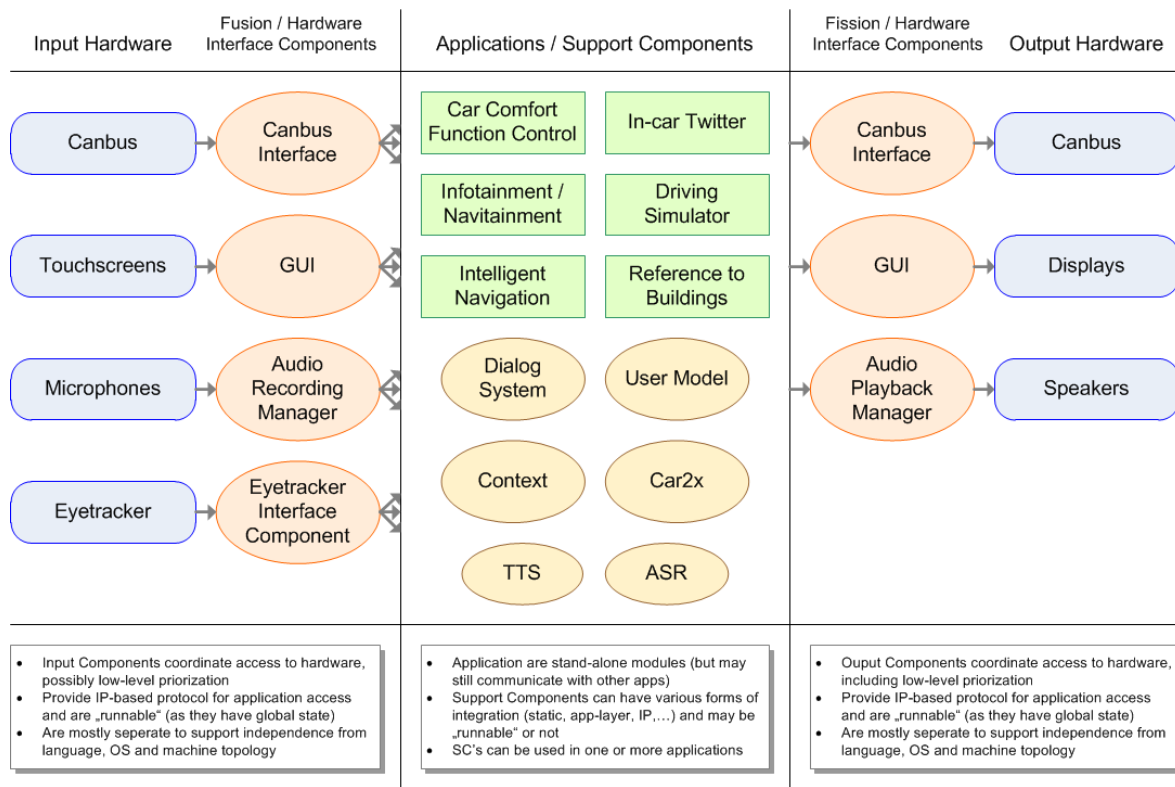


Figure 6.11: Individual CARMINA modules fit into four categories: input modules, output modules, application modules, and support functions.

aspects of this thesis. It seeks to contribute to the questions raised as part of the project, more precisely the personalization aspects and applications, but it also extends beyond in many areas. In addition, the knowledge management component developed in this thesis (KAPCOM) is also a core part of CARMINA and is used in many of its showcase scenarios.

SIM-TD

The second most relevant project is SIM^{TD}, which stands for “Sichere Intelligente Mobilität – Testfeld Deutschland” (Safe Intelligent Mobility – Test Field Germany)⁶. This government-funded four-years project attempts to improve safety, efficiency, and comfort of road traffic – individual drivers and the transport network as a whole – by building on existing yet in the automotive context still rather novel technologies, such as WLAN, 2G/3G networks, and car-to-X communication⁷. Overall, SIM^{TD} is the larger project of the two, since it also involves many partners from the German automotive industry: from car manufacturers over suppliers to public bodies. A number of functions (ADAS as well as IVIS, including value-added) are selected and analyzed in the first phase of the project. Some of these functions are:

⁶<http://www.simtd.de/>

⁷Source: SIM^{TD} project proposal

- Road preview
- Construction Site Information
- Dynamic Route Planning
- Detour Recommendation
- Obstacle Warning
- Traffic Jam Warning
- Road Weather Warning
- Emergency Vehicle Warning
- Traffic Sign Assistant
- Traffic Light Assistant
- Junction Assistant
- Internet-based Traffic Information Transmission
- Location Information Services

These functions are developed by different parties, who have been creating and testing prototypes already prior to the project. However, their integration into a single complete system is a new aspect in SIM^{TD} . This integration takes place in the second phase. The SIM^{TD} system consists roughly of the components outlined in Figure 6.12. A single vehicle of the fleet is equipped with a Vehicle Application Unit (AU), which runs the function logic, human-machine interface (HMI), and auxiliary services (e.g. logging) in *OSGi Framework* bundles, connected to a Communication and Control Unit (CCU) device, which takes care of all external communication. There are three types of communication: with other vehicles (car-to-car) using the low-latency 802.11p protocol, with road infrastructure such as traffic lights and electronic signs using the same protocol, and with the experiment coordination center. This center can assign special tasks to each driver as part of experimental series, collects sensor data and driver feedback, and evaluates the results. It exchanges data with the official traffic management of the involved state (Hesse) and city (Frankfurt), which in turn programs the road infrastructure and uses it to push information (e.g. weather or obstacle warnings) to vehicles. Obviously, a core connection point is the HMI component, which represents the interface between the user and the system. Various work lines deal with different aspects of this interface: from merging presentation requests of different functions and choosing the right time to display information, over interaction concepts with different hardware, to general design and usability rules.

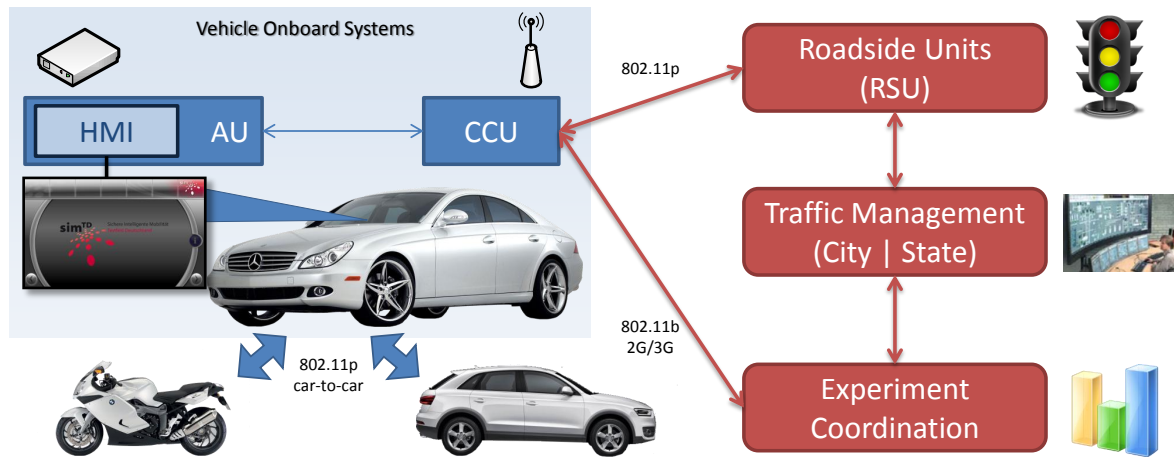


Figure 6.12: The SIM^{TD} Architecture. One part of the instrumentation takes place in the vehicle, while the other part affects the road infrastructure and central management centers.

In the last project phase, the developed system will be evaluated in a large-scale field study: In a test area around Frankfurt, a fleet of 120 selected vehicles (cars and motorcycles) will be equipped with the SIM^{TD} hardware. They will be driving on a carefully designed and balanced set of roads, occasionally receiving tasks and events from a fleet management center and report feedback and sensor data. This procedure allows a thorough testing of all components, their interaction, and quantification of their impact on driving under realistic conditions.

Other than in CARMINA, there are only very few personalization aspects in SIM^{TD} by design, so there was no immediate application of concepts from this work in the project. However, that was not necessary, since it serves as an excellent opportunity to demonstrate the benefits of personalization: The work areas cover a large assortment of driver assistance functions, which are elaborate, functional, and highly beneficial already in their current state. Adding to this version further intelligence and personalized behavior, it is possible to create an “enhanced” version. As of now, this enhanced version is not yet ready to be evaluated in a test fleet on the road, since even the non-augmented functions represent a leap forward in technology that requires thorough testing. Nevertheless, it generates a valuable carryover for this work that could also shift into the focus of a potential successor project. Hence, the SIM^{TD} functions serve both as a starting point for the current state of advanced in-car applications, as well as a showcase for the potential of personalization.

In the following section, an application is presented which demonstrates how the SIM^{TD} functions can be personalized using Speaker Classification techniques.

6.2.5 Adaptation of an In-vehicle Information System

In the previous sections, we have learned that IVIS may have a great potential for improving safety in a vehicle when designed appropriately. When this design accounts for aspects that are specific to the individual, the benefit is estimated to be largest. Therefore, it is reasonable to use a non-adaptive IVIS as a basis for the demonstration of personalization concepts. In this section, we are going to merge both major techniques previously introduced – Speaker Classification and personalization through user adaptivity – into a single showcase system based on functions from the SIM^{TD} system (see Section 6.2.4). A live working system is going to be presented in conjunction with a driving simulation as proof of concept and demonstration of the ideas.

Goal

The SIM^{TD} system is designed following closely the industry norms and code of practice, which give recommendations with respect to many design aspects. Several aspects have been further evaluated in user studies in the project, such as which type of progress bar might be appropriate. Nevertheless, the current version of the system remains a “one-size-fits-all” solution: it does not take into account the preferences or requirements of particular user groups, such as men/women and young/elderly people. Even if the standards have been defined in a way that *most* people do not run into problems, there is still room to *optimize* the behavior for many users. Hence, we want to use the information provided by a Speaker Classification system to make the system adaptive.

Three functions from the system have been prototypically selected for this application. The first is the obstacle warning function, which is a type of local danger warning. This ADAS function receives messages from the traffic management center about different types of obstacles the driver might encounter on the road, which are a broken-down vehicle, a construction site or moving construction site, lost cargo, pedestrians or animals wandering on the motorway, accidents, emergency vehicles, or general dangers not further specified. If the danger is deemed relevant (i.e. in the vicinity of the driver, on the same lane, etc.), then a warning is generated and displayed when the driver is nearing the danger site. The warning remains on the screen until the site is reached, with a counter showing the distance to arrival.

The second function is called traffic light phase assistant. Its purpose is to give the driver more insight into the timings of traffic lights on his way so that she can optimize her maneuvering behavior. The function is activated whenever the car approaches a traffic light. It receives the data directly from the roadside unit, which is equipped with a special wireless connection. There are two types of information that can be displayed for each individual traffic light: If the light is currently red, the function can show the wait time that is remaining until it becomes green again.⁸ If the traffic light is green, and there are other traffic lights further down the road (as is common within city areas), the function can also opt to show

⁸In some other countries, this functionality is often already provided next to the traffic light display.

the recommended speed that is estimated to allow the driver to catch a “green wave” under normal conditions, i.e. to pass the subsequent traffic light(s) as well without stopping if she retains the speed. The implementation that was used for this application further announces the warning message through a message spoken via an integrated female TTS voice.

The third function is a parking display, which – contrary to the first two functions – has to be activated manually by user from the main screen. This function basically extends the navigation screen by adding the locations of parkings on the birds-eye map view. It further pauses the scrolling of the map to allow the user its manipulation, such as panning and zooming. Clicking a parking displays additional information, such as opening hours and free lots.

The implemented system makes use of the user properties *age class* and *gender* to adapt the presentation of these functions. We have created five adaptation rules which exploit these properties. Three of them are meant to make the system more accessible and driving safer for elderly users, while the remaining two relate to gender-specific needs and preferences. More precisely, the rules can be specified as follows:

1. It is well-known that with increasing age, the sight usually decreases. For elderly drivers above a certain age, i.e. that are classified into the *Senior* class, the size of the text should be increased to make the text easier to read, possibly at some other cost.
2. In the same way as sight decreases, hearing and the neural processing of auditive information decrease as well. For *Senior* users, the speaking rate of the TTS should therefore be decreased to allow for a better understanding of warning messages.
3. Reaction time often also decreases with age. For *Senior* users, the system should adapt the parameter that controls how early warning messages are presented to do so slightly sooner, if possible, in order to give elderly people additional time to read and understand the message. (There are also indications that young drivers tend to switch off warning systems if they feel disturbed by them too often and too early, so lowering the sensitivity for this age group might have also been reasonable. However, without a real study that compares cost and gain, this rule would be too arbitrary.)
4. Many studies have confirmed that men are more attracted to female voices and vice versa, and they usually configure their on-board navigation systems accordingly when the option is given. Consequently, in order to make the system more attractive (and thereby possibly increase its impact), the voice used by the TTS should be set to a male voice for *Female* users and to a female voice for users classified as *Male*.
5. Many parking garages offer special parking lots for women located in more secure (e.g. observed or guarded) areas. To make it easier for them to discover where such places are available, the parking assistance function should use a different highlighting for parkings that have one or more women parking lots when the user is *Female*.

Note that the selection of adaptation rules was made, as it should be for real systems, with the cost of misclassification in mind. This means that if the user is classified incorrectly, this does not have any serious or irreversible consequences. If the age is misclassified, longer warning times or slower speaking rates might be noticed, but should not result in discomfort when the error is corrected again later, e.g. through supply of more speech material. The same holds for the change of voice and addition of women parkings, although this change is more obvious. As opposed to age, however, gender is recognized correctly with a much higher accuracy.

Architecture

The simulation environment that has been set up to realize the scenario just described consists of several components, which are depicted in Figure 6.13. A real Mercedes R-Class demonstration vehicle serves as the installation base (although this is not a technical prerequisite). The hardware set-up consists of two 7-inch touch screens for the two front seats located in the center console and a rear projection onto the front windshield using a wide angle projector and a special projection surface. In addition, all seats are equipped with hi-fidelity directional microphones (*Sennheiser ME 105-NI*) that are used to capture the corresponding passenger's speech. The microphones are connected over XLR with a *MOTU 8pre* external audio interface board, which in turn is connected to a server PC that runs all the software needed for this scenario. A driving simulation that has been developed as part of CARMINA is controlled using the steering wheel of the car (via CAN bus) and projected onto the windshield (see Figure 6.10 on page 240). A "light" version of the SIM^{TD} application unit executes the basic functions of the system, including the presentation task scheduling and HMI, which is displayed on the driver's console screen. It does however neither connect to a traffic center, nor does it use the "real" functions. Instead, it receives the presentation tasks directly from the driving simulator through a special uplink. When speech is recorded through the microphone, it is forwarded to a separate classification component. Using the FRISC classification approach, the age and gender of the speaker are detected. The result is sent to the KAPCOM knowledge management component (see Section 6.2.8). When the simulator generates a new presentation, it retrieves the current user information stored in the knowledge base and decides which adaptation should be performed.

Implementation

In order to implement the adaptation, a few adjustments had to be made to the default SIM^{TD} HMI, which is described in the following. In SIM^{TD} , in order to trigger a display or TTS message, a *presentation task* is created. These tasks coordinate the occupation of the user interface between the different functions, which otherwise do not cooperate. When two presentation tasks request presentation at the same time, a scheduler is used to resolve the conflict. It uses a priority system in conjunction with usability rules to decide which function is allowed to output its message at what time. This also allows a more long-term planning

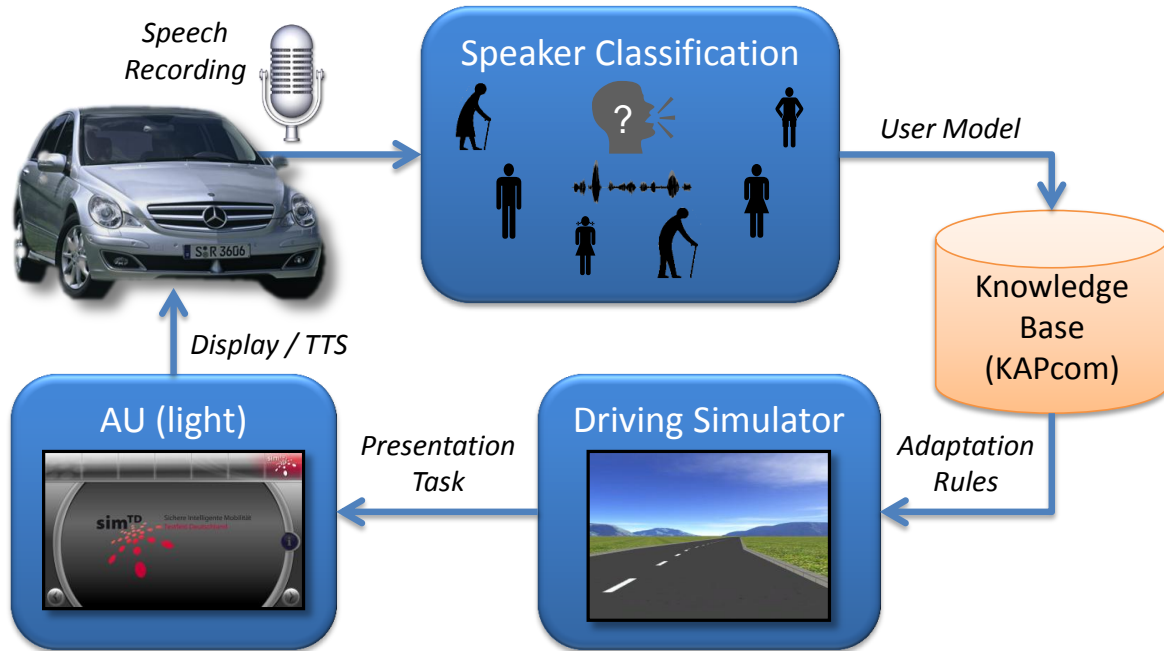


Figure 6.13: Architecture of a system running the adaptive SIM^{TD} functions.

of tasks. The default GUI (see Figure 6.13) has three separate output *channels*, which are the main area, which can display one function at a time, and the smaller icon area above, which can hold up to seven presentation tasks in condensed form. The third channel is the TTS. If a conflict cannot be resolved through temporal rescheduling, then a different display strategy can be chosen. For example, the more important task is shown in the main area, while the less critical function is sent into the icon area (from which the user can bring it onto the main screen by selecting it). In addition to the automatically appearing presentation tasks, the user can switch between a few primary screens by touching the arrow buttons in the lower region. One of them is the navigation area (see Figure 6.14 C), which contains all important functions of a navigation system, such as a map that can be panned and zoomed. By hitting buttons on the left, additional icons such as for parkings, webcams, and events can be enabled on the map.

A presentation task is defined in an XML template. Each function has its own set of template parameters. For example, the local danger warning functions can specify which type of danger the user can expect and in what distance. Just before the task is displayed, the XML template is converted to a format that matches the output channel, which is a UI representation implemented in Flash for display channels or a command sent to the VoCon synthesizer in case of TTS messages.

In order to support the HMI adaptation, several parameters were added to the presentation task templates:

- a parameter for switching between the *normal* and a *decreased sight* layout (which for

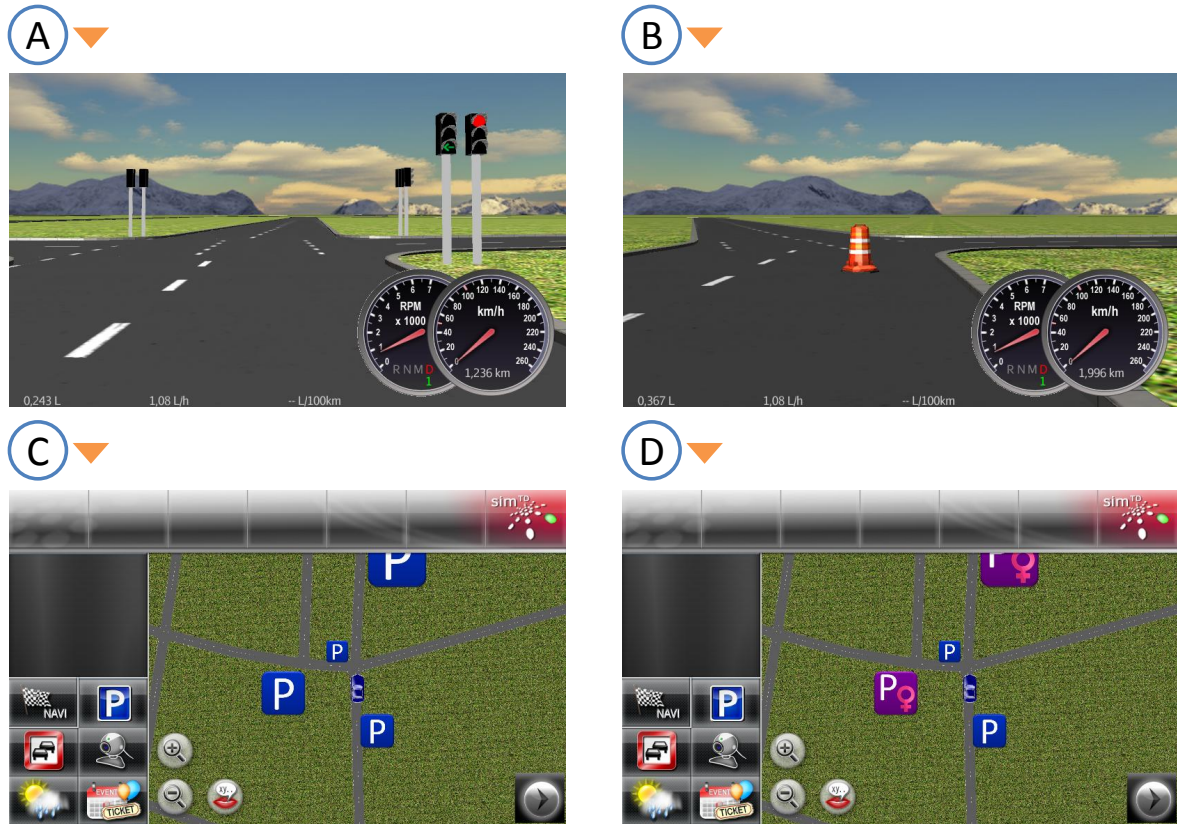


Figure 6.14: Simulator and adapted parking assistance functions. (A) shows a traffic light situation that also triggers a traffic light phase assistance presentation as in Figure 6.15 B. (B) An obstacle for which an obstacle warning as in Figure 6.15 C is generated by manually setting a trigger. (C) Non-adapted parking assistance screen based on the situation in A (parkings are not visible in the 3D view). (D) Adapted parking assistance showing a different icon for parkings with women parking lots.

instance could also be activated if a short-sightedness was explicitly known),

- a parameter for setting the speed of the TTS,
- and a parameter that determines which voice the TTS uses.

The TTS-related parameters are forwarded to the speech synthesizer, where they will cause the corresponding effect. For the graphical UI, new variants of the obstacle warning and traffic light phase assistance functions were created as shown in Figure 6.15. As can be seen, the main difference in both cases is the font size, which has been increased by approximately 50%. This obviously had some effects on the layout as well, which has been adjusted correspondingly. Under normal circumstances, the non-adapted versions remain to be preferred, as they can contain longer text (which had to be shortened in some cases on the adapted layout) and

have a cleaner, less “bloated” appearance. Nevertheless, in these two cases, the trade-off for the gain in readability is relatively low – on a screen with more textual contents, the effect could be more severe.

In the absence of the “full” SIM^{TD} system, the presentation tasks are generated by the driving simulator. The traffic light phase assistance tasks are generated automatically when the user approaches a traffic light situation built into the course (e.g. as in Figure 6.14 A). Obstacle warnings on the other hand are realized using manual triggers, which are also part of the track. Since the simulator generates the task, it is also the component that specifies the adaptation rules. The information it needs to perform the adaptation, i.e. the very basic user model consisting of age class and gender, is available in the knowledge base. For simplicity, the adaptation is performed directly as part of the simulator (instead of using a built-in adaptation mechanism that would have also been available in KAPCOM). Most of the rules are rather simple and consist only of filling in the presentation task templates accordingly. For example, if the user model has set the age class to *Senior*, the *AdaptForDecreasedSight* template parameter will be set to 1 and the *TTSSpeed* parameter is set to 75 instead of 100. In case of the reaction time, no new parameter had to be created, as the preferred timings are already part of the default presentation model. Normally, when a presentation task is created for an obstacle located at a certain position (see Figure 6.14 B), the estimated time of arrival at the location is computed based on the current driving speed and used as end time for the task. The start time can then be set to a fixed amount of time prior to that end time, based on the contents of the message, which is then the presentation duration. It can also be updated while driving to accommodate changes in speed. The adapted behavior modifies this duration by changing the presentation start time parameter. Instead of changing the absolute value, however, it applies a linear function, so that the duration slowly increases with age. The advantage of this method is that a finer grained age classification, or even an exact specification of the age, results in a smoother behavior, while we use the age class mean (or another representative value) otherwise.

Finally, in order to achieve the adapted parking space information screen, a web server was created, which replaces the web server that normally supplies the map directly from the navigation system in SIM^{TD} . This implementation is instead synchronized with the simulator, from which it retrieves a top-down view map (see Figure 6.14 C), and also connected to the knowledge base. Each time a map is requested, the driver’s gender is looked up in the user model. If the driver is a women, instead of the neutral parking map, the adapted version shown in Figure 6.14 D is returned to the HMI.

6.2.6 Multi-Speaker Positioning: Who Sits Where?

Staying with the automotive domain and voice as a knowledge source, there are several possibilities how information can be obtained from it and be made available to diverse applications. This section presents a case study that exploits the additional input that is available when the car interior is equipped with multiple microphones. In the optimal case, there will be



Figure 6.15: Functions adapted to decreased sight / visual impairment. (A) Traffic light phase assistance function with normal font size (note that not all possible slots for traffic lights have been used). (B) The same function with the adapted layout using larger fonts at the cost of clarity and number of possible traffic lights. (C) Non-adapted obstacle warning screen showing the type of obstacle and distance. (D) Adapted obstacle warning featuring larger fonts at the cost of maximum text length (which for this warning is not a problem).

one (or more) per seat, which is not unrealistic, as basic microphones are inexpensive and are useful beyond this showcase. The two pieces of information that we are interested in are the identity of a user, which links to the user profile, and the position (i.e. seat) in the car. In order to obtain the speaker's identity, we do not introduce a completely different technology, but rather show how the concepts developed in the FRISC approach can be re-used for speaker recognition (which is the area for which they were originally proposed). This task also has aspects generally related to the field of indoor positioning, although good instrumentation greatly simplifies this part of the task. Getting both pieces in combination requires some more thought and tools. The whole concept was first introduced in [Feld, Schwartz, and Müller \(2010\)](#).

Goal

Knowing what seat a particular passenger occupies enables a multitude of possibilities for personalization, both on the car's on-board HMI as well as on any other personal devices. A large part of these scenarios is driven by the role-based paradigm that characterizes the automotive context: A driver has a very specific task (driving) and very limited options for interaction, a front-seat passenger is to some extent as well involved in the driving but can also handle more non-critical information, while typically back-seat passengers can dedicate their full attention to entertainment or mobile office.

For example, if the car knows who just entered it as driver, it could turn his mobile phone to silent mode to suppress calls as soon as the engine is started. It would not do so when the same person enters as passenger. Incoming emails could be indicated to the driver using a discreet icon on the screen and be read out when requested. For other passengers, the email would be displayed as text and possibly be announced using highly directed audio. Traffic-related information would be shown to both front seats, but the co-driver's map might include additional personalized points of interest. Only backseat passengers with possibly larger screens would by default be suggested movies based on their taste or even offered to resume the movie they just started to watch earlier on that day. Infotainment related to landmarks that the car passes could even be matched to the side a passenger is sitting on, depending on the view.

Roles also limit what a certain passenger can do. Adding to this the knowledge from Speaker Classification, e.g. that a passenger is a child, might alter that behavior in the sense that less control is granted. Of course, since the user profile could be enriched with detailed information on what certain parties are allowed to do, flexible scenarios with special requirements are also possible. An example of such a special scenario would be the cab ride, where the passenger has a clearly defined amount of control, but can also benefit from personalized services.

Other features such as controlling car comfort functions benefit as well from knowing who sits where. As an example, one might consider the temperature setting for a seating zone or the seat position to be matched to the user's preferred default, even if he enters a car that he never went with before – a new car, a rented car, or that of a friend.

The subtitle of this concept, “Who Sits Where?”, is intentionally coined after the motto of on-line speaker diarization, which is “Who Speaks When?” or “Who is Speaking Now?” (Friedland & Vinyals, 2008). The two challenges are closely related: To answer the first question, we have to answer the second one first. On the other hand, the diarization task is simplified, yet not made trivial, by the multiple microphone channels and location meta information. Moreover, the applications are very similar.

Method

Multi-speaker positioning is facilitated by the concurrence of multiple knowledge sources and multiple technologies, speaker recognition probably being the most important one. Fig-

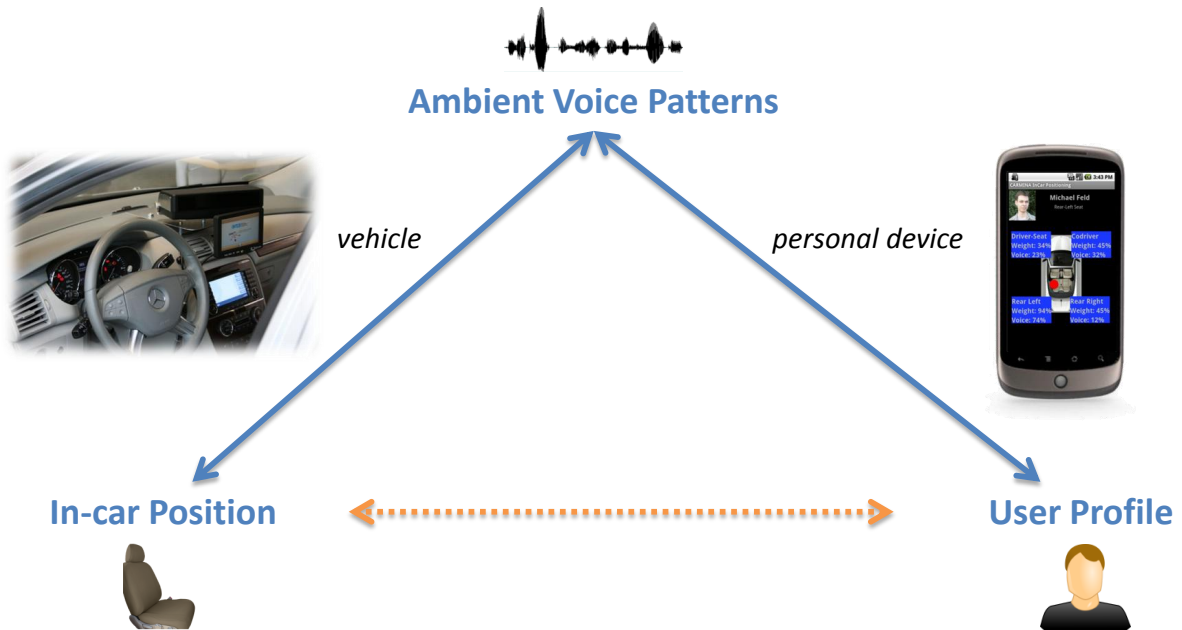


Figure 6.16: The idea behind multi-speaker positioning: Neither the vehicle, nor the personal device can associate the user with a seat, but they can by using speech and joining their knowledge.

Figure 6.16 summarizes the relationships. As depicted, there is a natural distribution of information. For technical reasons, only the vehicle has knowledge about the association between position (seats) and sensors. The sensors in this example are only the microphones, but we could also add other sensors such as weight sensors integrated into the seats and combine the knowledge to reinforce it (see Section 6.2.7). While this allows the car to determine which seats are taken, it does not generally have knowledge about the association between sensor values and user profiles. (It could have so for some users, but being limited to that defeats some of the more attractive use cases.) Such information is however often present in a user profile stored on the user's mobile phone or other personal mobile device. In case of speech, the sensor values would be a biometric identity – a voiceprint (voice profile) – of the user that can be matched against a reference, which is essentially a task of speaker verification. As it becomes clear in Figure 6.16, the voice is the link between user and position: If the voice spoken at a given location X matches that of user Y , it can be concluded that Y sits at position X .

Since not all users want to publish their user profile or even voiceprint (or more delicate information such as weight) right away to other people's cars, the car must pro-actively send sensor data to the device. The device and, in consequence, the user can then decide whether the profile should be shared with the car or not, which might require obtaining consent from the user. When the device is using a knowledge management component such as KAPCOM, it can also define on a finer granularity level which settings should be automatically shared



Figure 6.17: Instrumentation of each seat in the demonstration vehicle with a screen and integrated microphone.

with whom.

From the perspective of positioning systems, the suggested method represents a hybrid approach according to the scheme introduced by [Schwartz, Stahl, Baus, and Wahlster \(2010\)](#). This scheme distinguishes exocentric and egocentric approaches. The *exocentric* approach covers situations where a device carried by the user sends a signal, which is tracked by an accordingly instrumented environment to determine the sender's position. The sensor can then optionally send back the position information to the user. An example is cell-phone positioning based on cell-id. The *egocentric* approach works in the opposite direction in many ways: Here, the user's device only receives data from senders placed in the environment to compute the position by itself, which has clear advantages with respect to privacy, since the users can control if they want to share their position information with a third-party. The reason why the multi-speaker positioning system is a hybrid approach is because sensors – in this case the microphones – are installed in the car and pick up signals generated by the user, e.g. casual conversations or voice commands.

Hardware and Architecture

The actual architecture of the proposed system consists of several hardware and software components that have all been installed into a real demonstration vehicle (Mercedes R-Class) as described in Section 6.2.5. Apart from the 7-inch front row screens, we make use of two additional 10-inch screens installed onto the back of the front row head rest for the two rear seats (see Figure 6.17). A wireless LAN allows nomadic devices to easily join the car network. In our case, several Windows Mobile-based smartphones and PDAs represent the driver's and

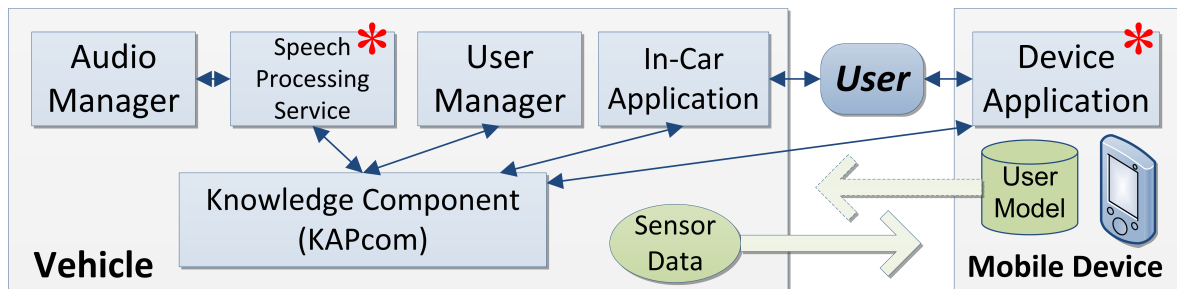


Figure 6.18: Architecture of the multi-speaker positioning application. Components are running either on the vehicle platform or on the mobile device. Components marked with * run parts of the FRISC approach.

passengers' personal devices running the mobile client software.

Figure 6.18 illustrates the software architecture of the positioning scenario. The central component that links together the other models is the KAPCOM knowledge component, which maintains a knowledge base with an automotive-domain ontology. In the positioning scenario, KAPCOM is used as a “blackboard” – a design pattern for problem solving often applied in artificial intelligence (see Figure 6.19 and Corkill, 1991). On the data acquisition side, an *Audio Manager* (see Section 6.2.8) obtains raw audio data from the microphones. It then streams the data to a *Speech Processing Service* running in the same process. This service is responsible for detecting speech and – if speech is present – computing a set of characteristic high-level audio features, the so-called voiceprint. This corresponds to the feature extraction step in the FRISC approach. A more detailed description of this step is provided in the following sections. Every voiceprint is stored in the knowledge base together with the information on which seat it was recorded.

When a mobile phone or PDA connects to the car's network, it automatically starts receiving notifications using a mobile client application when new voiceprints become available. Given the user profile that is stored on the personal device, it can then locally perform the detection step with the sensor data from any seat, i.e. determine in how far a given voiceprint matches the signature stored in the user profile. It may take multiple voiceprints from the “live” system to confirm a match with reasonable confidence. Once a match has been detected, the device owner's location inside the car is known and the client can use this knowledge to perform adaptation on the device, such as changing the phone's ringing scheme and other notifications when the user is the driver. The current demonstration UI on the device (see Figure 6.20 left) provides a visualization of the match likelihood for the evidence coming from each seat. Figure 6.21 shows how the voiceprint matching works.

In a second, optional step, the device owner can decide to share his identity with the car, thereby allowing applications running on its system to also take advantage of personalization. Triggered by a user initiative (see Figure 6.20 right) or a setting on the device, the user information is posted back to the knowledge base. There are multiple possibilities for the

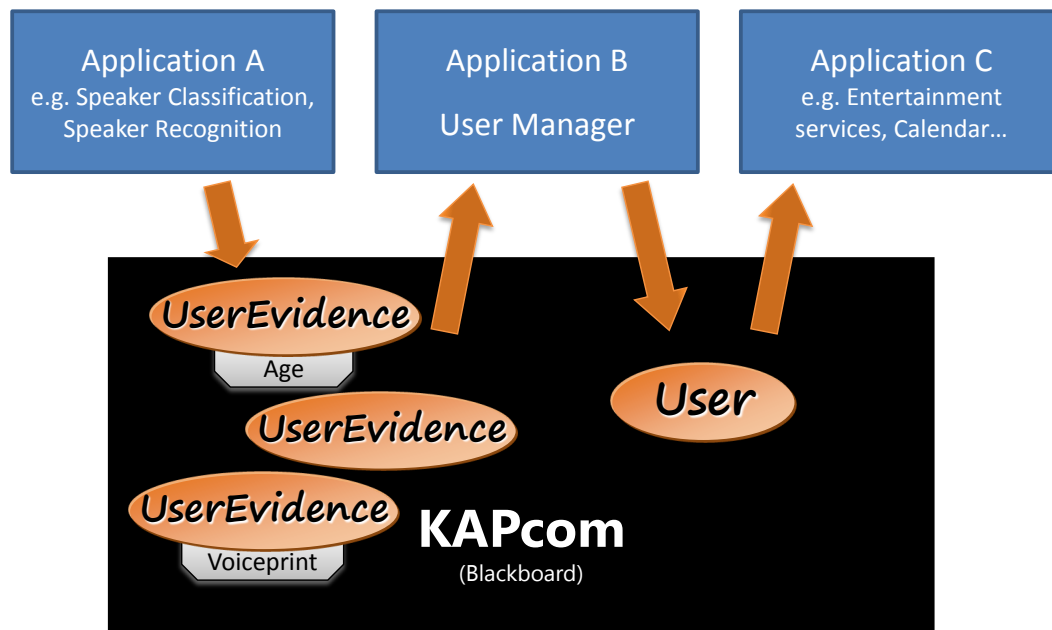


Figure 6.19: In this application, the knowledge base is used like a “blackboard”, where components read and write data and thereby iteratively increase the value.

composition of this information: It may be an authentication token that “unlocks” a user profile already stored in the car, it might be a complete user profile that is exchanged, or it could only be some particularly relevant aspects like age, sight, music preferences, etc. In either case, the device performs the role of linking position to user information. Another component running in the car, the *User Manager*, manages the association of user profiles to seats based on evidence and authentication data. If a device provides a user profile, its task is to merge this data with the existing knowledge and make the resulting profile available to an application rendering the display contents for a specific seat. Such applications should be sensitive to changes in the user knowledge and adapt themselves accordingly.

Speaker Recognition Approach

The speaker recognition method applied in this set-up uses a modified version of the FRISC approach introduced in Chapter 4. The modified system is depicted in Figure 6.22, which shows the overall similarity to Figure 4.3 on page 110, although it is less complex. The following paragraphs explain the changes.

The first change is an architectural one. Instead of having one coherent classification pipeline, the process is split across two contexts: the vehicle and the device. The vehicle performs the pre-processing, segmenting, and feature extraction parts, whereas the device deals with the classification and post-processing aspects. This is necessary to keep up the data separation between user and car for privacy reasons. Another consequence of the separation is that we are dealing with a speaker verification task instead of speaker identification, since each

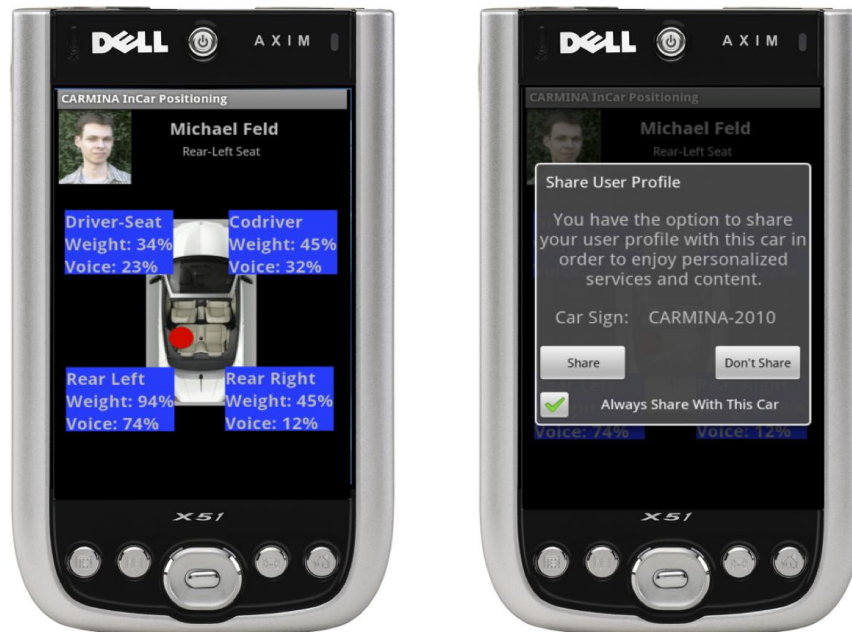


Figure 6.20: Depiction of the UI shown on the mobile device in the demonstration application. *Left:* The system has identified its owner as the passenger on the rear left seat. (Note: Even though the display also shows a weight, no actual weight sensors were connected in this scenario.) *Right:* After positioning, the system asks whether the user wants to share his data with the car.

device matches only with a single user profile independent from all other devices. Technically, this is easily accomplished by setting up the classification as a detection task using the corresponding models, and storing one model per device.

A rather obvious change of this set-up are the classes: Instead of classifying age and gender, we now “classify” the identity. The main difference however is that the SVM is missing in this version, i.e. the GMM-SVM was reduced to a GMM-UBM. The reason for doing so is less related to the classification performance, but rather to runtime considerations: On a resource-limited mobile device, the impact of the additional SVM on the classification time is quite severe, so it was decided to remove it. As a consequence, the GMM now has to adopt the target classes, i.e. we employ speaker-specific GMMs instead of utterance-based models. Furthermore, the final scores used for classification are the log-likelihood ratios of the GMM-UBM, which are obtained by subtracting the log-likelihood of the UBM from that of the speaker model. Since we consider the identity in general to be a structure with more variables than age and gender, the number of mixtures was increased to 1024.

With the GMM now being the target model, we produce one score (LLR) per frame. To generate a decision for a complete utterance, a post-processing step is needed. In our case, this is a simple averaging step performed on the frame results. Majority voting was also tried

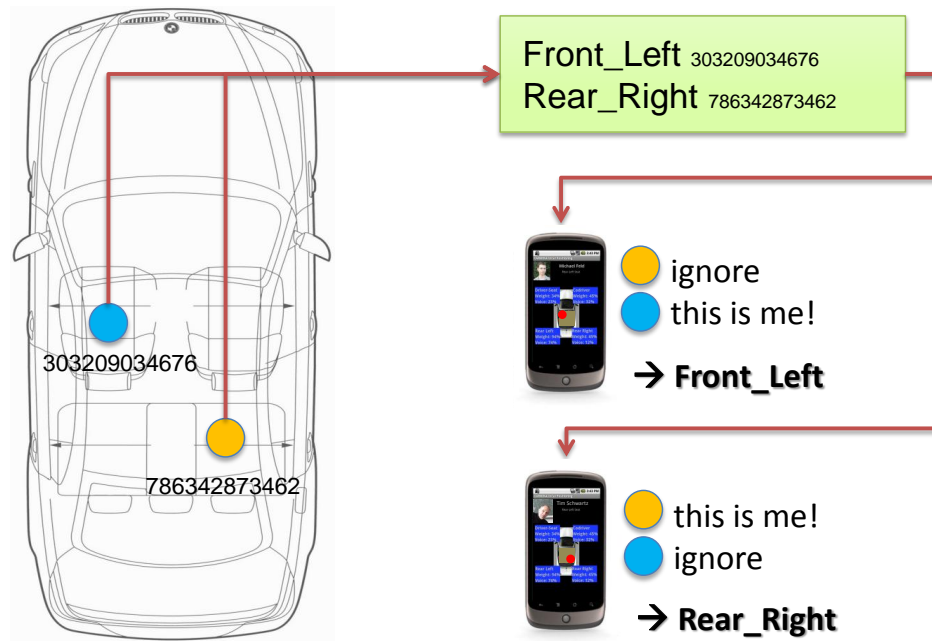


Figure 6.21: Illustration of the voiceprint matching using an example of two passengers. The voiceprint is denoted by a unique sequence of digits. Each mobile device receives the same data.

as an alternative decision method.

The voiceprint representation consists of the MFCC features used in FRISC, i.e. the MFCC coefficients 1–12, except that they are extracted in steps of 10 ms instead of 5 to reduce the amount of data that has to be sent over the WLAN connection. Voiceprints modeled as GMMs can be saved on the device in a compact binary format of approx. 200 KB each. The UBM is also stored on each device.

In order to further reduce sensitivity to noise and silence, for one variant a binary Vector Quantization (VQ) model was trained on the whole corpus in unsupervised fashion. This is different from the type of silence filtering applied in the main experiment series. The idea was that the clustering would pick up the idea of voiced vs. unvoiced frames, which is very well expressed in MFCCs. Essentially, this has proven to be a reasonable assumption, as can be seen in the results when the corresponding frames are filtered out. In addition, the live system features a VAD component that is based on a manually tuned intensity threshold.

Recognition Results on a Sample Corpus

For training and evaluation, a corpus was recorded inside the standing vehicle described earlier. The corpus consists of ten adult speakers (7 male, 3 female) and a total of 76 minutes of material categorized by different conditions (seat, read vs. spontaneous, overlapping, doors closed vs. open). The training set consists of 30 minutes of speech, while the evaluation corpus

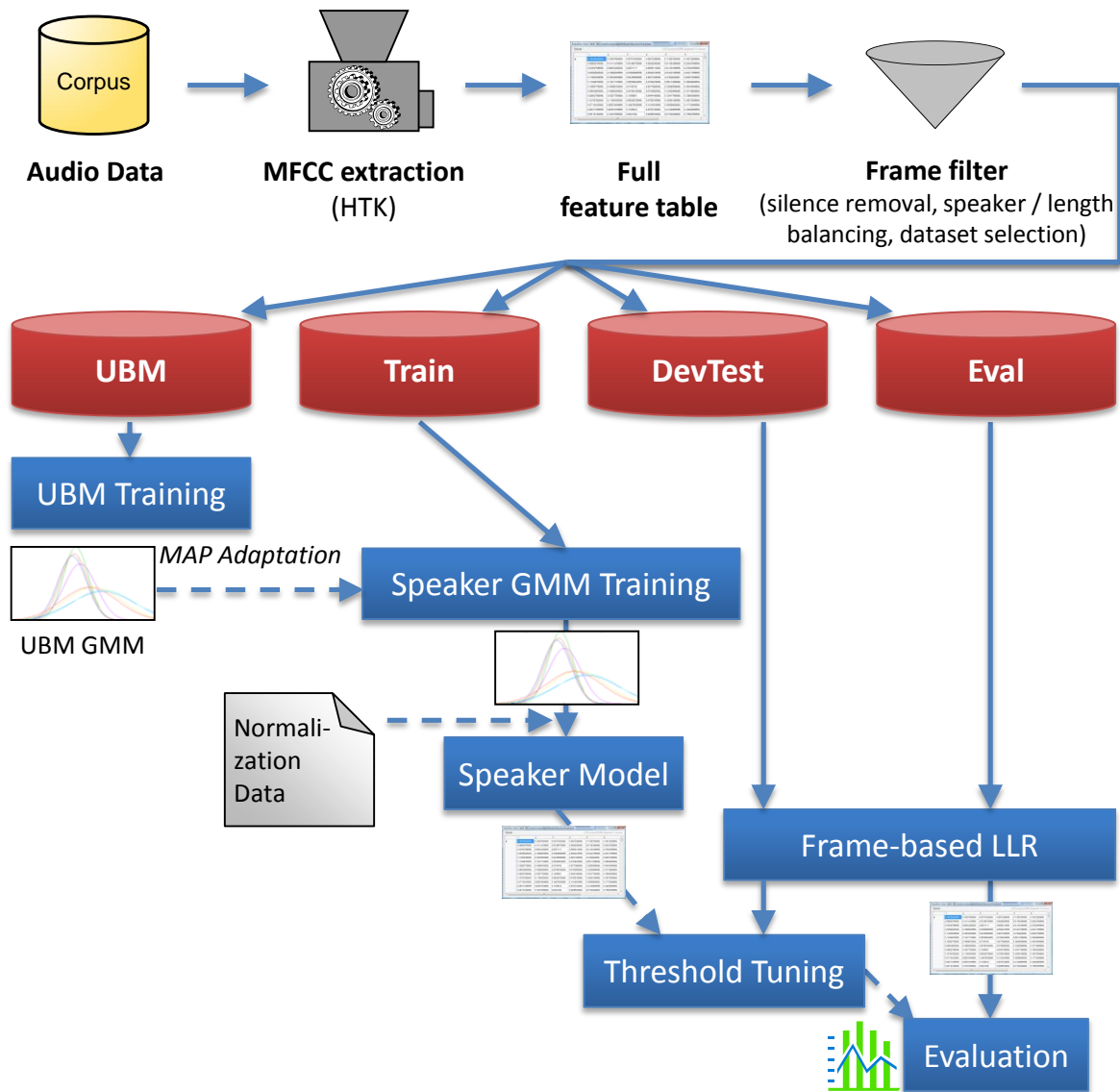


Figure 6.22: Speaker recognition using a simplified version of FRISC (see also Figure 4.3 on page 110).

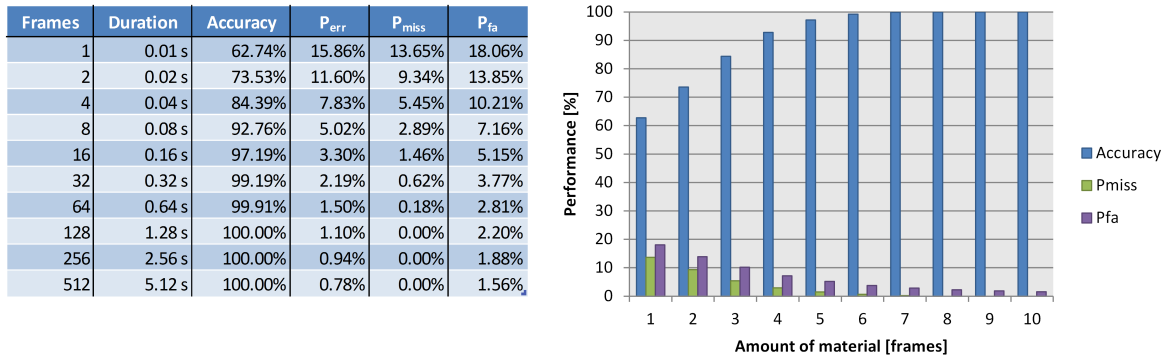


Figure 6.23: Results of the in-car speaker recognition on the evaluation set of a test corpus.

uses 20 seconds (2000 frames) per speaker. There are two types of performance measures applied in this task: accuracy and error rates. The *accuracy* describes what percentage of test samples are assigned to the correct speaker, i.e. it measures how well the system can *identify* the correct speaker out of the full set of (in this case) ten speakers. The second category quantifies the errors a system can make when it tries to *detect* whether a test sample is from a given target speaker, which are *misses* and *false alarms*. A mobile device as in our scenario only has a single user profile stored and hence performs the *detection* task. The false alarm rate is possibly the most critical measure here, because it tells in how many cases the device will incorrectly report a match. Our evaluation therefore takes into account different numbers of frames over which scores were averaged.

Figure 6.23 illustrates the results. In a single-frame scenario (10 ms of speech), already 62.7% of frames are classified correctly at a chance level of 10%. However, it is clearly evident that recognition rates improve significantly when more speech becomes available. At 64 frames, which corresponds to 640ms of pitched voice (roughly a second of ordinary speech), almost 100% accuracy are achieved in this test. The corresponding chance of a false alarm is still 2.8%. It can be lowered to 1.6% when using approx. one fourth of the eval material for averaging.

6.2.7 Fusion of Knowledge Sources

In the examples presented in the previous sections, we have mainly focused on speech as the only knowledge source. Given our knowledge of the performance of Speaker Classification, we have to be aware that there is always some degree of uncertainty involved in the decisions based on this knowledge. Even if the methods are expected to improve over time, the inherent uncertainty prevents it from ever being considered a reliable source of information.

Instead of investing all efforts only into perfecting a method based on a single source, there is also another option, which is to find complementary knowledge sources for pattern recognition. It is very likely that these will suffer from similar problems, if they are based on sensor data. However, it is known since the early days of AI that multiple uncertain sources

can be combined to form a new, more reliable source (see e.g. Schank & Colby, 1973). The reason for this is that with real sensor data, there are always cases where one of the methods does not have suitable features to discriminate the classes, while the same is true for the other sources, but probably for different cases.

In case of age and gender recognition in the vehicle, such a complementary source could be the image data recorded by cameras built into the car, which is likely to be a good indicator of a person's age. A further example is the weight information provided by the weight sensors integrated into the seats. This is obviously not very effective as a sole source of information, since the weight greatly varies between individuals. Yet, if the speech-based classifier yields similar scores for the *Children* and *Senior* classes, which are especially close in terms of voices, the weight data could pivot the result decisively towards one of the classes. Similarly, many in-vehicle ADAS work by using the combination of multiple sensors, such as image and radar. Schiele, Andriluka, Majer, Roth, and Wojek (2009) have shown for instance that combining visual cues with laser data significantly improves precision and recall of people (e.g. pedestrian) recognition.

The benefit of multiple knowledge sources does not only surface under normal conditions, but also when the data is noisy, faulty, or simply missing. For example, as long as the user has not provided any speech, the remaining knowledge sources can be used as a fallback, although the result is expected to be worse than in conjunction with speech. Similarly, when the system detects that multiple people are speaking simultaneously or that there are too many environmental noises, it could decide to suspend or reduce the impact of Speaker Classification in the fusion mechanism and rely only on the other sources. Likewise, when image data is not available because the user has turned her face away from the camera, the influence of speech-based information would increase.

Aside from age and gender, another important property of the driver state to be detected using multiple sources is the attention. Here, we can use high-level imagery data such as the head rotation, more specific pupil information that is often obtained using methods from eye tracking, audio cues and speech-based information including linguistic features, driving performance metrics such as lateral control, brainwave interfaces, and more.

In knowledge fusion, it is also a good idea to take the context into account as a possible source of information: For instance, where someone is driving might offer clues as for the identity. Time of day and duration of driving on the other hand can be part of the fusion that leads to the driver's attention or drowsiness.

Bayesian networks present one method to model the probabilistic relations between different sources of informations. They are particularly suitable for classification tasks because their states are nominal. In short, they represent conditional probabilities that a particular observation is caused by the presence of another observation. Consider the example in Figure 6.24, which models the user state property *stress*: The arrows indicate a direct influence, such as high traffic increasing the probability of stress and stress affecting a person's speech. The conditional probability tables (CPTs) are specified for each node and all possible conditions of predecessor nodes. Using Bayes' theorem (see Section 2.5.5), the probability of the

system variable stress can be computed when some or all of the other node values are known.

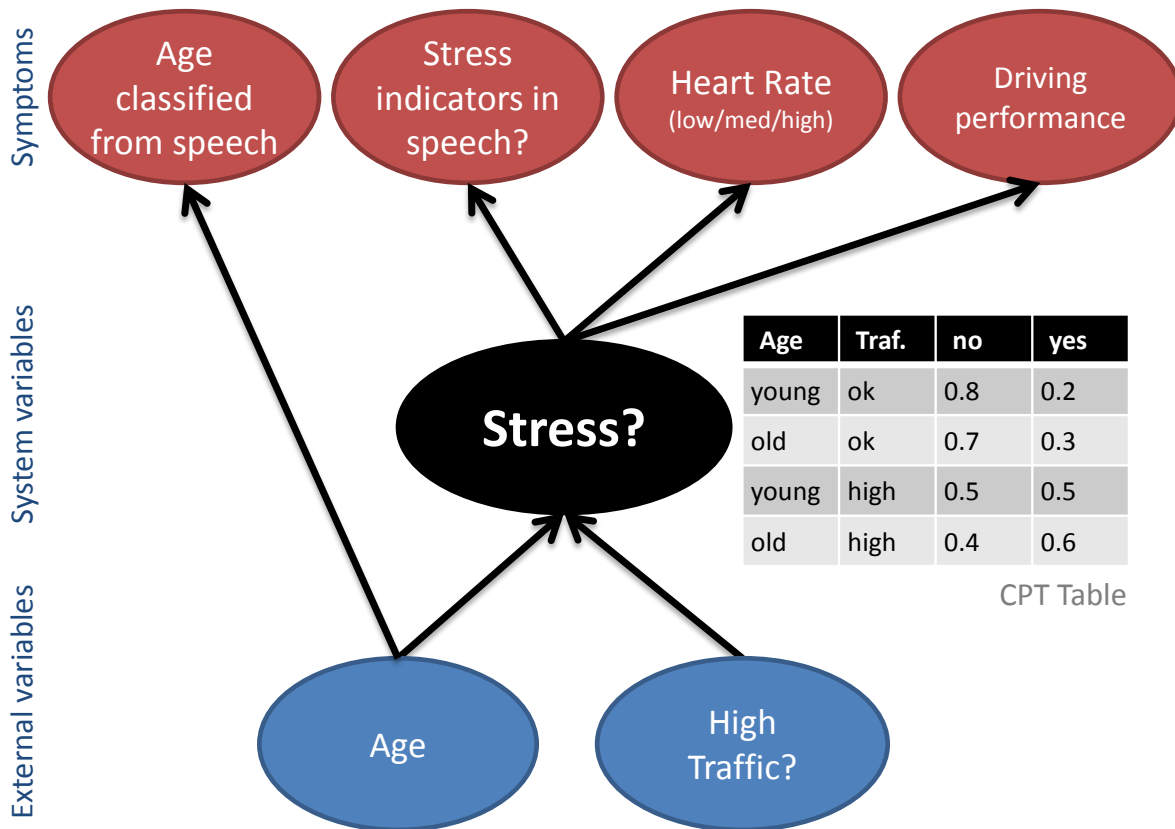


Figure 6.24: Example of a Bayesian network that models stress. In this simple model, stress depends on the age and the traffic conditions. It in turn influences the speech, heart rate, and driving performance. Age also influences the speech. Speaker Classification can help fill in values for two of the nodes. CPT tables for other nodes are not shown.

Another possibility for knowledge fusion is applying fuzzy reasoning. Here, a set of logical rules is used to infer new knowledge from existing knowledge. Each fact is annotated with a confidence, which is propagated along the reasoning process, e.g. by summing up confidences of all preconditions of a rule.

6.2.8 Integration in a Vehicular Personalization Component

As part of this thesis, the technical aspects dealing with personalization in the automotive environment have been implemented in a component called *Knowledge management, Adaptation and Personalization COMponent* (KAPCOM). As the name already indicates, it bundles the three main activities related to personalization as discussed in the previous sections in a single architecture. Having the functionality integrated in a single component has a consider-

able impact on the system's ability to share knowledge between applications and to maximize the transparency of adaptation for the user.

Architecture

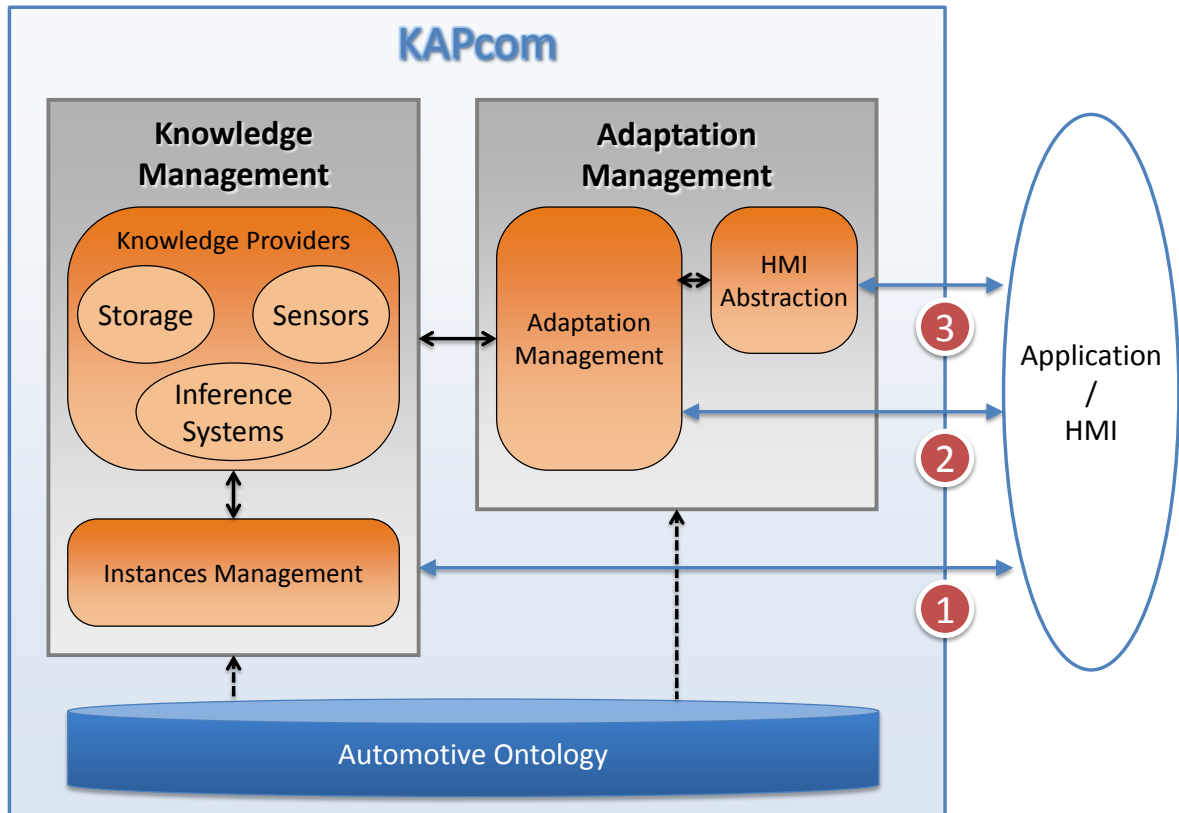


Figure 6.25: General architecture of KAPCOM and the different levels (1 – 3) of personalization support offered.

Figure 6.25 illustrates the various roles and the architecture of KAPCOM. On the knowledge side, it acts as a knowledge management component, which means that applications can store and retrieve knowledge. It also has limited support for inferencing, by using knowledge providers that take care of the inference process. The basic functionality can also be of use for applications that query the component much like a semantically enhanced database. Even more, it can be used for message exchange, a system-wide event mechanism, or as a blackboard infrastructure (see Section 6.2.6).

One of the main strengths of KAPCOM is its ability to use its knowledge for adaptation. This is facilitated on different levels. Depending on the degree of influence that applications like to retain, they can choose different mechanisms to enable adaptation. These three levels are also indicated in Figure 6.25. On the lowest level (1), the adaptation is fully application-defined. The application accesses the ontology to obtain all needed knowledge and by itself

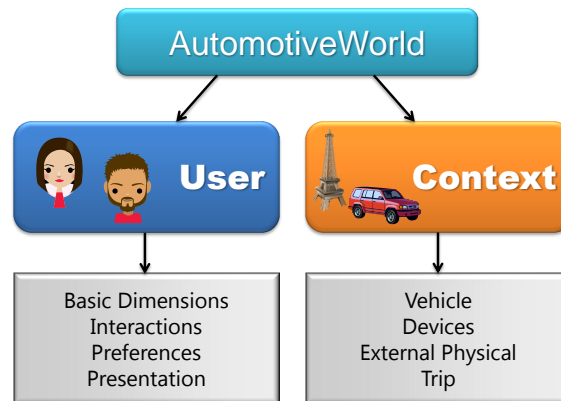


Figure 6.26: Areas of the ontology, which map to first and second level concepts.

comes to a conclusion of what should be adapted, and applies the effect. This approach makes it difficult to ensure a consistent interface and behavior. On an advanced level **(2)**, the application uses the adaptation strategy management functions of KAPCOM. In addition to the generic strategies, it can specify specialized strategies and have the component decide if any and which of them should be applied. Global usability rules, such as making adaptation transparent, are more easily followed using this method. The highest level **(3)** would be to use a generic HMI adaptivity framework (which is currently not part of KAPCOM). Such a framework would merely receive the description of the core UI elements and take care of exchanging adaptation strategies with KAPCOM.

Automotive Ontology

The model that backs knowledge management in KAPCOM is the *Automotive Ontology* (Feld & Müller, 2011; Feld & Endres, 2010). Since information is what most advanced in-car functions are based on, be it sensor data, user profiles, traffic broadcasts, or car-to-car messages from peer vehicles, such a component is of vital importance. More precisely, the key to success in making such an ecosystem work lies in the organized and efficient access to the data: The heterogeneity and distributedness of data sources makes it essential to collect everything in one place, making it available to all functions that depend on it. As the data becomes enriched with structure and semantic meaning, we are also making the transition to knowledge management. By further specifying meta information such as time, confidence, and privacy, functions can employ their own reasoning and knowledge can be shared across the boundaries of a vehicle without being dependent on a strict low-level protocol or a particular manufacturer.

The basic layout of the ontology is shown in Figure 6.26. The root of the ontology is the *AutomotiveWorld* (often also simply called *Root*). A few concepts are top-level collection concepts. They are *Users*, *Vehicles*, and *Trips*. The ontology generally has a user-centered view on the world, although the largest part of the remaining knowledge is unaffected by

which user is currently logged in and could be shared. For this reason, we differentiate between *user* and *context* model. In the work at hand, the user model is the more relevant one since it is the primary source for personalization, and the information obtained through Speaker Classification will be stored therein.

The most personal information including age and gender is located in the *Basic Dimensions* concept. This concept was created largely after the General User Model Ontology (GUMO, see Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005), which collects user dimensions such as physiological and emotional state, skills, and characteristics. The native GUMO part is complemented by contact information, which is one of the less detailed aspects of GUMO. Here, the extensible XML Contacts Schema⁹ was chosen for the representation of contact information, such as names, addresses, and phone numbers. *Preferences* are an important aspect of adaptive systems. Many preferences are application-specific, but there are also more general ones, which should be included in the ontology. We distinguish between preferences and interests, the latter being more abstract and less related to HMI. Preferences could be navigation preferences (e.g. types of road, driving at night), visual preferences (e.g. font, colors), interaction settings, privacy settings, and others. Interests include music, movie, literature, and other interests using an extensible taxonomy of categories, which would be used for instance in the recommendation of points of interest for navigation. *Interactions* seen from the perspective of a particular user are part of the user model. It contains discourse information on human-machine interaction, but also on inter-human interaction such as conversations. Dialog systems are heavily based on knowledge in this branch of the ontology. Knowing who is talking to whom also helps with resolving the topic of a discussion and creating an interaction graph. Another application is to determine whether a certain user is currently “occupied”, i.e. spending most of his cognitive resources on some resource-intensive task. Similarly, the *Presentation Model* is specific to one user. It is the primary source of information when the system autonomously adapts the user interface, a process that benefits if every adaptable entity has an instance representing it in the knowledge base. The model describes concepts of the user interface at a high abstraction level, so it does not take the role of a user interface manager or support rendering. The granularity is just sufficient to model the entities that are subject to adaptation rules. Examples are informational or warning messages, segments of speech output, or different display regions of the screen. Additional concepts are the *Authentication* node, which facilitates the identification of a user based on IDs or biometric data (e.g. voiceprint), and the *Services* node for storage of messages, appointments, contacts, etc.

There are several aspects of the automotive domain that should be taken into account for knowledge representation. Some of these aspects are so universal that they affect the general design of the model. For instance, as a car is in motion most of the time, the context may change very often and also quite dramatically even in a short interval. Thus, the aspect of *time* plays a vital role for the design of an in-car knowledge management system. Being

⁹<http://msdn.microsoft.com/enus/library/ms735869.aspx>

able to associate a point in time or some other recurring attribute with every instance and even property in the knowledge base allows advanced and efficient reasoning. Further, we are dealing with a lot of uncertain information in the car, e.g. data based on sensors. Therefore, having a *confidence* associated with each fact can help applications assess the reliability of knowledge. Another recurring factor is the concept of geographic referencing, i.e. *location*. Most context information in fact applies only within certain geographical boundaries. Finally, to enable rich collaboration and road community scenarios, a setting that specifies the *privacy* level (e.g. vehicle, traffic operators, and other road users) of a fact is needed.

The Automotive Ontology and the knowledge management of KAPCOM allow each value stored in the knowledge base to be annotated with one or more pieces of meta information according to the categories just introduced. This allows for example to keep a history of earlier knowledge, quickly filter instances by their geographic proximity, or decide what information can be shared with other vehicles.

Adaptation

Instead of hard-coding adaptation rules into the application and merely requesting the required knowledge from KAPCOM (path 1 in Figure 6.25), the component has been designed to provide a framework for a more universal handling of this task. Working with adaptation techniques ultimately leads to the discovery of certain recurring elements that describe the adaptation. In KAPCOM, these elements have been collected and summarized under the concept of an *adaptation strategy*. One goal of this concept is to provide a consistent way to describe adaptation. A second goal is to provide a programming model which allows applications to specify adaptation in a declarative and generic way. Without such a programming model, applications either have to rely on passive methods or need to handle all aspects themselves.

Adaptation strategies offer a generic way to describe an effect. Some typical examples are:

- Increasing / reducing information (e.g. number of elements)
- Changing the method for navigating (scrolling/paging) through long lists
- Adjusting colors
- Changing the layout (e.g. element size)
- Re-ordering items in a list
- Changing the instant of presentation
- Increasing / decreasing the duration of presentation (e.g. speech rate)
- Determining repetition behavior (i.e. interval length)
- Choosing output modalities

- Adjusting the style / mood of a message.

The ultimate goal is that strategies can be written to automatically take into account the current state of the ontology. However, this concept is currently still in the process of maturing, and is also believed to be beyond the scope of this work.

Audio Management

Apart from Speaker Classification, audio technology has many applications in the vehicle: Acoustic signals are used to convey warnings and alerts, ASR is used to give commands or – in conjunction with TTS – to navigate menus, they are used to read and compose messages, entertainment systems play back music or videos, telephone calls are made. For many of the experiments and demonstrators built as part of this work, the low-level access to audio signals and management of audio devices was therefore a prerequisite.

In the car, the required audio management capabilities are different from other environments, such as a personal computer. For instance, each seat may need to be accessed separately or in combination with certain other seats, depending on the situation and the role of the passengers. Also, the sharing of audio devices between applications is more critical. For example, incoming voice commands should generally be processed semantically only by a single application, while the voice features may be re-used for different classification algorithms.

A device ontology, which is part of the Automotive Ontology, provides a place where devices, such as each seat's loudspeaker and microphone, can be registered, but it does not offer access to the actual hardware. To fill this gap, the *Automotive Audio Manager* (AAM) has been created. It consists of a server component that can be controlled using a network protocol similar to the KAPCOM protocol, or through configuration files. It connects to both recording and playback hardware and implements low-level audio processing using the kernel-mode audio mixing API (also called *core audio*) in Windows.

For automotive applications, the use of AAM has several advantages: Most importantly, it handles several aspects of the sharing of physical devices. The built-in mixer can multiplex and mix several outputs, and doing so it can give priority to certain types of content (e.g. warnings) or adjust their volume depending on the situation. Similarly, microphone input can be forwarded to multiple targets for processing. For these purposes, it is tightly integrated with KAPCOM. As a special feature, the AAM can convert the audio from or to the application's format (e.g. by changing the sampling rate or bit depth) when needed. Further, post-processing and filtering can be added to the audio processing chain, e.g. to normalize volume or to apply high-pass filtering. It offers flexible audio source / target selection, including processing of audio over the network. Therefore, content, knowledge, and hardware do not need to be present on the same machine but can be distributed. Another advantage is that applications can abstract from physical devices. The AAM uses the concept of "logical devices" to refer to different locations inside the car and collections, such as the "driver's microphone" or the "rear speakers". An example is shown in Figure 6.27.

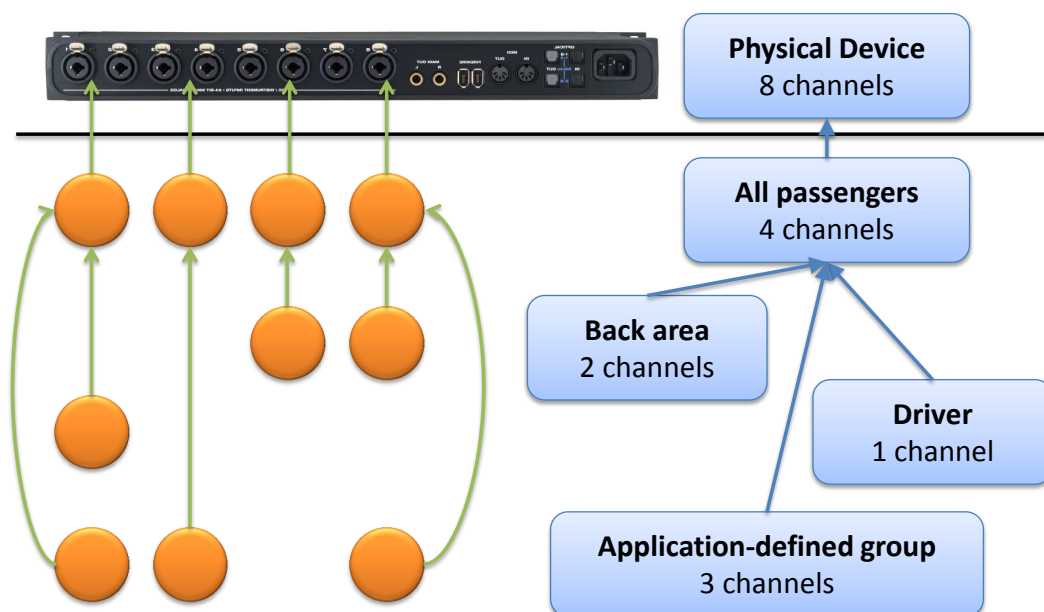


Figure 6.27: Logical devices can be used to refer to different locations in the vehicle by connecting them accordingly to a physical device. The depicted board is the *MOTU 8pre*.

Figure 6.28 illustrates the recording pipeline in AAM: The operating system sends the recorded audio data to the AAM, which has a *PhysicalDevice* instance associated with the OS device. In our demonstration vehicle, this is connected to the eight-channel MOTU sound board. Next, all logical devices that are based on the physical device, will receive the corresponding channels using multiplexing. For example, the front left *LogicalDevice* receives the channel connected to the front left microphone. Then, all filters registered for the device are applied, e.g. volume normalization. Each application or function that intends to work with audio data registers one or more connectors and a preferred audio format with the logical device. The filtered data is converted into the requested format by resampling. Finally, the connector sends the data to the application using one of several possible protocols, such as the real-time streaming protocol (RTSP).

6.3 Speaker Classification for Mobile Devices

The second domain for which the potential of Speaker Classification should be studied is the mobile devices domain. Mobile devices are generally all electronic gadgets that people carry with themselves, although this consideration is limited to the more intelligent and personal types of devices. The most widely used type of device today falling under this category are smartphones. Since their invention shortly around the beginning of this century, they have made a leap forward in terms of technology and popularity. Specifically after the introduction

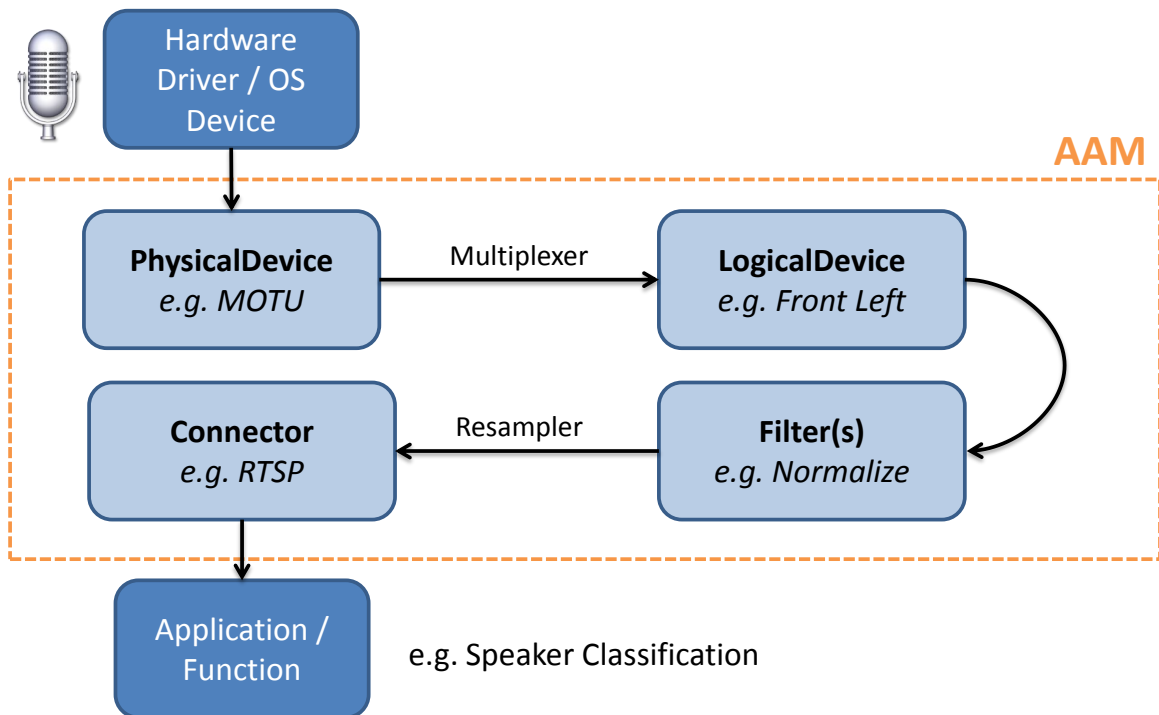


Figure 6.28: The audio recording pipeline in AAM. See the text for a description.

of the Apple iPhone in 2007, the percentage of people carrying such a device has reached respectable regions. Besides smartphones, personal digital assistants (PDAs), which can be considered their predecessors, belong to this category as well, as are larger handhelds and certain intelligent music players, navigation devices, and cameras.

The domain itself is quite different from the automotive domain. In this section, we will look at some of the special characteristics. From some of these observations we will derive motivations for Speaker Classification, while others influence the way Speaker Classification is performed. We will also see the concrete example of a mobile application, which uses age and gender recognition to personalize itself.

6.3.1 Characteristics of the Mobile Devices Domain

Most mobile devices, even more than vehicles, are tied to a particular person (hence also *personal* mobile devices). They are rarely shared or used by someone else. Most people consider them an integral part of their privacy due to the data stored on them (messages, contacts, call histories, photos...), and have also built up a certain dependency. For the goal of personalization, this is a great thing, because it allows many assumptions to be made about the user that would otherwise needed to be verified. With respect to Speaker Classification, it might however seem that the technology is not needed if the user never changes. While the argument is certainly true to some degree, there are still many situations where it can be

useful.

First, specifying a user profile that contains all the static information we could possibly extract from the voice (age, gender, height, language, accent, dialect, social status...) is not part of typical device usage. It is unlikely that users would accept providing this information as part of the device set-up, even if they had to do it just once, having only a vague idea about the possible personalization that it might entail. It is a general rule in UI design that such “non-essential” interfaces greatly drain from user satisfaction. Apart from that, many people dislike being asked for personal information explicitly, simply for psychological reasons or because they feel that the data might not be kept safe.

Second, the user might install apps that do not have access to the personal information stored on the device. For good reason, this information is not provided to apps by default even if it were present. In consequence, every app would have to build their own user profile, asking the same initial questions again if they feel they can provide enough value to the user in return to justify them. One such application that will be presented in more detail is a shopping assistant that can be installed when the user enters a correspondingly equipped local store. Using Speaker Classification, and provided the speech is available, the app can immediately start adapting itself without bothering the user with further questions.

Third, there are numerous paralinguistic properties that are dynamic, so they cannot be stored on the device in advance, personal or not. Examples are stress, cognitive load, emotion, tiredness, affect, and intoxication. Any kind of personalization that is based on these properties can make use of Speaker Classification on the device to get an estimate of the user’s current status.

And last, not all portable devices are personal. There are some devices handed out to users and collected again later, which also offer a rich user interface and functions similar to PDAs. For example, some museums and parks hand out digital guides that accompany visitors on their tours. Other possible scenarios include airports, where the primary purpose of the device would be navigation; libraries; or institutions where the use of personal devices is not allowed, such as certain medical or research facilities.

A different aspect of the mobile scenario is the mobility itself. People move all the time, so the device continuously changes its position. Unlike the in-car scenario, it is almost impossible to make any predictions about the set-up, such as whether the device will be held or in which pocket it will be kept. Also, there is no seat that could be instrumented. In effect, this means that there are fewer alternate sensory sources that might be used instead of or parallel with Speaker Classification, such as weight sensors or a static camera.

As the environment is also changing more dramatically compared to the relatively well-shielded car, the system should be more robust with respect to noise and interference. On the upside, it enables diverse scenarios as the device is taken to indoor locations.

Also mobility-related is the aspect of distraction. Like drivers, pedestrians are traffic participants, and need to devote a certain share of their attention to the avoidance of collisions and other accidents. However, the criticality is much lower than in the automotive domain, as pedestrians move slower and have the choice to walk in safer areas, e.g. on the sidewalk.

Nevertheless, mobile applications should take the aspect of distraction into account. Even better, they should be designed “eyes-free”, so that they can also be controlled from the pocket. This again highlights speech as a valuable modality, particularly for output (e.g. using headphones). Yet, as for the input, the mobile device usually does not have to be “hands-free”, but it is still more comfortable to keep the device in the pocket and use speech to control some core functions.

Finally, mobile devices are characterized by their limited hardware resources. This makes it necessary to provide an optimized, compact, and possibly less memory- and CPU-intensive implementation of Speaker Classification, which is the exact purpose of embedded classification modules presented in Section 5.4 (see also [Feld & Müller, 2008](#)). Early studies performed on PDAs with the original AGENDER system ([Feld, 2005](#)) have shown that the classification time can indeed increase beyond tolerable limits. By optimizing the feature extraction algorithms, the classification time could be reduced by an approximate factor of 9¹⁰, as shown in Figure 6.29. Still, the low computing power is a motivation to consider the server-based classification scenario: In stand-alone, local scenarios, classification should be able to run fully on the device. This corresponds largely to the desktop-based classification, except that all processing is done on a machine with much less computational power. However, if there is a connection to a more powerful server for classification at our disposal, we also want to be able to take advantage of this additional resource. However, as the user is expected to be constantly changing his or her location, we cannot assume a permanent network connection in the server-based classification scenario, but need to be able to handle sporadic connections. There are many other hardware properties that might affect the way personalization occurs, such as the size factor.

6.3.2 A User-adaptive Mobile Shopping Assistant

This application presents a working implementation of Speaker Classification on a mobile device platform. Specifically, it is based on an application called *Mobile ShopAssist*. It can be used to look up information on products in a physical shop to help the user decide before buying. Speaker Classification will help to add product and feature recommendations to the application. The work was originally presented in [Feld and Kahl \(2008\)](#).

Goal

The Mobile ShopAssist is a PDA-based multilingual multimodal digital shopping assistant ([Wasinger, 2006](#)). The relevant input and output modalities are: speech, handwriting/text, and gesture. With the Mobile ShopAssist demonstrator, it is easy to retrieve the specific information about different products. For this purpose, the user can write the name of a product and/or the requested feature on the touch-sensitive display of the PDA. Clicking on

¹⁰The optimization generally reduced the amount of interpolation applied during the computation of F0, which had a huge impact on runtime performance, but a relatively small degradation only of classification accuracy. For more details, see [Feld \(2005\)](#).

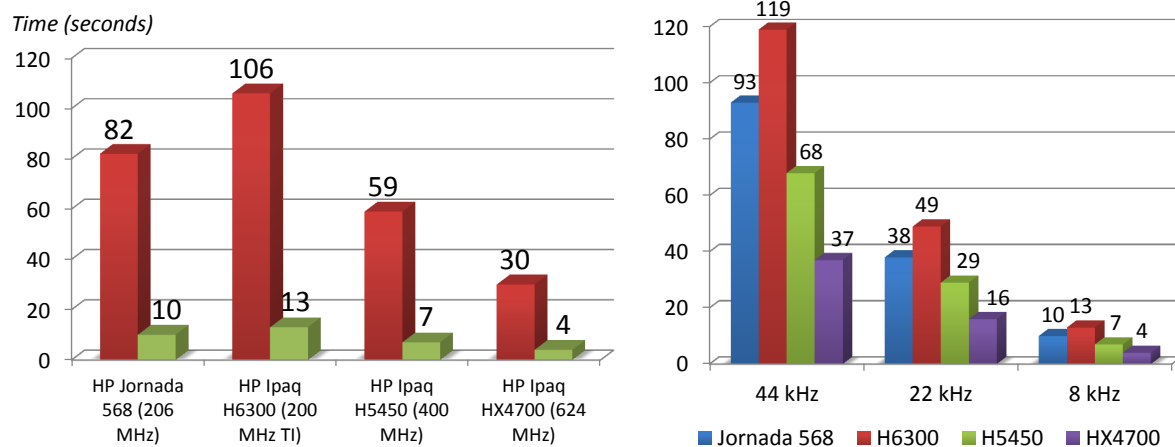


Figure 6.29: Runtime performance of classification with AGENDER on various PDAs. *Left:* Comparison between the original (red) and resource-adapted (green) feature extraction routines. An utterance of 2.3 seconds and 8 kHz was used. *Right:* Comparison of feature extraction between different PDAs and sampling frequencies.

the product image represents another input mode. For the gesture input of the feature, the user can click on the corresponding expression in a scrolling text bar. The third alternative for input is speech, which is recognized using *IBM Embedded ViaVoice*. These input modalities can be combined, for instance speech input for the feature (e.g. *price*) and gesture input for the object (e.g. *iPhone*) (Wasinger, Stahl, & Krüger, 2003). After processing the input, the system outputs the value of the requested feature. Furthermore, this assistant facilitates the comparison of two products by contrasting their features.

The Mobile ShopAssist was designed as a client-server architecture. After the user has logged in, the system loads his or her specific profile from the user model managing system *UbisWorld* (Heckmann, 2005). In the previous version, there was only a standard profile for users who had no *UbisWorld* login. According to the approach presented in this paper, this standard profile can be adapted dynamically to the current user. The product data is sent on the PDA when the user stands in front of the corresponding shelf.

Our goal is to improve the Mobile ShopAssist's modalities to support the user in making informed decisions about products offered in a shop by dynamically integrating knowledge about the user in the information filtering process.

Consider the following scenario: A 43 year old customer enters a mobile phone store with the intention to buy a new phone. Looking at the products on display, he is presented with an overwhelming number of articles. He finds it hard to choose from them by just looking at the outer packaging, especially since many of the advertised features such as games and ringtones are rather aimed at teenagers. Using the Mobile ShopAssist on his PDA, he can scan through the core information for each phone more quickly, but the comparison is still

time-consuming and involves a lot of manual filtering.

This application outlines how Speaker Classification can be utilized in this and similar scenarios to provide the necessary information about a user for whom no profile data is available. It shows how this information is then used in the Mobile ShopAssist application to adapt the displayed information according to the user profile obtained from the voice. This version uses the seven classes from AGENDER (see Table 4.1 on page 104): *children* (up to 12 years), *teenagers* (13-20 years), *adults* (21-64 years) and *seniors* (65 years and older).

Recommendations and Score Computation

Today, we find recommendations on many sites that offer products on the web. For every item a website visitor views, there may be a personalized list of recommendations. The underlying technology that is responsible for generating the recommendations, a recommender system, can range from a simple content-based approach to complex collaborative filtering. The user profile is an important part of the system's filtering algorithm and typically contains items that the user himself has rated or items that he has viewed. When the user has provided explicit demographic information, this could be used to complement the data from the web service. However, it may take quite some time before a reliable profile has been generated, and before that is the case, recommendations may often appear random or plain misfit. A user may even choose to completely opt out of signing up, be it for convenience or privacy reasons. In either case, Speaker Classification is seen as an appropriate way to bootstrap a user profile when there is no other information available yet.

For our scenario, we have created a product assortment for a fictitious mobile phone store. Mobile phones are one category of products, on the choice of which the speaker characteristics presented here have quite a large impact. We chose *teenagers*, *adults*, and *seniors* as target age groups for this application.

The content for the basic recommender system we have implemented in our prototype is provided by product data annotations. For each mobile phone feature, e.g. "WLAN capability", we can specify recommendation ratings. A rating can be created for every subset of user classes and indicates whether that feature is often requested and typically important for this audience w.r.t. the feature's setting. For example, WLAN functionality may receive an increased rating for the *adults* group, while ring tones and gaming capabilities (*Java*) are rated high for the *teenagers* group. In the same way, features that make the mobile phone more accessible to elderly people, like readability, can be promoted. Table 6.2 lists some of the features we incorporated in this application. To obtain a recommendation score for a product, all ratings matching the current user's profile are added together. It should be pointed out that the choice of the actual parameters in a real application needs to be subject to adequate market research in the chosen product category. Market studies done by dedicated market research enterprises could serve as a reference.

Feature	Style / Color	Weight	UMTS	Display Size	Digital Camera
Suggested Relevance	custom, per product	lower weight preferred by <i>women</i>	mostly requested by <i>adults</i>	large displays preferred by <i>seniors</i>	good cameras important for <i>teenagers</i>
Samsung SGH U 600	classic	81g	no	5.59cm	3.2 megapixels
Sony-Ericsson S500i	fancy	94g	no	5.08cm	2 megapixels
SecuPoint Secu-B	old-fashioned	130g	no	large, text-only	none
Siemens SXG 75	classic	134g	yes	5.5cm	2 megapixels

Table 6.2: Selected mobile phone features and phones that may exhibit statistically significant influence on purchase decisions for certain audiences.



Figure 6.30: The Mobile ShopAssist showing the list of product available in the shop. The list has been sorted according to the recommendation ratings computed based on the current user's age and gender.

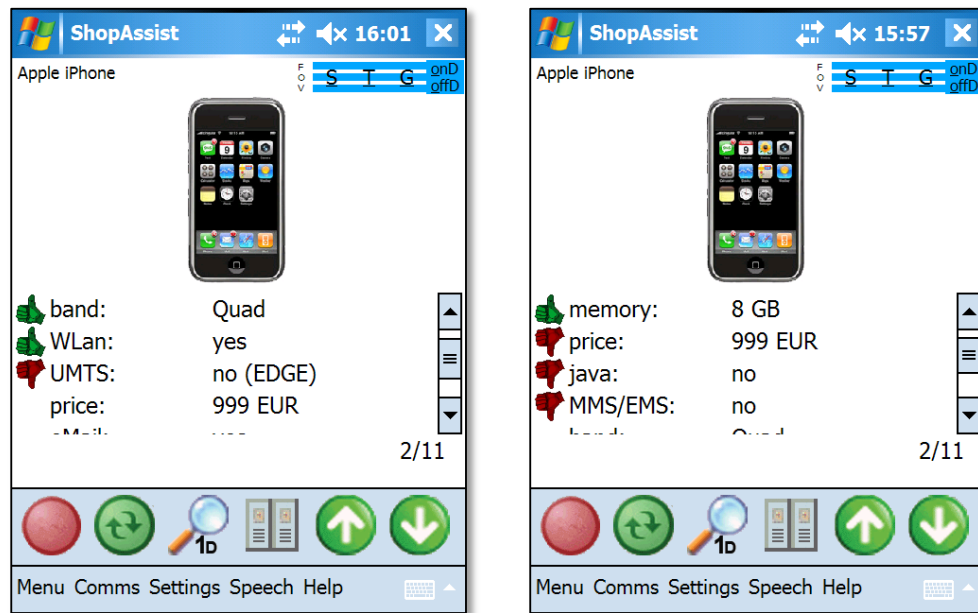


Figure 6.31: Screenshots of the Mobile ShopAssist in product details mode: The left image shows the specific order of the *iPhone*'s features for an *adult* user whereas the right image shows the product's feature ordering for a *teenager*.

Adaptation of the User Interface

The speech that is used to control the Mobile ShopAssist is also forwarded as input to the classifier. The speech data is transferred over a WLAN TCP connection to a server running the classification. The application can then request an updated user profile from the server. Using this model, we provided two means of GUI adaptation:

1. The user-adaptive product list is sorted according to the product's score (see Figure 6.30).
2. The detail page highlights features of special relevance (i.e. score) of both supporting and inhibiting type (see Figure 6.31), e.g. the price of a mobile phone is less important for adults than for teenagers.

Although this application is relatively simple, it suits well to demonstrate the various ways how personalization based on Speaker Classification can be integrated into existing mobile applications. Since there was no user study, the effectiveness of this particular adaptation cannot be rated. It is however believed that, given appropriate market research data, it could help people find the products they are interested in more quickly.

6.4 Speaker Classification for Phone-based Services

The third example of a domain where personalization can be based on Speaker Classification are telephone-based services. It is a natural domain for speech technologies and also responsible for much of the motivation presented in previous work such as AGENDER (Müller, 2005; Feld, 2006). Like in the other examples, we will give a description of special aspects of the domain and present some example scenarios. The most common personalization applications are in the area of automated response services and call centers.

6.4.1 Characteristics of the Phone-based Services Domain

If we consider the number of alternate sources of user knowledge available, it has been decreasing in the first two domains, and is reaching its lowest level for telephones. With the exception of video phones, audio is the only channel that can be used to non-intrusively obtain user information. It is also much less likely that users will have a profile already available, mostly because there is no easy and convenient way to associate the caller with the profile. On the other hand, however, it is usually guaranteed that there will be an ample amount of speech available.

The channel through which the user calls has a strong impact on the classification algorithm. Depending on the carrier and network conditions, there may be considerable noise that can overlay the features used for classification. Especially in calls from cellular networks, there can be interruptions and coding-related artifacts, which degrade the method considerably if they are not filtered. Like in the mobile devices scenario, users may be calling from outside locations with environmental noise.

Depending on the application, it might be necessary to have the result of Speaker Classification prior to a certain point in the dialog, such as in ACD (see Section 6.4.3). Here, we face the problem of *initiation*: In order to react accordingly to the user, e.g. by forwarding the call, we first need to have some input to base our decisions on. A possible solution could be to combine this with an early prompt that records some basic information from the caller, such as a problem description or a topic selection. The case is more difficult when the adaptation is more basic, such as the adaptation of the dialog language. It is not an easy task to communicate to the user that she may start talking in her native language without giving an introduction first, possibly in multiple languages.

From a technical perspective, call centers are usually backed by a large telephony infrastructure, which manages call distribution, waiting queues, automated dialogs, speech synthesis, etc. Therefore, the implementation of Speaker Classification needs to be very scalable to run on many channels in parallel. It should also be able to use the resources available through the server infrastructure to its advantage, such as by using more complex user models or to work at higher frame rates.

In phone-based services, there is also the additional option of exploiting the feedback channel to improve classification. Sometimes, callers are asked for their personal information or are associated with a customer profile where this information is already present. It is possible

to use this information to post-hoc confirm or reject the prediction, which corresponds to a new training sample. The current version of FRISC does not support live training, but it would be possible to add this feature in the future. Doing so would not only help to improve the general classification accuracy, but it would also serve as an adaptation to the specific call center and possibly type of clientele.

6.4.2 Recording Caller Information

It is very valuable for call centers to have statistical information about their callers, for instance to optimize training of their staff or even to learn about their customer base in general so they can adjust their products and advertisements. If Speaker Classification is run on the incoming calls, it can provide exactly that without any additional effort. By just accumulating caller data, some insight can be gained. Most privacy concerns can also be dispelled since the data is anonymous. Such a system based on the FRISC system has been running for some time in a large telecommunications industry call center in Germany.

Recording caller information is also valuable in all areas related to security, such as an emergency number.

6.4.3 Automatic Call Distribution

Many call centers train their agents to properly deal with the difficult situations they encounter every day. There are several well-known user groups for which it is helpful to have a certain level of background knowledge, such as elderly people (who might be forgetful or have a different approach to many things than younger generations) or children. Also, some calls are very emotional, e.g. when customers give vent to their anger about a product, a deescalating strategy should be applied. In both cases, the caller receives a more “personal” treatment. Such training is most effective when different agents serve different target groups or cases. Also, new agents usually do not have the experience to deal with the more difficult cases appropriately. The challenge in both cases is to determine the agent who should pick up the call in advance. This is where *automatic call distribution* (ACD) helps.

With ACD (see Figure 6.32), callers are first classified by a system according to several criteria, such as their age, gender, and emotional state. Based on this knowledge, a corresponding agent is chosen to pick up the call. There is still some room in how the initial classification works. One option is to collect some a priori information about the intention of the call. Today, many call centers already do this to perform a different kind of ACD, namely to forward callers by topic. They do for instance present the caller with a prompt asking them to tell about the cause of their call. For example, a technical support hot-line might have an initial prompt of “Please tell us about the problem you are experiencing”. They might then run a word spotting algorithm on the recorded speech data to identify keywords and then select the agents based on their knowledge about the corresponding topics, which is similar to a menu selection.

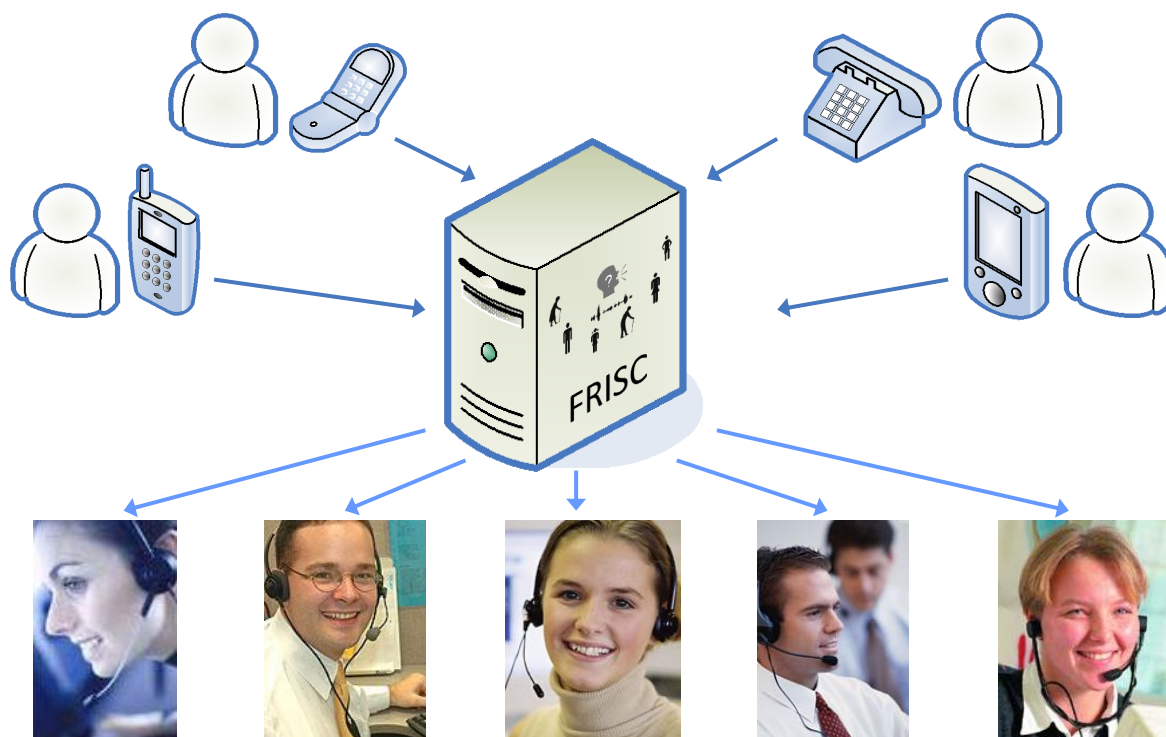


Figure 6.32: Illustration of the concept of automatic call distribution (ACD).

6.4.4 Dialog Adaptation

A third possible application in phone-based services is mainly targeted at fully automated dialog systems. In this case, the system can be personalized by making its behavior user-adaptive. We distinguish between form and function as the target of the adaptation (see Section 6.1.5). Adapting the form hereby refers to a change in the dialog structure, while the function refers to the actual content being returned.

An adaptation of the *form* might for example be performed based on the age of the user. Elderly users might receive more guidance by introducing additional prompts or repeating the information already presented. Young callers could be presented with additional explanations and confirmation prompts. If the system determines that the speaker is possibly in a hurry because the voice indicates stress, the dialog might be more streamlined, leaving out additional offers. *Nuance* has recently proposed a system for use in *T-Mobile* call centers where gender and language of the speaker are used to adapt the same features for the TTS employed in the system, which, in case of gender, resembles the approach taken in Section 6.2.5. With respect to age, they also suggest certain high-level settings of the ASR to be adapted, such as to favor single-word recognition over complex grammars.

Adaptation of dialog *function* has also been demonstrated in various contexts, such as an automated cinema ticket hot-line where the caller can ask for a movie suggestion. This suggestion would be based on the detected age and gender information, e.g. it might prefer

family movies for children, action movies for male teenagers, and more classical theater for seniors. Further, children can be directed away from movies they would not be allowed to see due to their age.

Dialog adaptation can also be combined with ACD. In the system proposed by Nuance, the caller can be transferred to a human agent when an anger condition is detected.

In partially automated systems, it is also possible to adapt some aspects of the dialog, such as the order in which the topics in a menu selection are presented. Likewise, the music or announcements presented to the caller, when she is waiting for an agent or when put on hold can be personalized.

7 Conclusion

This last chapter serves as a summary of the work done in this thesis. Apart from discussing the results, an outlook on possible future research is appended.

7.1 Research Questions Revisited

In order to highlight the contributions of this thesis, in this section, we will revisit the questions posed introductorily in Section 1.2, and summarize the answers given throughout this essay.

1. **What are the components of a Speaker Classification approach for the estimation of speaker age, gender, and other characteristics, that is designed to deliver high classification performance, yet be sufficiently flexible to be applied to a variety of classification problems in multiple application domains?**

In the thesis at hand, we have constructed a Speaker Classification approach called FRISC and tested it in the context of age and gender recognition. The decisions leading to the system design as is were generally derived from motivations found in literature close to the phonetical foundations of speech production, as opposed to algorithmic methods based on large but random feature collections. A study on synthesized voices has served to strengthen the relationships between the two sciences and indicates that similar age cues are indeed contained in these models. Likewise, the results were interpreted in the light of the underlying processes. In this regard, we have given arguments for the use of MFCCs as the primary frame-based feature in this system. Prior to the design of the system, studies of other systems have pointed out strengths and weaknesses of the various methods. The final solution is a GMM-SVM supervector approach, which combines generative and discriminative models, and which has proven to be a good composition. By using hybrid short and long term feature classifiers, we can confirm that the short-term features contain some information that is not present in the long-term counterpart. Even though only age and gender were considered as single speaker characteristics, the approach is sufficiently general to be also applied to other speaker properties, such as stress and emotion. The architecture of this approach is flexible enough to be used in embedded and mobile scenarios. It contains several parameters that can be adjusted to account for limited resources, e.g. the step width of MFCC features or the number of mixtures in the GMM. In most configurations, including the one achieving the best classification performance, the runtime of even a development-phase version performs considerably faster than real-time.

2. **Which are the parameters that affect the performance of such a Speaker Classification system, and how can a principled approach be created to empirically determine the relative impact of each of them?**

This work is the only one of its kind known to the author that incrementally explores an extensive parameter space in a rigidly structured experiment for age and gender classification. The basic experimental procedure follows even stricter rules with respect to

transparency and data set composition (see Section 5.5.3) than many official machine learning challenges. The examined parameters have been selected from all stages of the classification pipeline, and include both parameters commonly discussed in other publications, such as the number of GMM mixtures and SVM kernel function, as well as less reported-on parameters, like the MAP relevance factor and the GMM initialization function. The results of this main experiment series are directly comparable because they have been obtained on the same data sets and with all parameters except one identical in each condition. There was also no same-set tuning involved, and the final evaluation was performed only once. This experiment provides plenty of insight on these parameters, which have been discussed individually in Section 4.3.3, for example the importance of the MFCC step width and the low impact of silence filtering. In this main series, we have also intentionally refrained from methods that may improve the numbers, but provide little insight or even obscure the process, such as repeated fusion. Given these characteristics, we describe it as a principled approach to parameter investigation. Even though the complexity of a single experiment run has prevented us from exploring all possible parameters and configurations, we can now also produce more elaborate decisions on the areas that need further study.

3. Which aspects of the approach can ensure the incremental processing of information, i.e. a fast initial estimation and a continuously increasing expected performance with the arrival of additional speech data?

The FRISC algorithm has been designed from ground up to work in incremental manner. This allows it to be used in domains where user adaptation has to be performed on only a small segment of data, but further speech can be used to improve the result over time, e.g. in the case of driver adaptation. Several aspects which facilitate the incremental behavior have been pointed out throughout the thesis, most importantly the frame-wise data handling (see Section 4.1.4). Using this approach, classification can be performed with any number of frames, while more frames still generally result in better recognition. Several aspects of MFCC extraction that limit other implementations to an off-line scenario have been configured to work in *live* scenarios. The implementation of MAP adaptation employed is particularly suitable for incremental processing by the use of post-build parameter re-estimation (see Section 5.3.4). The remaining parts of the architecture have also been chosen to be compatible with this concept.

4. Is speaker age and gender estimation or the Speaker Classification approach as a whole influenced by the speaker's language or culture?

In Section 2.1.4, we have learned of the existence of language and culture dependent factors in speech that might affect how Speaker Classification performs when used in different regions of the world. Not only to demonstrate the universal applicability of the approach, but also since globalization is a topic of practical relevance, we have set out to investigate the effects of cross-cultural classification in a cooperative study with researchers in South Africa using a special corpus. As far as the data is concerned, the

variety of languages and language families encountered in this country can be considered ideal with respect to this goal. The study, which is described in Section 4.5, consisted of an analysis part, that studies the effects of the different languages on high-level features, and a cross-language applicability part, which examined how much language-specific training affects the performance. Although the experimental part was not as extensive as the main series, the results reveal that there are indeed aspects of age and gender which are different between cultures, which is confirmed by the numbers obtained. This means that the approach as a whole works universally, but the models will be restricted to the culture used for training and have to be adapted to the target audience. Hence, in a global application, language identification should be fused with recognition of other properties in a single system to provide optimal results.

5. How can the approach developed for age and gender be used to recognize the identity of a user through his or her voiceprint?

Speaker recognition and Speaker Classification are closely related. As seen in Section 2.4.1, the one can be seen as a special case of the other. Motivated by a practical application in the vehicle, namely determining passenger positions based on ambient speech, we investigated the feasibility of FRISC for biometry in Section 6.2.6. As it turns out, the approach achieves very low error rates in a ten-speaker detection set-up. Some parameters, such as the number of mixtures, were adjusted to the new task. Since the final version was also running on a low-power mobile device, the architecture was reduced to a conceptually simpler GMM-UBM, yet the majority of the framework was identical. Generally, the study suggests that having a versatile feature such as MFCCs as a basis, one can move from specific characteristics like age and gender to other and more generic voiceprints.

6. How can an architecture or framework be designed that allows speech-based classification technology to be created, evaluated, and deployed according to the diverse requirements in desktop, server, and embedded scenarios?

A substantial part of the work was dedicated to making Speaker Classification robust and accessible for use in a multitude of application scenarios. At the core of this strategy, this thesis presents SPEACLAP – a framework and integrated development environment dedicated to the design and evaluation of speech-based pattern recognition technology. Its exclusive focus on speech as the input data distinguishes it from numerous other machine learning environments and makes it particularly suitable for the task. In Section 5.5, we have seen how even a complex set-up like the FRISC main experiment series can be realized in SPEACLAP. Section 5.6 provided numeric evidence on the IDE’s runtime performance for both training as well as classification times on the large corpora used in these experiments. Coping with the even stricter runtime requirements of server and embedded scenarios is facilitated through the introduction of embedded modules (see Section 5.4). These are designed within the IDE and then compiled to optimized code in a fully automated build process. For each build, the configuration can

be selected that matches the target platform in terms of operating system, computing power, memory, etc. Further outstanding features of the environment that were detailed in Section 5.3 are its support for different optimized storage providers, the ability to efficiently deal with very large data volumes, the all-in-one concept of the IDE including extensibility using compiled scripts written in the integrated editor, the advanced job management and scheduling engine, its data and report archival mechanisms, and many more.

7. How can non-intrusively acquired speech-based user characteristics such as age and gender help to personalize the user experience in user-centered domains with different characteristics and demands?

The contribution of this thesis is not limited to the development of Speaker Classification technology, but also considers the advancement of personalization technology based on it as one of its primary goals. With a special focus on the automotive domain, several personalization concepts have been developed that serve as a basis for the integration of Speaker Classification with other automotive applications. The KAPCOM knowledge component (Section 6.2.8) was created in order to supply well-structured management of user knowledge obtained from speech and other sources, and share it with functions and possibly even other vehicles. As part of this thesis, an application that adapts the SIM^{TD} in-vehicle information system to the user has been implemented to demonstrate how both age and gender recognition can benefit personalization. This particular example improves both safety for elderly drivers, as well as comfort for other groups. A further demonstration shows how the user's profile can be associated with the seat taken by using only speech to make the connection. Using this novel concept, we can adapt content in both car and device to particular persons or to the role of the user in the vehicle. Aside from automotive applications, a short survey of Speaker Classification in the domain of telephone-based services and mobile devices is presented. In the latter case, we also implement an application based on a mobile shopping scenario that has been augmented to recommend different products and features for different target groups.

7.2 Outlook

In the last years, we have seen an increasing interest in Speaker Classification technology. With the contributions to its maturity by the work at hand, the goal of seeing more real-world applications based on it is expected to be one step closer. In spite of this progress, the field is quite large and there are numerous open issues that could not be addressed.

Based on the results of the experiments conducted in Chapter 4, including both parameter exploration and fusion, it should now be reasonable to return to a higher abstraction layer and study the fusion aspects of classification systems in a similarly rigorous evaluation, e.g. comparing how different feature classes, feature sets, and classifiers can be combined. For

age and gender, we might also want to go one step further and consider fusion with other knowledge sources, such as camera images and ambient sensors. Having the system automatically choose the best source available is a popular strategy in computer science (e.g. as with “always-best-connected”).

Then, we want to go beyond age and gender to study further speaker properties, in particular those that have not received as much attention as other, but are very relevant in such active communities as the automotive field. Examples are the attention state (or drowsiness), distraction, and level of alcohol consumed.

Obviously, for the classification framework, the next step is to encourage its utilization in audio-based classification tasks. Since it is freely available, it might pose an interesting alternative to other, more generic tools due to its special features in this field. In parallel, there are still several areas where its functionality and robustness should be improved in the near future. Examples are native integration of more standard classification algorithms (e.g. neural networks or decision tree learning), providing more embedded module templates for mobile platforms such as *Android* or the *iPhone*, integration of fusion layers based on Bayesian network due to their high popularity, and additional convenience features for working with reports and graphs.

One of the most pressing issues however is also to intensify the consideration of the personalization aspects. In this work, we have just scratched the surface of research that deals with these topics. Again, the vehicular context entails great opportunities for advances that will benefit society immediately. At the moment, we are only considering a single source of information for personalization. Future use cases will be based on much more complex scenarios though, taking into account the user’s driving history, preferences, skills, and – equally important – the driving context, such as traffic density and local events. We then also intend to study the aspects of adaptation in greater detail, incorporating a way for a system to automatically select the best adaptation strategy in a given situation. Finally, we need to deal with the issues that are inevitably linked to adaptation: visibility, explicability, and reversibility are the top challenges here.

References

- Ajmera, J. (2006, June). Effect of age and gender on LP smoothed spectral envelope. In *Proceedings of Speaker Odyssey* (p. 1-4). IEEE.
- Amditis, A., Andreone, L., Polychronopoulos, A., & Engström, J. (2005, July). Design and development of an adaptive integrated driver-vehicle interface: Overview of the AIDE project. In *Proceedings of IFAC 16th World Congress* (p. 1195-1195). Prague, Czech Republic: Elsevier.
- Assmann, P. F., & Nearey, T. M. (2007, August). Relationship between fundamental and formant frequencies in voice preference. *Journal of the Acoustical Society of America*, 122(2), 35-43.
- Barak, B., & Schiffman, L. G. (1981). Cognitive age: A nonchronological age variable. *Advances in Consumer Research*, 8, 602-606.
- Barnard, E., Davel, M., & Heerden, C. van. (2009, September). ASR corpus design for resource-scarce languages. In *Proceedings of the 10th Annual Conference of the International Speech Communication Association (INTERSPEECH 2009)* (p. 2847-2850). Brighton, United Kingdom: ISCA.
- Barras, C., & Gauvain, J.-L. (2003, April). Feature and score normalization for speaker verification of cellular data. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)* (Vol. 2, p. 49-52). Hong Kong, China: IEEE.
- Bellegarda, J. R. (2007). Language-independent speaker classification over a far-field microphone. In C. Müller (Ed.), *Speaker Classification II: Selected Projects* (Vol. 4343, p. 104-115). Springer Berlin / Heidelberg.
- Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 1-127.
- Bezooijen, R. van. (1995). Sociocultural aspects of pitch differences between Japanese and Dutch women. *Language and Speech*, 38(3), 253-265.
- Birkholz, P., Kröger, B. J., & Neuschäfer-Rube, C. (2011). Model-based reproduction of articulatory trajectories for consonant-vowel sequences. *IEEE Transactions on Audio, Speech and Language Processing*, 19(5), 1422-1433.
- Bishop, C. M. (2007). *Pattern recognition and machine learning* (Corr. 2nd printing ed.; M. Jordan, J. Kleinberg, & B. Schölkopf, Eds.). Springer New York.
- Blom, J. (2000, April). Personalization – a taxonomy. In *Proceedings of the CHI 2000 Conference on Human factors in Computing Systems: extended abstracts* (p. 313-314). The Hague, Netherlands.

- Bocklet, T., Maier, A., Bauer, J. G., Burkhardt, F., & Nöth, E. (2008, March). Age and gender recognition for telephone applications based on GMM supervectors and support vector machines. In *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'08)* (p. 1605-1608). Las Vegas, NV, United States: IEEE.
- Bocklet, T., Stemmer, G., Zeissler, V., & Nöth, E. (2010, September). Age and gender recognition based on multiple systems - early vs. late fusion. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2830-2833). Makuhari, Japan: ISCA.
- Boersma, P. (1993). Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *Proceedings of the Dutch Institute of Phonetic Sciences (IFA)* (Vol. 17, p. 97-110). Amsterdam, Netherlands.
- Boersma, P. (2001). Praat, a system for doing phonetics by computer. *Glott International*, 9(5), 341-345.
- Boersma, P., & Weenink, D. (2011). *Praat: doing phonetics by computer (version 5.2.27)*. Computer program. (Retrieved June 19, 2011, from <http://www.praat.org/>)
- Brandherm, B., & Jameson, A. (2004). An extension of the differential approach for Bayesian network inference to dynamic Bayesian networks. *International Journal of Intelligent Systems*, 19(8), 727-748.
- Brouwer, R. F. T., Hoedemaeker, M., & Neerinx, M. A. (2009). Adaptive interfaces in driving. In D. Schmorow, I. Estabrooke, & M. Grootjen (Eds.), *Foundations of Augmented Cognition. Neuroergonomics and Operational Neuroscience* (Vol. 5638, p. 13-19). Springer Berlin / Heidelberg.
- Brümmer, N. (2004, January). Application-independent evaluation of speaker detection. In *Proceedings of Odyssey 2004 Speaker and Language Recognition Workshop* (p. 1-8). Stellenbosch, South Africa: ISCA.
- Bubb, H. (2003). Der Fahrer im 21. Jahrhundert [The driver in the 21st century]. In (p. 25-33). Braunschweig: VDI-Verlag.
- Campbell, W., Sturim, D., Reynolds, D., & Solomonoff, A. (2006, May). SVM based speaker verification using a GMM supervector kernel and NAP variability compensation. In *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'06)* (p. 1-4). Toulouse, France: IEEE.
- Castronovo, S., Endres, C., Del Fabro, T., Schnabel, N., & Müller, C. (2011, February). Multimodal conspicuity-enhancement for e-bikes: the hybrid reality model of environment transformation. In *Proceedings of the 16th International Conference on Intelligent User Interfaces (IUI 2011)* (p. 433-434). Palo Alto, CA, USA: ACM.
- Castronovo, S., Mahr, A., Pentcheva, M., & Müller, C. (2010, September). Multimodal dialog in the car: Combining speech and turn-and-push dial to control comfort functions. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 510-513). Makuhari, Japan: ISCA.
- Chang, C.-C., & Lin, C.-J. (2011, May). LIBSVM: A library for support vector machines.

- ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 27:1-27:27.
- Chong, J., Friedland, G., Janin, A., Morgan, N., & Oei, C. (2010, June). Opportunities and challenges of parallelizing speech recognition. In *Proceedings of the Second USENIX Workshop on Hot Topics in Parallelism* (p. 1-7). Berkely, CA, United States: ACM.
- Corkill, D. D. (1991, September). Blackboard systems. *AI Expert*, 6, 40-47.
- Cornuéjols, A., & Miclet, L. (2008, May). What is the place of machine learning between pattern recognition and optimization? In *Teaching Machine Learning Workshop (TML 2008)* (p. 1-6). Saint-Etienne, France.
- Davis, A., Nordholm, S., & Togneri, R. (2006, March). Statistical voice activity detection using low-variance spectrum estimation and an adaptive threshold. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2), 412-424.
- Davis, S. B., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 357-366.
- Dellwo, V., Huckvale, M., & Ashby, M. (2007). How is individuality expressed in voice? An introduction to speech production and description for Speaker Classification. In C. Müller (Ed.), *Speaker Classification I: Fundamentals, Features, and Methods* (Vol. 4343, p. 1-20). Springer Berlin / Heidelberg.
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1-38.
- Deutsch, D., Jinghong, L., Dooley, K., Henthorn, T., Shen, J., & Head, B. (2009, August). Absolute pitch and tone language: Two new studies. In J. Louhivuori, T. Eerola, S. Saarikallio, T. Himberg, & P.-S. Eerola (Eds.), *Proceedings of the 7th Triennial Conference of European Society for the Cognitive Sciences of Music (ESCOM 2009)* (p. 69-73). Jyväskylä, Finland.
- Dobry, G., Hecht, R. M., Avigal, M., & Zigel, Y. (2009, September). Dimension reduction approaches for svm based speaker age estimation. In *Proceedings of the 10th Annual Conference of the International Speech Communication Association (INTERSPEECH 2009)*. Brighton, United Kingdom: ISCA.
- Dötzer, F., Kosch, T., & Strassberger, M. (2005, June). Classification for traffic related inter-vehicle messaging. In *Proceedings of the 5th IEEE International Conference on ITS Telecommunications* (p. 1-4). Brest, France: IEEE.
- Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification* (Second ed.). New York, United States: Wiley-Interscience.
- Endres, C., Schwartz, T., Feld, M., & Müller, C. (2010, February). Cinematic analysis of automotive personalization. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI 2010)* (p. 1-6). Hong Kong, China: ACM.
- Endres, C., Schwartz, T., & Müller, C. (2011, February). Geremin: 2D microgestures for drivers based on electric field sensing. In *Proceedings of the 16th International Conference on Intelligent User Interfaces (IUI 2011)* (p. 327-330). Palo Alto, CA,

- United States: ACM.
- Eyben, F., Wöllmer, M., & Schuller, B. (2009, September). openEAR – Introducing the Munich Open-source Emotion and Affect Recognition toolkit. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII 2009)* (p. 576-581). Amsterdam, Netherlands: IEEE.
- Fawcett, T. (2006, June). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861-874.
- Feld, M. (2005). *Portierung von Merkmalsextraktion auf die PocketPC-Plattform [Porting feature extraction to the PocketPC platform]* (Technical Memo No. TM-05-01). Saarbrücken, Germany: Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI).
- Feld, M. (2006). *Erzeugung von Sprecherklassifikationsmodulen für multiple Plattformen [Creating Speaker Classification modules for multiple platforms]*. Diploma thesis, Department 6.2 Computer Science, Saarland University, Saarbrücken, Germany.
- Feld, M. (2008, August). Embedded modules for Speaker Classification. In *Proceedings of the Second IEEE International Conference on Semantic Computing 2008 (ICSC 2008)* (p. 370-377). Santa Clara, CA, USA: IEEE.
- Feld, M., Barnard, E., Heerden, C. van, & Müller, C. (2009, December). Multilingual speaker age recognition: Regression analyses on the Lwazi corpus. In *Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2009)* (p. 534-539). Merano, Italy: IEEE.
- Feld, M., Burkhardt, F., & Müller, C. (2010, September). Automatic speaker age and gender recognition in the car for tailoring dialog and mobile services. In *Proceedings of the 11th Annual Conference of the International Speech Communication* (p. 2834-2837). Makuhari, Japan: ISCA.
- Feld, M., & Endres, C. (2010, November). Sharing user and context models in automotive HMI. In *Adjunct Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2010)* (p. 10). Pittsburgh, PA, United States.
- Feld, M., & Kahl, G. (2008, July). Integrated speaker classification for mobile shopping applications. In W. Nejdl, J. Kay, P. Pu, & E. Herder (Eds.), *Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2008)* (p. 288-291). Hannover, Germany: Springer.
- Feld, M., & Müller, C. (2008, August). Speaker classification for mobile devices. In *Proceedings of the 2nd IEEE International Interdisciplinary Conference on Portable Information Devices (Portable 2008)* (p. 1-5). Garmisch-Partenkirchen, Germany: IEEE.
- Feld, M., & Müller, C. (2009, June). An integrated development environment for speech-based classification. In *Proceedings of the 13th International Conference "Speech and Computer" (SPECOM 2009)* (p. 443-447). St. Petersburg, Russia.
- Feld, M., & Müller, C. (2011, November). The automotive ontology: Managing knowledge inside the vehicle and sharing it between cars. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications (Au-*

- tomotiveUI 2011*) (p. 1-8). Salzburg, Austria: ACM.
- Feld, M., Müller, C., & Schwartz, T. (2010, February). Second multimodal interfaces for automotive applications (MIAA 2010). In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI 2010)* (p. 441-442). Hong Kong, China: ACM.
- Feld, M., Schwartz, T., & Müller, C. (2010, November). This is me: Using ambient voice patterns for in-car positioning. In *Proceedings of the First Joint International Conference on Ambient Intelligence (AmI-10)* (p. 1-5). Malaga, Spain: Springer.
- Feldman, S. I. (1979). Make – a program for maintaining computer programs. *Software - Practice and Experience*, 9(4), 255-65.
- Friedland, G., & Vinyals, O. (2008, October). Live speaker identification in conversations. In *Proceeding of the 16th ACM International Conference on Multimedia (MM'08)* (p. 1017-1018). Vancouver, BC, Canada.
- Gajšek, R., Žibert, J., Justin, T., Štruc, V., Vesnicher, B., & Mihelič, F. (2010, September). Gender and affect recognition based on GMM and GMM-UBM modeling with relevance MAP estimation. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2810-2813). Makuhari, Japan: ISCA.
- Garofolo, J., et al. (1998). *Getting started with the DARPA TIMIT CD-ROM: An acoustic phonetic continuous speech database*. Gaithersburg, MD, United States: National Institute of Standards and Technology.
- Görlich, D. (2009). *Laufzeit-Adaption von Benutzungsschnittstellen für Ambient-Intelligence-Umgebungen mittels raumbasierter Nutzungsmodelle [Runtime adaptation of usage interfaces for ambient intelligence environments using spacial usage models]*. Doctoral dissertation, Department of Engineering und Technology, University of Kaiserslautern, Kaiserslautern, Germany.
- Gründl, M. (2005). *Fehler und Fehlverhalten als Ursache von Verkehrsunfällen und Konsequenzen für das Unfallvermeidungspotenzial und die Gestaltung von Fahrerassistenzsystemen [Faults and misconduct as cause of traffic accidents and consequences for the accident avoidance potential and the design of driver assistance systems]*. Doctoral dissertation, University of Regensburg.
- Hao, J. (2002). *Cross-racial studies of human vocal tract dimensions and formant structures*. Doctoral dissertation, Faculty of College of Health and Human Services, Ohio University.
- Harris, J. (2005, September). *Enter the ribbon*. Weblog entry. Available from <http://blogs.msdn.com/b/jensenh/archive/2005/09/14/467126.aspx>
- Heckmann, D. (2005). *Ubiquitous user modeling*. Doctoral dissertation, Department of Computer Science, Saarland University, Saarbrücken, Germany.
- Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., & Wilamowitz-Moellendorff, M. von. (2005). GUMO - the General User Model Ontology. In L. Ardissono, P. Brna, & A. Mitrovic (Eds.), *Proceedings of the 10th International Conference on User Modeling (UM 2005)* (p. 428-432). Springer Berlin / Heidelberg.

- Heerden, C. van, Barnard, E., Davel, M., Walt, C. van der, Dyk, E. van, Feld, M., et al. (2010, March). Combining regression and classification methods for improving automatic speaker age recognition. In *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'10)* (p. 1-4). Dallas, Texas, United States: IEEE.
- Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4), 1738-1752.
- Hermansky, H., & Sharma, S. (1998, November). TRAPS – classifiers of temporal patterns. In *Proceedings of the Fifth International Conference on Spoken Language Processing (ICSLP'98)* (p. 1-5). Sydney, Australia.
- Hill, D. R. (2007). Speaker Classification concepts: Past, present and future. In C. Müller (Ed.), *Speaker Classification I: Fundamentals, Features, and Methods* (Vol. 4343, p. 21-46). Springer Berlin / Heidelberg.
- Hu, J., Cheng, C., & Liu, W. (2006). Robust speaker's location detection in a vehicle environment using GMM models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(2), 403-412.
- Jameson, A. (2001, July). *Personalization for E-commerce*. Half-day tutorial presented at UM 2001, the Eighth International Conference on User Modeling, Sonthofen.
- Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in Kernel Methods – Support Vector Learning* (p. 169-184). MIT Press.
- Kenny, P., Ouellet, P., Dehak, N., Gupta, V., & Dumouchel, P. (2008, July). A study of interspeaker variability in speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5), 980-988.
- Kharroubi, J., Petrovska-Delacretaz, D., & Chollet, G. (2001, September). Combining GMM's with support vector machines for text-independent speaker verification. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH 2001 Scandinavia)* (p. 1761-1764). Aalborg, Denmark.
- Klepeis, N., Nelson, W., Ott, W., Robinson, J., Tsang, A., Switzer, P., et al. (2001). The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Analysis and Environmental Epidemiology*, 11(3), 231-252.
- Kleynhans, N. T., & Barnard, E. (2005, November). Language dependence in multilingual speaker verification. In *Proceedings of the 16th Annual Symposium of the Pattern Recognition Association of South Africa* (p. 117-122). Langebaan, South Africa.
- Kockmann, M., Burget, L., & Černocký, J. (2010, September). Brno university of technology system for interspeech 2010 paralinguistic challenge. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2822-2825). Makuhari, Japan.
- Krishnaswamy, S., Loke, S. W., Rakotonirainy, A., Horovitz, O., & Gaber, M. M. (2005, February). Towards situation-awareness and ubiquitous data mining for road safety:

- Rationale and architecture for a compelling application. In *Proceedings of the Conference on Intelligent Vehicles and Road Infrastructure* (p. 1-6). Melbourne, Australia.
- Ladefoged, P. (2006). *A course in phonetics* (5th international student ed.). Boston, MA: Thomson Wadsworth.
- Lasarczyk, E. (2010, September). Acoustics vs. articulation in articulatory speech synthesis: One vocal tract target configuration has more than one sound. In *Konferenz Elektronische Sprachsignalverarbeitung (ESSV'10) [Conference on Electronic Speech Signal Processing]* (p. 104-111). Berlin, Germany: TUDpress.
- Lasarczyk, E. (2012). *Machine vs. human: A cross-discipline study on synthetic speaker age recognition*. (Publication forthcoming as part of doctoral dissertation)
- Leeuwen, D. A. van, & Brümmer, N. (2007). An introduction to application-independent evaluation of speaker recognition systems. In C. Müller (Ed.), *Speaker Classification I: Fundamentals, Features, and Methods* (Vol. 4343, p. 330-353). Springer Berlin / Heidelberg.
- Levelt, W. J. M. (1991). *Speaking: From intention to articulation* (Second printing ed.; A. Joshi, Ed.). The MIT Press Cambridge / London.
- Levelt, W. J. M. (1999, June). Models of word production. *Trends in Cognitive Sciences*, 3(6), 223-232.
- Li, M., Jung, C.-S., & Han, K. J. (2010, September). Combining five acoustic level modeling methods for automatic speaker age and gender recognition. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTER-SPEECH 2010)* (p. 2826-2829). Makuhari, Japan: ISCA.
- Lingenfelser, F., Wagner, J., Vogt, T., Kim, J., & André, E. (2010, September). Age and gender classification from speech using decision level fusion and ensemble based techniques. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2798-2801). Makuhari, Japan: ISCA.
- Mackay, W. E. (1991). Triggers and barriers to customizing software. In S. P. Robertson, G. M. Olson, & J. S. Olson (Eds.), *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)* (p. 153-160). New York: ACM.
- Martin, A. F. (2007). Evaluations of automatic speaker classification systems. In C. Müller (Ed.), *Speaker Classification I: Fundamentals, Features, and Methods* (Vol. 4343, p. 330-353). Springer Berlin / Heidelberg.
- Martin, A. F., Doddington, G. R., Kamm, T., Ordowski, M., & Przybocki, M. A. (1997, September). The DET curve in assessment of detection task performance. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EUROSPEECH-1997)* (p. 1895-1898). Rhodes, Greece: ISCA.
- Mattes, S. (2003). The lane change task as a tool for driver distraction evaluation. In H. Strasser, H. Rausch, & H. Bubbs (Eds.), *Quality of Work and Products in Enterprises of the Future* (p. 57-60). Stuttgart: Ergonomia.
- Meinedo, H., & Trancoso, I. (2010, September). Age and gender classification using fusion

- of acoustic and prosodic features. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2818-2821). Makuhari, Japan: ISCA.
- Meinedo, H., & Trancoso, I. (2011, August). Age and gender detection in the I-DASH project. *ACM Transactions on Speech and Language Processing*, 7(4), 13:1-13:16.
- Meixner, G., Seissler, M., & Breiner, K. (2011). Model-driven Useware Engineering. In H. Hussmann, G. Meixner, & D. Zühlke (Eds.), *Model-Driven Development of Advanced User Interfaces* (Vol. 340, p. 1-27). Springer Berlin / Heidelberg. Available from http://dx.doi.org/10.1007/978-3-642-14562-9_1
- Mendoza, L. A. F., Cataldo, E., Vellasco, M., & Silva, M. (2010, September). Classification of voice aging using parameters extracted from the glottal signal. In K. Diamantaras, W. Duch, & L. S. Iliadis (Eds.), *Proceedings of the 20th International Conference on Artificial Neural Networks (ICANN'10): Part III* (p. 149-156). Thessaloniki, Greece: Springer Berlin / Heidelberg.
- Metze, F., Ajmera, J., Englert, R., Bub, U., Burkhardt, F., Stegmann, J., et al. (2007). Comparison of four approaches to age and gender recognition for telephone applications. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'07)* (p. 1089-1092). Honolulu, Hawaii: IEEE.
- Minematsu, N., Yamauchi, K., & Hirose, K. (2003, September). Automatic estimation of perceptual age using speaker modeling techniques. In *Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH 2003 - INTERSPEECH 2003)* (p. 3005-3008). Geneva, Switzerland: ISCA.
- Müller, C. (2005). *Zweistufige kontextsensitive Sprecherklassifikation am Beispiel von Alter und Geschlecht [Two-layered context-sensitive speaker classification on the example of age and gender]*. Doctoral dissertation, Computer Science Institute, University of the Saarland, Saarbrücken, Germany.
- Müller, C. (Ed.). (2007). *Speaker Classification I: Fundamentals, features, and methods* (Vol. 4343). Springer Berlin / Heidelberg.
- Müller, C., Biel, J.-I., Kim, E., & Rosario, D. (2008). Speech-overlapped acoustic event detection for automotive applications. In *Ninth Annual Conference of the International Speech Communication Association INTERSPEECH-08* (p. 2590-2593).
- Müller, C., & Feld, M. (2006). Towards a multilingual approach on Speaker Classification. In *Proceedings of the 11th International Conference "Speech and Computer" (SPECOM 2006)* (p. 120-124). St. Petersburg, Russia: Anatolya Publishers.
- Müller, C., & Wittig, F. (2003). Speech as a source for ubiquitous user modeling. In *Proceedings of the Workshop on User Modeling in Ubiquitous Computing in conjunction with the Ninth International Conference on User Modeling (UM 2003)* (p. 46-50). Pittsburgh, USA.
- Nasoz, F., Lisetti, C. L., & Vasilakos, A. V. (2010, October). Affectively intelligent and adaptive car interfaces. *Information Sciences*, 180, 3817-3836.
- Nguyen, P., Le, T., Tran, D., Huang, X., & Sharma, D. (2010, September). Fuzzy support

- vector machines for age and gender classification. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2806-2809). Makuhari, Japan: ISCA.
- NHTS BRIEF – National Household Travel Survey. (2006, November). Online report. Available from <http://nhts.ornl.gov/briefs/Commuting%20for%20Life.pdf>
- NIST Language Recognition Evaluation. (2011). Workshop Series. Available from <http://www.nist.gov/itl/iad/mig/lre11.cfm>
- Oliver, N., & Pentland, A. P. (2000, June). Driver behavior recognition and prediction in a smartcar. In J. G. Verly (Ed.), *Proceedings of the SPIE Conference on Enhanced and Synthetic Vision* (Vol. 4023, p. 280-290). Orlando, FL, United States.
- Pechmann, T., & Habel, C. (2004). *Multidisciplinary approaches to language production* (Vol. 157; W. Bisang, H. H. Hock, & W. Winter, Eds.). Mouton de Gruyter Berlin / New York.
- Pfleger, N., & Schehl, J. (2006, September). Development of advanced dialog systems with PATE. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech 2006 - ICSLP)* (p. 1778-1781). Pittsburgh, Pennsylvania, United States: ISCA.
- Plichta, B. (2002). Best practices in the acquisition, processing, and analysis of acoustic speech signals. *Working Papers in Linguistics*, 8(3), 209-217.
- Porat, R., Lange, D., & Zigel, Y. (2010, September). Age recognition based on speech signals using weights supervector. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2814-2817). Makuhari, Japan: ISCA.
- Pulakka, H. (2005). *Analysis of human voice production using inverse filtering, high-speed imaging, and electroglottography*. Master's thesis, Helsinki University of Technology, Espoo.
- Reynolds, D. A., Campbell, J. P., Campbell, W. M., Dunn, B., Gleason, T., Jones, D., et al. (2003, December). Beyond cepstra: Exploiting high-level information in speaker recognition. In *Proceedings of the Workshop on Multimodal User Authentication (MMUA)* (p. 223-229). Santa Barbara, CA, United States.
- Reynolds, D. A., Quatieri, T. F., & Dunn, R. B. (2000). Speaker verification using adapted Gaussian mixture models. *Digital Signal Processing*, 10(1-3), 19-41.
- Reynolds, D. A., & Rose, R. C. (1995). Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1), 72-83.
- Reynolds, D. A., & Torres-Carrasquillo, P. (2005, March). Approaches and applications of audio diarization. In *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)* (Vol. 5, p. 1-4). Philadelphia, PA, United States.
- Ritthoff, O., Klinkenberg, R., Fischer, S., Mierswa, I., & Felske, S. (2001, October). YALE: Yet Another Machine Learning Environment. In R. Klinkenberg et al. (Eds.), *LLWA*

- 01 – Tagungsband der GI-Workshop-Woche Lernen – Lehren – Wissen – Adaptivität (p. 84-92). Dortmund, Germany.
- Road vehicles – ergonomic aspects of transport information and control systems – simulated lane change test to assess in-vehicle secondary task demand* (No. ISO 15006:2011). (2010).
- Rothenberg, M. (1973). A new inverse-filtering technique for deriving the glottal air flow waveform during voicing. *The Journal of the Acoustical Society of America*, 53(6), 1632-1645.
- Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1), 43-49.
- Schank, R. C., & Colby, K. M. (Eds.). (1973). *Computer models of thought and language*. New York, NY, United States: W. H. Freeman & Co.
- Schiel, F. (1998). Speech and speech-related resources at BAS. In *Proceedings of the First International Conference on Language Resources and Evaluation* (p. 343-349). Granada, Spain.
- Schiele, B., Andriluka, M., Majer, N., Roth, S., & Wojek, C. (2009, May). Visual people detection: Different models, comparison and discussion. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking* (p. 1-8). Kobe, Japan: IEEE.
- Schötz, S. (2004). Prosodic cues in human and machine estimation of female and male speaker age. In *Nordic Prosody: Proceedings of the IXth Conference* (p. 215-223). Lund, Sweden.
- Schötz, S. (2006). *Perception, analysis and synthesis of speaker age*. Doctoral dissertation, Department of Linguistics and Phonetics, Lund University, Lund, Sweden.
- Schötz, S. (2007). Acoustic analysis of adult speaker age. In C. Müller (Ed.), *Speaker Classification I: Fundamentals, Features, and Methods* (Vol. 4343, p. 88-107). Springer Berlin / Heidelberg.
- Schröder, M., Charfuelan, M., Pammi, S., & Steiner, I. (2011, August). Open source voice creation toolkit for the MARY TTS platform. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)* (p. 3253-3256). Florence, Italy: ISCA.
- Schuller, B., Batliner, A., Steidl, S., & Seppi, D. (2011, November). Recognising realistic emotions and affect in speech: state of the art and lessons learnt from the first challenge. *Speech Communication*, 53(9-10), 1062-1087.
- Schuller, B., Steidl, S., Batliner, A., Burkhardt, F., Devillers, L., Müller, C., et al. (2010, September). The INTERSPEECH 2010 Paralinguistic Challenge. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (p. 2794-2797). Makuhari, Japan: ISCA.
- Schultz, T. (2007). Speaker characteristics. In C. Müller (Ed.), *Classification I: Fundamentals, Features, and Methods* (Vol. 4343, p. 47-74). Springer Berlin / Heidelberg. Available from http://dx.doi.org/10.1007/978-3-540-74200-5_3

- Schultz, T., & Waibel, A. (2001, August). Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35, 31-51.
- Schwartz, T., Stahl, C., Baus, J., & Wahlster, W. (2010). Seamless resource-adaptive navigation. In M. W. Crocker & J. Siekmann (Eds.), *Resource-Adaptive Cognitive Processes* (p. 239-265). Springer Berlin / Heidelberg.
- Schwarz, J., et al. (2006, October). *Code of practice for the design and evaluation of ADAS*. Project deliverable in the Preventive and Active Safety Applications Integrated Project (PREVENT). Available from http://www.prevent-ip.org/download/deliverables/RESPONSE3/D11.2/Response3_CoP_v3.0.pdf
- Sedaaghi, M. H. (2009). A comparative study of gender and age classification in speech signals. *Iranian Journal of Electrical & Electronic Engineering*, 5(1), 1-12.
- Shami, M., & Verhelst, W. (2007). Automatic classification of expressiveness in speech: A multi-corpus study. In C. Müller (Ed.), *Speaker Classification II: Selected Projects* (Vol. 4343, p. 43-56). Springer Berlin / Heidelberg.
- Shriberg, E. (2007). Higher-level features in speaker recognition. In C. Müller (Ed.), *Speaker Classification I: Fundamentals, Features, and Methods* (Vol. 4343, p. 241-259). Springer Berlin / Heidelberg.
- Sorin, A., Ramabadran, T., Chazan, D., Hoory, R., McLaughlin, M., Pearce, D., et al. (2004, May). The ETSI extended distributed speech recognition (DSR) standards: client side processing and tonal language recognition evaluation. In *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)* (Vol. 1, p. 129-132). Montreal, Canada: IEEE.
- Sundar, S. S., & Marathe, S. S. (2010). Personalization versus customization: The importance of agency, privacy, and power usage. *Human Communication Research*, 36(3), 298-322.
- Trautmüller, H., & Eriksson, A. (1994). *The frequency range of the voice fundamental in the speech of male and female adults* (Online Manuscript). Department of Linguistics, University of Stockholm. Online Manuscript. Available from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.64.5133>
- Tseng, M., Jiao, R., & Wang, C. (2010). Design for mass personalization. *CIRP Annals – Manufacturing Technology*, 59(1), 175-178.
- Ussat, C. (2010, Februar). Personalisierte Optionsangebote im Fahrzeugnavigationssystem: Eine qualitative Studie zur Ermittlung des Kontextes und möglicher Assistenzarten [Personalized optional offers in the vehicle navigation system: A qualitative study to determine the context of possible types of assistance]. In M. Bick et al. (Eds.), *Proceedings of the 5th Conference on Mobile and Ubiquitary Information Systems (MMS 2010)* (p. 43-56). Göttingen: GI.
- Vashitz, G., Shinar, D., & Blum, Y. (2008). In-vehicle information systems to improve traffic safety in road tunnels. *Transportation Research Part F: Traffic Psychology and Behaviour*, 11(1), 61-74.
- Wada, T., Shinozaki, T., & Furui, S. (2010, December). Investigations of features and estimators for speech-based age estimation. In *Proceedings of the Second APSIPA Annual*

- Summit and Conference* (p. 470-473). Biopolis, Singapore.
- Wahlster, W., Feld, M., Gebhard, P., Heckmann, D., Jung, R., Kruppa, M., et al. (2010). The shopping experience of tomorrow: Human-centered and resource-adaptive. In M. W. Crocker & J. Siekmann (Eds.), *Resource-Adaptive Cognitive Processes* (p. 205-237). Springer Berlin / Heidelberg.
- Wahlster, W., & Kobsa, A. (1989). User models in dialog systems. In W. Wahlster & A. Kobsa (Eds.), *User Models in Dialog Systems* (p. 4-34). Springer Berlin.
- Wasinger, R. (2006). *Multimodal interaction with mobile devices: Fusing a broad spectrum of modality combinations*. Doctoral dissertation, Department of Computer Science, Saarland University, Saarbrücken, Germany.
- Wasinger, R., Stahl, C., & Krüger, A. (2003, September). Robust speech interaction in a mobile environment through the use of multiple and different media input types. In *Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH-2003)* (p. 1049-1052). Geneva, Switzerland: ISCA.
- Weiser, M. (1991, September). The computer for the 21st century. *Scientific American*, 265(3), 94-104.
- Wiedenbeck, S. (1999). The use of icons and labels in an end user application program: an empirical study of learning and retention. *Behaviour & Information Technology*, 18(2), 68-82.
- Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques* (2nd ed.). San Francisco: Morgan Kaufmann.
- Wooters, C., Fung, J., Peskin, B., & Anguera, X. (2004). Towards robust speaker segmentation: The ICSI-SRI fall 2004 diarization system. In *Proceedings of the NIST Rich Transcription Fall 2004 Evaluation Workshop* (p. 1-6).
- Wu, K., & Childers, D. G. (1991). Gender recognition from speech. part I: Coarse analysis. *Journal of the Acoustical Society of America*, 90(4), 1828-1840.
- Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., & Woodland, P. (1999). *The HTK Book*. Cambridge: Entropic Ltd.
- Yumoto, E., Gould, W., & Baer, T. (1982). Harmonics-to-noise ratio as an index of the degree of hoarseness. *Journal of the Acoustical Society of America*, 71(6), 1544-1550.
- Zühlke, D. (2004). *Useware-Engineering für technische Systeme [Useware engineering for technical systems]*. Springer Berlin / Heidelberg / New York.