NTNU

Norwegian University of
Science and Technology

Open Set Speaker Identification

Jørgen Johan Antonsen

Master of Science in Electronics
Submission date: June 2017
Supervisor: Torbjørn Svendsen, IES
Co-supervisor: Øystein Birkenes, Cisco Systems Norway

Norwegian University of Science and Technology
Department of Electronic Systems

I would like to thank my supervisor Dr. Øystein Birkenes and Professor Torbjørn Svendsen for weekly guidance and help with the task. I am also very grateful for the support and encouragement of my family. Finally I would like to thank PhD students Abdolreza Sabzi Shahrebabaki and Negar Olfati for help with getting into Kaldi.

Summary

Digital assistants that communicate through speech are one of the new technologies that have emerged this decade. Progress in the field of speaker recognition have opened up possibilities for having digital assistants for groups of people, where the assistant can offer personalized assistance and receive commands from multiple people. This master's thesis investigates techniques for speaker identification in a group meeting scenario, where the availability of speech data for system training often can be low. Speaker identification and verification experiments on the RSR2015 database have been conducted with different GMM-UBM- and i-vector-based systems. It has been found that the tz-normalized GMM-UBM system gave best the performance, with a recognition rate of 81.3% and an EER of 7.8%. The GMM-UBM system has overall performed better that the i-vector system.

Recent research proposes the usage of deep learning techniques for speaker identification, and a framework for bottleneck feature extraction have been included in thesis, with experiments on bottleneck features left for future work. In addition to experiments, the thesis also contains a short guide to setting up a speaker identification system in SIDEKIT, which has been the main toolkit used in this task. The full implementation of the scripts used in experiments can be found in the Appendix.

Sammendrag

Digitale assistenter som kommuniserer gjennom tale er en av nye teknologier av dette tiåret. Fremgang i forskning på taleridentifikasjon åpner opp muligheter for felles digitale assistenter for flere brukere, der assistenten tilbyr personlig assistanse og mottar kommandoer fra flere personer. Denne masteroppgaven undersøker teknikker for taleridentifikasjon i et møtescenario, der tilgjengeligheten på taledata for trening av et system ofte er lav. Taleridentifikasjons- og talerverifikasjonseksperimenter på RSR2015-databasen har blitt utført med forskjellige GMM-UBM- og i-vektor-baserte systemer. Det tz-normaliserte GMM-UBM-systemet har gitt best resultater, med gjenkjenningsrate på 81.3% og EER lik 7.8%. De GMM-UBM-baserte systemene har jevnt over gitt bedre resultater enn de i-vektor-baserte systemene.

Nylig forskning foreslår bruk av teknikker basert på dyp læring. Denne oppgaven inkluderer derfor et rammeverk for bottleneck feature extraction, der ekspermenter overlates til fremtidig arbeid. I tillegg til eksperimenter inkluderer også oppgaven en guide til oppsett av et taleridentifikasjonssystem i SIDEKIT, som har vært hovedverktøyet brukt i oppgaven. En full implementasjon av alle skript finnes i Appendix.

Table of Contents

List of Figures

Abbreviations

CMN	=	Cepstral mean normalization
CMVN	=	Cepstral mean and variance normalization
CMS	=	Cepstral mean subtraction
DCF	=	Detection cost function
DCT	=	Discrete cosine transform
DET	=	Detection error tradeoff
DFT	=	Discrete Fourier transform
DNN	=	Deep neural network
EER	=	Equal-error-rate
FA	=	False acceptance
FAR	=	False acceptance rate
FR	=	False rejection
FRR	=	False rejection rate
I-vector	=	Identity vector
ID	=	Identity
GMM	=	Gaussian mixture model
HMM	=	Hidden Markov model
JFA	=	Joint factor analysis
LDA	=	Linear discriminative analysis
LDC	=	Linguistic Data Consortium
LP	=	Linear prediction
LPC	=	Linear predictive coding
LPCC	=	Linear predictive cepstral coefficients
MAP	=	Maximum a posteriori
MFCC	=	Mel-frequency cepstral coefficients
minDCF	=	Minimum detection cost function
MIT	=	Massachusetts Institute of Technology
NIST	=	National Institute of Standards in Technology
PLDA	=	Probabilistic linear discriminative analysis
PLP	=	Perceptual linear prediction
RASTA	=	Relative spectral
RBM	=	Restricted Boltzmann machine
SNR	=	Signal-to-noise ratio
SRE16	=	NIST Speaker Recognition Evaluation 2016
SVM	=	Support vector machine
TI	=	Texas Instruments
TV	=	Total variability
UBM	=	Universal background model
VAD	=	Voice activity detection

Chapter 1

Introduction

From the digital revolution that started around the beginning of this millennium, new technologies have transformed from digital tools into companions. The user interface has gone from being buttons and knobs that require the study of thick manuals before use, to intuitive symbols and touch-sensitive pads that even a 3-year can master effortlessly. And as the world has opened its eyes for artificial intelligence and its wide possibilities, the new interface will be interaction with devices through other ways than just touch is possible. Today most smartphone and tablet operating systems comes with digital assistants that communicate through speech commands.

Another natural step in the humanization of user interfaces is the usage of voice biometrics for e.g. granting system access. The idea of using biometrics for identification consists of measuring some physiological trait that is assumed to be unique for each individual, and associate the data with the individual involved. Examples of biometrics in application are fingerprints, retina scanning, voice recognition, face recognition and DNA analysis. Many of the latest smartphones include identification by fingerprint, face recognition and retina scanning, and it might seem as there is a race now in the relatively young market of smart speakers, where functions of the house can be controlled by voice commands, that now also allows for personal assistance by recognizing the speaker.

On behalf of Cisco Systems Norway AS, this thesis has been written as an investigation in preparement for a speaker identification system for use in a video conference scenario, where the system, from commands to the systems personal assistant, should identify the speaker giving the command at any time. More specifically, the thesis will give an overview the current state-of-the-art speaker identification systems and the theory behind, including recent promising approaches based on deep learning.

The structure of this report is as follows: Chapter 2 presents background theory relevant the work done in this thesis. Chapter 3 gives an overview of how to set up a speaker identification experiment in Sidekit. Chapter 4 explains the experimental protocol, presents and discusses results. Chapter 5 concludes the thesis with a closing discussion and propositions for future work.

[27] [2]

Chapter 2

Theory

Throughout the history of the field of speaker identification, the research community has moved through several paradigms defined by the changing state-of-the-art models for speaker recognition, influenced by research within the field and from other fields like speech recognition, in addition to technological advancements and the growth of available resources and demand. Early approaches to speaker recognition involved human inspection of the speech spectrum (see Section 2.6). Today, the automatic approach, i.e. speaker recognition with computers, have been shown to outbeat humans [28]. The error rates of the automatic approach continues to decrease, but there are still many challenges when the available data is sparse, or the quality of the data is low - requiring further innovation.

This chapter will define automatic speaker recognition and its sub-categories, and present theory relevant to today's speaker identification systems, as well as the systems themselves.

2.1 Speaker recognition

Automatic speaker recognition can be defined as the task of inferring a persons identity based on analysis of recorded speech, in which the analysis is done by a computer. The field relies on the fact that a speech signal contains a wide range of information besides just words, such as information about speaker identity, gender, age group, mood, social setting of the recording (e.g. formal speech vs. casual conversation) and recording environment. As will be presented further on in this chapter, filtering out the *unimportant* information in a speech signal is a crucial part of the speaker identification task.

Speaker recognition is an umbrella term that is mainly divided into the fields of speaker verification and speaker identification (although there are several sub-fields such as speaker classification and speaker diarization). Speaker verification, or speaker authentication, involves only one speaker, and consists of accepting/rejecting an utterance after some means of comparison with a model of the claimed identity of the speaker. Speaker identification, on the other hand, involves a group of speakers, where the task is to decide from which of the speakers a given utterance comes from.

Further, the recognition task can be both text-dependent and text-independent, the first in which the system requires a specific word or phrase to do the identification task, while the second is modeled to work regardless of the phonetic content of the utterance. Because of the phonetic constraints, the text-dependent systems generally gives higher recognition rates, and requires less training data to model the acoustic space. This comes at the cost of restricted speech content, which might not be suitable for applications that does not involve controlled situations with text prompts. Probably for the sake of generalizability, the majority of research within speaker recognition focuses on text-independent systems.

Finally, a speaker identification task can be either *closed set* or *open set*. In the closed set scenario, it is guaranteed that the utterance in question comes from one of the speakers known to the speaker system. In the case of an open set, the utterance might also come from a speaker *unknown* to the system, requiring some method of rejection so as not to erroneously classify the utterance to come from one of the known speakers.

2.2 An automatic speaker identification framework

Before going into technical details, we will look at a general speaker identification system and introduce some terminology. As mentioned in the previous section, speaker identification distinguishes itself from speaker verification by involving sets of speakers instead of just one speaker. This increases the difficulty as the task goes from simply accepting or rejecting an utterance to comparing scores from all speakers in a set, identifying the original speaker, or in the case of open set identification, determining whether the utterance originates from a known speaker at all.

Figure 2.1 illustrates the toolchain of an open set speaker identification task in a video conference meeting scenario. An utterance from a known or unknown speaker is recorded by a microphone, then sampled and preprocessed before task-relevant features are extracted. The extracted features are then modified, enabling comparison techniques or *scoring* with existing speaker models. Scores from all speaker models are then compared by a decision function, determining 1) whether the utterance comes from one of the known speakers or not, and 2) if it does, which of the speakers it belongs to. As will be shown, tasks 1) and 2) can be solved simultaneously by adding a *complementary model* that represents all speakers outside the set of known speakers.

To build individual speaker models, one needs speech data from each speaker that is to be known by the system. The set of known speakers are in speaker recognition known as the *enrollment speakers*, and a speaker is enrolled into the system when *enrollment data* from the speaker is processed to build its model. After the enrollment process, the performance of the speaker recognition system can be evaluated using *test data*, which, in an open set scenario will consist of data from speakers in and outside the enrollment set. The set of all speakers involved in testing the system will be referred to as the *test speakers*.

Training independent models is in the state-of-the-art speaker recognition systems acheived through initially training a general speech model known as the universal background model (UBM), then using speaker adaptation techniques to train the enrollment speaker models. In speaker recognition, *training data* refers to the data used to train the general UBM. Model training and adaptation will be further explained in Section 2.8.

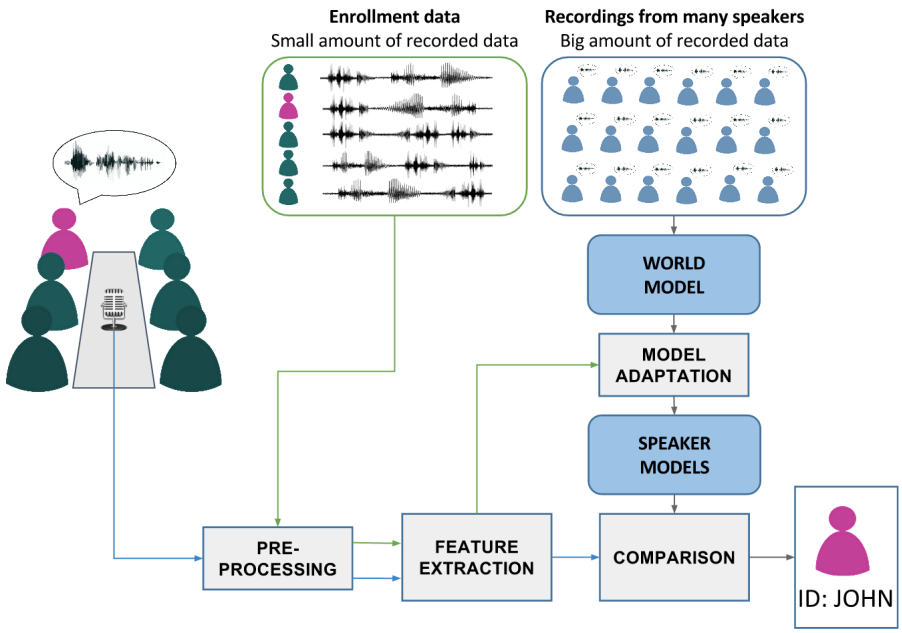

Figure 2.1: The toolchain of open set speaker identification. Preprocessing, feature extraction and training of world model omitted from figure.

2.3 Statistical background

This section provides a statistical background for the later material in this chapter.

2.3.1 Gaussian mixture model

The Gaussian or normal distribution is a probability distribution function of a random variable X following Equation 2.1, where μ and σ^2 is the mean and variance of the distribution.

$$P(X = x) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

If $\mathbf{X}$ is an array of length N, X is modeled by the *multivariate* Gaussian distribution, that has the form as in Equation 2.2, where μ and $\mathbf{\Sigma}$ is the mean vector and covariance matrix of the distribution, respectively.

$$p(\mathbf{x}|\mu, \mathbf{\Sigma}) = \mathcal{MN}(\mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{\frac{1}{2}}|\mathbf{\Sigma}|^2} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)} \tag{2.2}$$

An array $\mathbf{x}$ of length N can be modeled as a weighted sum of M N-dimensional Gaussian distributions, as shown in Equation 2.3. This known as a Gaussian mixture model (GMM). The GMM can model N-dimensional data of various character by changing its

mean, covariances and weights $\mathbf{a} = [a_1, a_2, \ldots, a_M]$. The weights are constrained by $\sum_i a_i = 1$. The *model* of $\mathbf{x}$ is denoted as $\lambda = (\mathbf{a}, \mu, \mathbf{\Sigma})$, μ and $\mathbf{\Sigma}$ being the mean vector and the covariance matrix, respectively. The GMM can then be written as

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} a_i p(\mathbf{x}|\lambda_i), \tag{2.3}$$

λ_i indicating the mixture parameters for mixture *i*.

Our goal is to infer a λ that maximizes the likelihood of a given feature vector x, that is,

$$\underset{\lambda}{\operatorname{argmax}}\, p(\lambda|\mathbf{x}), \tag{2.4}$$

which is equal to

$$\underset{\lambda}{\operatorname{argmax}}\, \frac{p(\mathbf{x}|\lambda)p(\lambda)}{p(\mathbf{x})} = \underset{\lambda}{\operatorname{argmax}}\, p(\mathbf{x}|\lambda)p(\lambda), \tag{2.5}$$

applying Bayes' theorem. Further applying the logarithm to Equation 2.5, this gives us the *maximized log-likelihood*, and reduces to

$$\underset{\lambda}{\operatorname{argmax}}\, log\ p(\lambda|\mathbf{x}) \tag{2.6}$$

.

Now, Equation 2.6 can be numerically solved using the *expectation-maximization* (EM) algorithm [11], which until convergence iterates between calculating the log-likelihood of Equation 2.3, i.e. $log\ p(\mathbf{x}|\lambda)$ (E-step), and then, given new values of $\mathbf{x}$, maximizing the log-likelihood of λ through Equation 2.6 (M-step).

2.3.2 Neural networks

The artifical neuron, or *perceptron*, is a simplified model of the neurons in the human brain. In the traditional model, shown in Figure 2.2, it is evoked to produce an output y if the sum of its weighted inputs $\boldsymbol{x} \cdot \boldsymbol{w}$ exceeds a certain threshold. However, for mathematical convenience, in current models the binary *activation function* is replaced with an approximating smooth function.

Perceptrons can be organized in layers to make multilayer perceptrons, or simply, neural networks. Neural networks are defined as *deep* if they have more than two *hidden* layers. That is, the input layer, three or more *hidden* layers followed by an output layer. The network can be trained for different tasks using labeled training data of any form, as long as the training data can be represented as an array of values to go into the input layer, and labeled with appropriate labeling values on the output layer. The network is then trained by maximizing the probability of the desired output of each layer with respect to the preceding weights. This process starts at the output layer, and is then repeated for the preceding layer with its desired output.

The number of layers and the number of neurons in each layer can be adjusted according to the application of the network. If the number of neurons and layers is too small, it might not be able to generalize to the content of the training data. If the number is too big

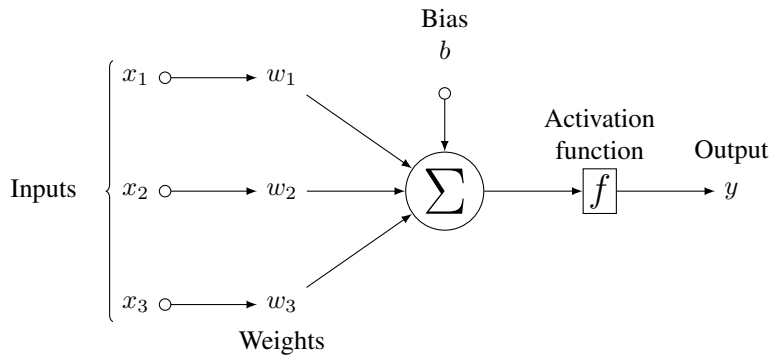

Figure 2.2: An artificial model of a neuron. Modified from `http://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network/132471`

there is a risk of overfitting to the training data, making the network too dependent on the specific character of the input data, not recognizing test input with the same label, but of a varying character.

2.4 Sources of variability in speech

No matter how sophisticated techniques for signal acquisition or speaker modeling can be, we cannot avoid the fact that we are dealing with random signals from the real world, which comes with noise and acoustical environments of numerous characteristics. Although the speaker recognition system should be designed for ideal performance, the quality and variability of the speech data will unavoidably affect how well the system performs. Thus, a robust system for speaker recognition must take all of these variabilities into account.

Hansen [17] divides sources of speaker variability in speaker recognition into three categories related to the physical characteristics of the speaker, the recording environment and equipment (i.e. variability related to technology), and to the conversation setting. Physical characteristics is related with the shape of the mouth, nasal region and the vocal tract, which will vary from speaker to speaker, depending on age, gender and health condition. Examples of technology-based factors are choice of microphone, background and channel noise during recording, bit-depth and sampling frequency. Finally, a speaker will behave differently in different conversational settings, and this also affects how the recorded speech sounds. Examples of different settings are one-to-one vs. group conversations, degree of familiarity between speakers and read text vs. speech without a script. All of these sources of variability will affect the design of the speaker recognition system. This includes the choice of speech features and the design of speaker models, which must be done in such a way that the erroneous performance of the system is minimized.

2.5 Sampling and preprocessing speech

This section explains the commons steps from recording speech, applying preprocessing techniques, to finally having a digital speech signal ready for feature extraction.

2.5.1 Sampling

As is commonly known, speech is recorded using microphones, devices that convert changes in air pressure to a continuous electric signal $x(t)$, which is then sampled at an interval T, known as the *sample rate*, producing a time-discrete signal $\tilde{x}[n] = x(nT)$. The inversion of T, $f = 1/T$, is called *sampling frequency*, and must be chosen sufficiently high as to capture the desired level of accuracy, given the bandwidth of $x(t)$, as well as computational and storage restrictions. A higher value of f gives more speaker discriminative information, but speaker recognition based on fundamental frequency only is also possible [13], and the current standard large datasets used in speaker recognition research are band-limited to 8 kHz bandwidth [16].

The time-discrete samples $\tilde{x}[n]$ are further quantized through some quantization function q that maps $\tilde{x}$ to discrete amplitude values. The process is summarized in Figure 2.3. We end up with a signal $x[n]$ that is discrete in time and amplitude, which is passed on to the preprocessing stages presented in the following subsections.

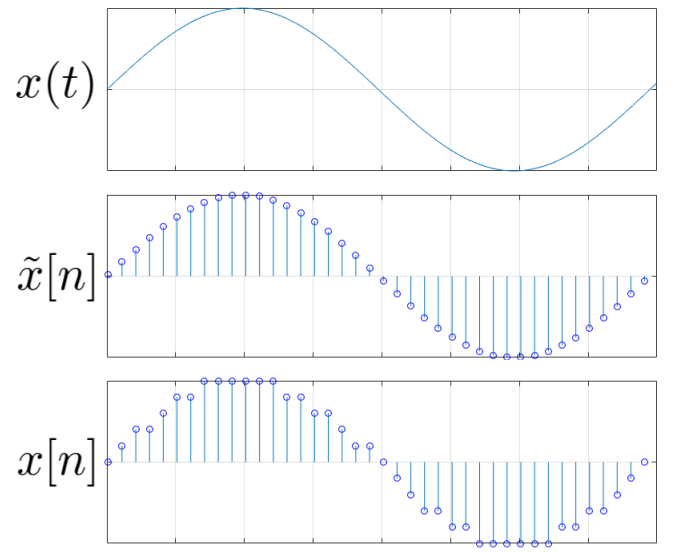

Figure 2.3: Recording and digitalizing a sine signal $x(t)$.

2.5.2 Pre-emphasis filtering

To normalize the spectral distribution of energy in speech, it is common to apply pre-emphasis filtering before further analysis. This emphasizes the high-frequency content of

the speech, which may conceal valuable speaker-discriminative information. Equation 2.7 gives a commonly used pre-emphasis filter of a signal $x[n]$, with common values for α being 0.95 and 0.97.

$$\tilde{x}[n] = x[n] - \alpha x[n-1] \tag{2.7}$$

2.5.3 Framing and windowing

Speech recordings in their unprocessed form are nonstationary processes that varies in both phonetic content and length, which makes the task of acoustic modeling and comparison a challenging task. To allow for analysis and modeling it is in speech processing common to view the sampled speech in short frames in time, making us able to assume stationarity. The length of these frames is usually in the range of 20 to 30 ms, and they are usually shifted 10 ms in time to take account for phone of short duration and uneven segmentations over transitions between phones. Each frame is then multiplied with a window function, which smoothens the overlap between neighbouring frames. Commonly used window functions include Hamming, Hanning, Blackman, triangular window, Bartlett, Welch and Gauss [2, p. 162]. Figure 2.4 illustrates the process of framing and windowing of a speech signal.

2.5.4 Voice-activity detection

Voice-activity detection (VAD) is a common step in the speaker recognition framework, as it removes unwanted parts of recorded speech, reducing the overall computational cost of the system, as well as eliminating irrelevant data. [2, p. 562] estimates as much as 30 % of speech in a "normal" speech recording to consist of silence. It however important to note that the silent parts of a speech recording do contain information about channel, that can be utilized for normalization purposes, not to mention the pattern of concurring silence and speech activity varies between speakers, and thus may be utilized for speaker recognition purposes.

A simple and effective VAD approach called *energy-based* VAD involves thresholding the energy of the speech content, discarding the speech frames energy lower than the set threshold as silence. A similar approach is the signal-to-noise rate (SNR)-based VAD, where frames with an SNR lower than a certain level is discarded. An alternative way, common in other fields of speech processing, is to model silence as a phone and detect it by using speech recognition techniques. Other rather recently proposed approaches to VAD include SVM- and neural network-based techniques [46].

2.6 Feature extraction

A robust speaker identification system requires an appropriate representation of the speech data, which ideally should be as unique for each respective speaker as possible. A fitting analogy from another subfield of biometrics would be the fingerprint, and indeed, the *voiceprint*[1], the spectogram of a speech segment, was in the early years of speaker

[1] 'The term is indeed constructed from the words *voice* and *fingerprint* [33]'

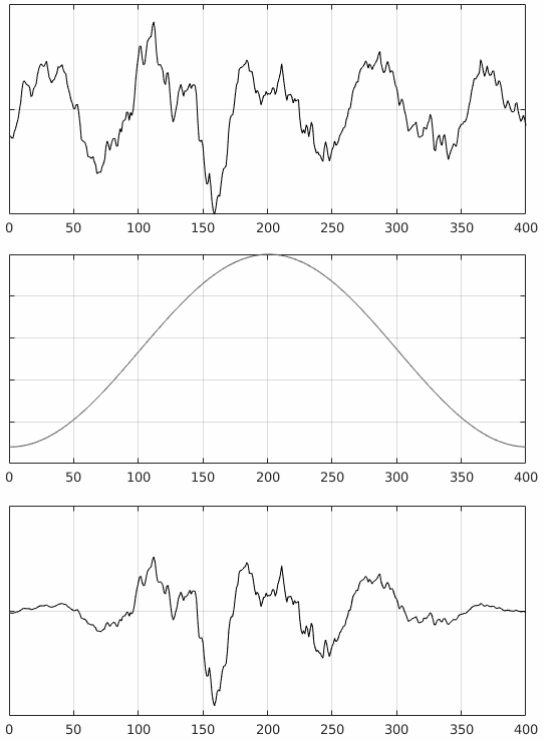

Figure 2.4: Framing and windowing of a speech signal. a) 25 ms frame of signal, b) The Hamming function, c) Signal in a) after windowing.

recognition seen as a promising feature, as experiments involving human inspection of the voiceprint gave good results [20]. The voiceprint has however proved to give poor results in changing acoustical environments (*sessions*) [27, p. 1465], thus some alternative enhancement of the speech is necessary to extract the speaker discriminative information. The following subsections present the most common and some recently probosed methods of feature extraction.

2.6.1 Mel-frequency cepstral coefficients

Originally developed for speech recognition, mel-cepstral frequency coefficients (MFCC) [8, 30] are the most widely used features in speaker recognition today. Computing MFCCs begins with obtaining the spectrum X_s of a speech segment x through the discrete Fourier transform (DFT),

$$X(f) = \sum_{n=0}^{N-1} x_n e^{-2\pi kn/N}, \tag{2.8}$$

and from this obtaining the *power spectral density*,

$$X_s(f) = |X(f)|^2, \tag{2.9}$$

then filtering X_s through a filterbank of M filters with center frequencies corresponding to the *mel*-scale, a non-linear frequency scale based on subjective experiments to model the human perception of pitch [40]. Taking the logarithm of the output value X_k at the center frequencies of each filter gives us the cepstrum, and finally a discrete cosine transform (DCT) as shown in Equation 2.10 computes the MFCCs,

$$\mathrm{MFCC}_i = \sum_{k=1}^{M} X_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{M}\right], \tag{2.10}$$

where X_k is the output of the kth mel-filter (of total M filters), and $i = 1, 2, \ldots, N$ where N is the total number of MFCC coefficients. The whole process is summarized in Figure 2.5.

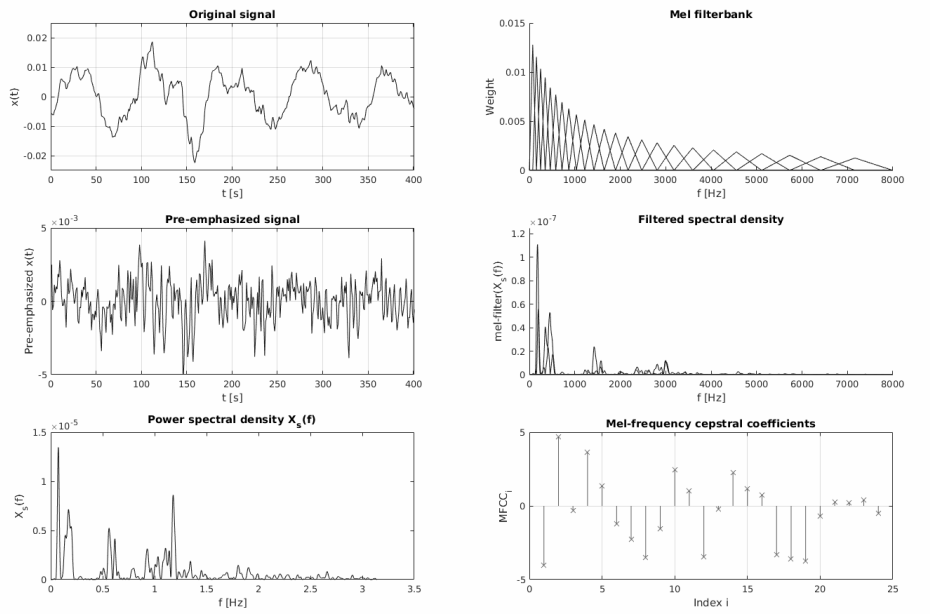

Figure 2.5: MFCC derivation from a 25 ms frame of a speech signal. From the top, the operations in the left row are performed first, followed by the operations in the right row.

As the energy of different speech data can vary, is common to apply *cepstral mean normalization* (CMN)[2], or *cepstral mean and variance normalization* (CMVN). CMN estimates the cepstral mean of a speech segment and then subtracts it from the MFCCs, CMVN does the same, but normalizes the cestral variance as well.

[2]Also known as *cepstral mean subtraction* (CMS), although CMN might be seen as a more generalized term.

Another effective normalization technique is by appying *relative spectral* (RASTA) filtering [18], that is designed to suppress spectral components of a speech recording that has different temporal characteristics than speech, such as very slow or very high variations.

2.6.2 Mel-filterbank coefficients

Although not commonly used directly as features for speaker identification, the direct outputs of the mel filterbank are often used as features for training phonetically aware neural networks (described in Section 2.7.2). Despite the linear relationship between MFCCs and mel-filterbank coefficients, the latter can be shown to be highly correlated, which have motivated the appliance of the DCT in generating the MFCC. However, this correlation turns out to be useful in the mentioned application.

2.6.3 Delta-features

As in other fields of speech processing, it is common to add spectral differences to the original feature vector, called *delta* features. Further, the differences of the delta features are also commonly added, and these are named *delta-delta* features. The calculation of delta, Δ and delta-delta $\Delta\Delta$, is summarized in Equation 2.11, $MFCC_i$ denoting the i-th mel-cepstral coefficient.

$$\Delta_i = MFCC_{i+1} - MFCC_i, \qquad \Delta\Delta_i = \Delta_{i+1} - \Delta_i, \tag{2.11}$$

2.6.4 Deep features

In [15] it is argued that spectral-based features such as MFCCs and perceptual linear predictive (PLP) coefficients mainly are made with speech recognition in mind and questions their ability to optimal discrimination of speaker information. DNN-based features are among the proposed alternative approaches to spectral features, motivated by the successes of DNNs in speech and image recognition. Recent research have shown promising results for DNN-based features also in speaker recognition [29, 26, 38, 32].

The term "deep" in deep features comes from the fact that the features are extracted from one of the deep layers of a neural network. This network has been initially trained for some classification task, and among phone, language and speaker classification, phone classification have by far given the best results [32]. New input data is then applied to the bottom layer of the network, and the deep features can be extracted from the outputs of one of the hidden layers. The input is subject to abstraction as it propagates through the network, and will ideally only contain a low-noise representation of the input data that maximizes speaker discrimination[3]. Further, if this hidden layer is set to have a much lower dimension than the other layers, i.e. a bottleneck layer producing bottleneck features, it can be seen as a compressed representation of the input. Thus, deep features allow for a compressed, speaker discriminative representation of the speech data. The extracted deep

[3]The choice of which layer to extract deep features from is also important. In [26], layers close to the input layer gave the best results.

features can be regarded as any other speaker discriminative feature, and used to train the models presented in Section 2.8.

Deep features can be combined (or stacked) with its original spectral features to create *tandem features*, combining the abstracted information of the deep features with the original spectral information, as some of the latter might have been lost in the deep feature extraction. This has shown to give the best results in recent comparisons between different deep feature approaches [26, 32].

2.7 Speech modeling

Speech can be viewed from several perspectives; as realizations of vectors from a general acoustic space, a sequence of utterances, a sequence of words, a sequence of phones or a sequence of sub-phonetic components. A model of speech will depend on the chosen level of abstraction. As will be presented in Section 2.8, state-of-the-art speaker text-*independent* recognition systems do not include word- or phone-based modeling, but are trained fully *unsupervised* on unlabeled speech data. For text-*dependent* approaches however, various techniques can be applied to take advantage of the lexical and phonetic constraints, borrowed or inspired by the field of speech recognition. This section briefly mentions two of these techniques.

2.7.1 Hidden Markov models

The hidden Markov model (HMM) is a Bayesian network consisting of a set of hidden *state* variables and a set of observable *evidence* variables, designed to model a temporally changing observation sequence such that one is able to infer the underlying state sequence. In speech applications, the observable variables can be speech features or some speech model parameters, and the hidden variables can be words or phones, an example with the latter shown in Figure 2.6.

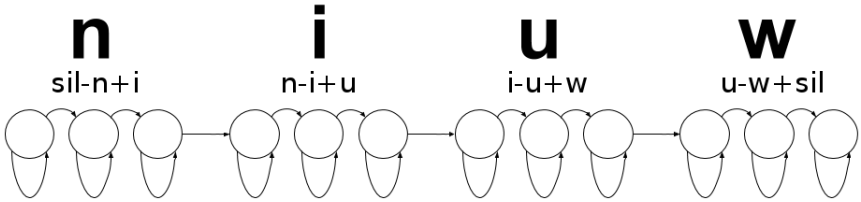

Figure 2.6: An example of a triphone-based HMM, modeling the word "new". Each circle represent states, and the arrows transitions between the states.

2.7.2 Phonetically aware neural networks

A central part of state-of-the-art speech recognition systems is the phonetically aware neural network, which is a DNN trained for recognizing phones, or more commonly, triphones. Given a set of speech data, each frame of the data are associated with a triphone

model. Obtaining these pairs of data and models is called phonetic alignment. Following training procedures described in Section 2.3.2, features from one frame, along with a number of preceding and succeeding frames (known as *context frames*) are used as input to the network, and the frame's aligned triphone is the output. The total number of output classes will thus be equal to the number triphones associated with the training data set. A properly trained network will now recognize the phone contained in new input, making it suitable for speech recognition. [14] argues that the advantage of replacing a GMM-UBM-based approach with an approach based on a phonetically aware neural networks, is having supervised training, thus outputs of the model that has a phonetic label, as opposed to the unsupervised approach of the GMM-UBM.

2.8 Speaker modeling

This section gives an overview over the most important and state-of-the-art speaker models over the past 20 years.

2.8.1 GMM-UBM with maximum a-posteriori adaptation

GMM-UBM was proposed as an approach for speaker recognition in 1995 [37], being the state of the art approach until the mid-2000s, and still central components in both joint-factor analysis (Section 2.8.3) and i-vector systems (Section 2.8.4). GMM-UBM stands as a robust and simple model.

A UBM is a *world model* which purpose is to model a general acoustic space. A GMM-UBM is a UBM consisting of a number of Gaussian distributions. The GMM-UBM is trained unsupervised through the EM algorithm (Section 2.3.1), using training data that ideally provides a rich amount of examples of phonetic realizations.

From a trained GMM-UBM, speaker models can be adapted using the speakers' enrollment data and maximum a-posteriori (MAP) adaptation. The MAP-adaptation adapts the mixture weights, mean and variances of the UBM. A description of the adaptation procedure can be found in [17, p. 87–88].

The combination of GMM-UBM and MAP highly reduces the amount of required enrollment data from each speaker, making it more applicable to a variety of applications where recording large sets of enrollment data is infeasible.

A set of T feature vectors $X = \{x_n, n = 1, 2, .., T\}$ can be compared to the UBM model λ_U and the individual speaker models λ_k, k denoting the speaker, by the computing

$$\frac{\mathrm{P}(X \text{ belongs to speaker } k)}{\mathrm{P}(X \text{ does not belong to speaker } k)} = \frac{\mathrm{P}(\lambda_k|X)}{\mathrm{P}(\lambda_U|X)}, \tag{2.12}$$

where, since feature vectors usually are assumed statistically independent,

$$\mathrm{P}(\lambda|X) = \prod_{n}^{T} \mathrm{P}(\lambda|x_n). \tag{2.13}$$

Computing the logarithm of Equation 2.12 gives us the log-likelihood ratio (LLR),

$$\mathrm{LLR}(X, k) = log(\mathrm{P}(\lambda_k|X)) - log(\mathrm{P}(\lambda_U|X)), \tag{2.14}$$

which in speaker recognition terms can be regarded as the *score* of X against the speaker model λ_k.

Among normalization techniques for GMM-UBM scores, *zero-normalization* (z-normalization) and *time-normalization* (t-normalization) are simple, but effective approaches. The two methods both involves estimating the mean μ_{norm} and variance σ_{norm} of a separate set of scores to compute the normalized scores $\mathrm{LLR}_{norm}(X, k)$ as shown in Equation 2.15.

$$\mathrm{LLR}_{norm}(X, k) = \frac{\mathrm{LLR}(X, k) - \mu_{norm}}{\sigma_{norm}} \tag{2.15}$$

Z-normalization compensates for inter-speaker variability [44]. The normalization parameters for speaker k are estimated from a set of scores from speakers other than k against λ_k. On the other hand, t-normalization compensates for inter-session variability [44], and its normalization parameters are estimated from scores of a feature vector X_k from speaker k against other speaker models $\lambda_l, l{\neq}k$. Finally, z- and t-normalization can be applied in cascade to produce zt-normalized scores (z-normalization followed by t-normalization) and tz-normalized scores (vice versa). Varying speaker and session variability properties will affect what normalization method that is the most appropriate.

2.8.2 Support vector machine - universal background model

The support vector machine (SVM) is a binary classifier that has proved to be quite applicable on the problem of classifying the GMM-UBM *supervectors*. The supervector is obtained by stacking the mean vectors of the MAP-adapted speaker model into one high-dimensional vector $\mathbf{m}$, and the SVM classifier separates the supervectors with a hyperplane, trained from labeled examples. This approach is known as the SVM-UBM, and its details can be found in [5].

As the SVM is essentially a binary classifier, the SVM-UBM is mostly utilized for speaker verification. However, several multi-class SVMs have been proposed, and the SVM-UBM-approach have been shown to be performing comparably to the GMM-UBM-approach for speaker identification [1].

2.8.3 Joint factor analysis

Proposed in 2004, joint factor analysis (JFA) [19] attempts to account for weaknesses in earlier systems when the test data is characterized by high speaker and/or session varibility. JFA does this by factorizing the GMM supervector $\mathbf{m}_{s,h}$ from speaker s and session h, representing it as a sum of four components:

$$\mathbf{m}_{s,h} = \mathbf{m}_0 + \mathbf{U}\mathbf{x}_h + \mathbf{V}\mathbf{y}_s + \mathbf{D}\mathbf{z}_{s,h}. \tag{2.16}$$

In Equation 2.16, $\mathbf{m}_0$ is the GMM supervector, $\mathbf{U}$ is a matrix that spans the session subspace, and $\mathbf{V}$ and $\mathbf{D}$ are matrices that spans the speaker subspace. $\mathbf{x}_h$, $\mathbf{y}_s$ and $\mathbf{z}_{s,h}$ are normally distributed random vectors.

As a side note, there exist various other suggested models that models the GMM supervector of the target speaker, one example being inter-session variability (ISV) modelling [43], which, as opposed to JFA, only models the session variability.

2.8.4 Identity vectors

Identity vectors, or just i-vectors, were proposed in 2009 [10]. I-vectors succeeded JFA, and is today regarded as the state-of-the-art speaker recognition approach.

I-vectors model $\mathbf{m}_{s,h}$ to consist of $\mathbf{m}_0$ and what is called the total variability (TV) space, spanned by a matrix $\mathbf{T}$. $\mathbf{T}$ represents what was in JFA divided into speaker and session variability subspaces. The motivation of jointly modeling the separate subspaces come from findings in [9], where the channel subspace modeling was found to also contain speaker information. Thus, the expression for $\mathbf{m}_{s,h}$ becomes

$$\mathbf{m}_{s,h} = \mathbf{m}_0 + \mathbf{T}\mathbf{w}_{s,h}. \tag{2.17}$$

As in Equation 2.16, $\mathbf{w}$ contains normally distributed variables called the *total factors*. The i-vector $\hat{\mathbf{w}}$ is estimated from $\mathbf{w}$ using Baum–Welch statistics extracted using the UBM [10].

After extraction of the i-vectors, classification is straightforward using support vector machines (SVM) or, as proposed in [10], cosine-distance scoring.

Another way of classification is through *probabilistic linear discriminant analysis* (PLDA), also a method of factor analysis that is similar to the JFA model, but with different subspace matrices dimensions. Details on PLDA can be found in [35]. In the recent years, PLDA has proved to give very good results, and is a part of today's state-of-the art i-vector-based speaker recognition framework [17].

2.9 System evaluation

Standalone scores themselves provide little information about the overall system performance before they are compared to one another. This section introduces error measures in evaluations of speaker recognition systems.

2.9.1 Acceptance and rejection

In speaker verification systems, the trial speaker provides a test utterance along with a claimed identity, and after processing and feature extraction, the system provides a score of the utterance against the speaker model of the claimed identity. After possible score normalization, the score s can be compared to a set threshold θ, and is then accepted as originating from the claimed speaker if $s \geq \theta$, or rejected otherwise. This decision is summarized in Equation 2.18.

$$\text{Decision}\{s\} = \begin{cases} accept\ s & \text{if } s \geq \theta \\ reject\ s & \text{if } s < \theta \end{cases} \tag{2.18}$$

Depending on the values of s and θ, the decision function in Equation 2.18 can decide correctly, that is, accepting an utterance if it indeed comes from the claimed speaker, or rejecting an utterance if it does not. Further, an incorrect decision will occur if the function *falsely accepts* an utterance that is not from the claimed speaker, or if it *falsely*

rejects an utterance that actually is. Thus if we look at all scores, we can define the *false-acceptance rate* (FAR) as the relative number of falsely accepted utterances, and the *false-rejection rate* (FRR) as the relative number of falsely accepted utterances, as summarized in Equation 2.19.[4]

$$\begin{aligned} \mathrm{FAR} &= \frac{number\ of\ falsely\ accepted\ utterances}{total\ number\ of speaker\ trials\ of\ incorrect\ speakers} \\ \mathrm{FRR} &= \frac{number\ of\ falsely\ rejected\ utterances}{total\ number\ of\ trials\ of\ correct\ speakers} \end{aligned} \tag{2.19}$$

To be able to evaluate the system independently of the decision threshold, one can calculate the FAR and FRR for a range of values of θ. This results in some commonly used error measures introduced in the following subsections.

2.9.2 Common error measures for speaker verification

Because of their dependency of θ, the FAR and FRR do not provide sufficient evaluation of the system unless evaluated for a range of values of θ. One such value of θ will give the *equal error rate* (EER), that occurs when FAR = FRR. The EER can summarize the general performance of a system, and is a popular error measure used in most speaker verification research.

The EER does not, however, consider cases where having a low FAR is more important than keeping the FRR down, such as in verification systems with high security (and the other way around). This has motivated the introduction of another commonly used measure, namely the detection cost function (DCF), defined as

$$\mathrm{DCF}(\theta) = \mathrm{P(FR)C(FR)P}(target) + \mathrm{P(FA)C(FA)}(1 - \mathrm{P}(target)),$$

where C is a cost function which values are manually set weight the FA and FR errors depending on the application of the verification system, P($target$) the prior probability of the target speaker, and P(FA) and P(FR) the posterior probabilities of FA and FR given θ, nontarget and target speaker, respectively. By changing θ one finds the lowest value of the DCF, known at the *minimum DCF* (minDCF).

By plotting the FAR against the FRR, we obtain the detection error tradeoff-curve (DET) [31], which is the current standard way of plotting the system performance of speaker recognition systems. As a supplement to the DET-curve, it is common to add the EER and minDCF points, providing a compact representation of the system performance that also eases comparison between different systems.

Figure 2.7 shows an example of a DET curve for a single speaker verification system, with EER and minDCF points.

[4] False acceptances (FA) and false rejections (FR) are also called *false positives* and *false negatives*, respectively

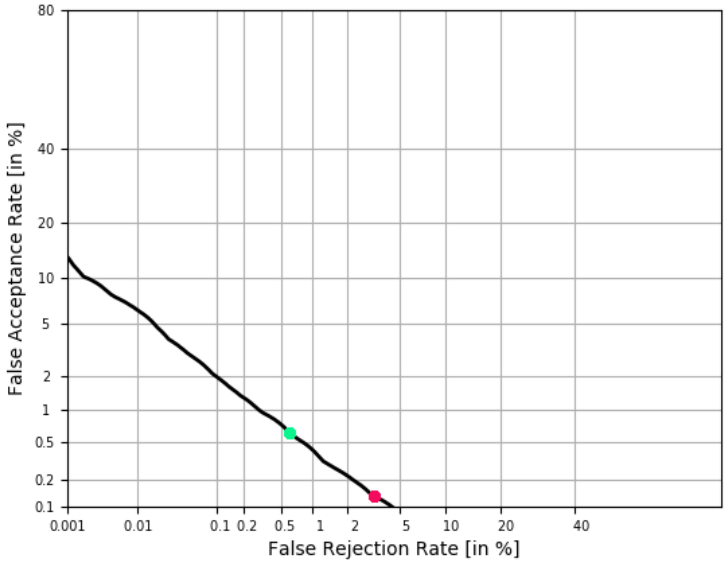

Figure 2.7: An example of a DET-curve. EER and minDCF point marked as green and red, respectively.

2.9.3 Error measures for speaker identification

The error measures mentioned so far have all concerned speaker verification systems. Speaker identification is generally given less attention within speaker recognition, and established measures exclusively for speaker identification system evaluation does not exist to the author's knowledge.

A speaker identification trial involves scoring the utterance in question against all models of the enrollment speakers in the set. A *naive* approach to classify the utterance would be to assign it to the highest scoring model, which might be considered good enough in a closed set setting [13]. In the case of a closed set, or in the need of a more strict classification, a global decision threshold θ can be introduced, where, as in Equation 2.18, the utterance is rejected for all models to which it scores lower than θ. In addition, the difference between the best scores must be taken into account, as an utterance might for some models give scores that are very close to one another. Thus for two scores s_1 and s_2 in a set of scores, where $s_1 > s_2$ and $s_1 > \theta$, the decision function $\mathrm{Decision}\{\mathrm{s}\}$ can be defined as in Equation 2.20.

$$\mathrm{Decision}\{s\} = \begin{cases} accept\ s_1 & \text{if } |s_1 - s_2| \geq \delta \\ reject\ s_1 & \text{if } |s_1 - s_2| < \delta \end{cases} \tag{2.20}$$

Further this classification procedure gives us three distinguishable errors:

- False acceptance (open set scenario), when an utterance from an unknown speaker is accepted to one of the enrolled models.

- False rejection (both open and closed sets), when an utterance belongs to an enrolled speaker, but is rejected.

- Erroneous acceptance [5] (both open and closed sets), when an utterance belongs to an enrolled speaker, but is erroneously classified as to belong to another of the enrolled speakers

One way of measuring the system performance of a speaker identification system is to use the relative recognition rate,

$$recognition\ rate = \frac{number\ of\ correctly\ classified\ utterances}{total\ number\ of\ trials},$$

and another be to plot the error rates for changing values of θ. A third would be to regard all trials as speaker *verification* trials, using the measures introduced in the previous section, as this would give indications of the robustness of the system also as a speaker identification system.

2.10 Databases for speaker recognition

Speaker recognition system evaluation requires designated datasets to make results comparable with the results of other researchers. This section presents some of the commonly used datasets used in today's speaker recognition research.

Many of these datasets are managed by the Linguistic Data Consortium (LDC), an organization that since 1992 have created and managed speech datasets for various purposes. Datasets are organized in catalogue numbers that indicate year of publication and type of data (e.g. speech, transcription). One is able to obtain datasets through individual purchases or memberships of the LDC, giving free access to datasets published after the date of registration [7].

2.10.1 Early speech corpora

Recording of speech corpora devoted for speech recognition research began in the 1980s, with TIMIT as one of the most popular, still in use in many evaluations today. TIMIT, collected by Texas Instruments (TI) and Massachusetts Institute of Technlogy (MIT), is a collection of phonetically rich English sentences read by American speakers with different dialects. Early databases recorded exclusively for speaker recognition include the KING database, focusing on telephone channel variation, and the YOHO database, consisting of digit passcodes. [27, p. 1442–1555].

[5] As proposed by the author, might be addressed differently in other literature.

2.10.2 Fisher, Mixer and Switchboard

The Switchboard corpus is a series of corpora recorded in the 1990s and early 2000s by Texas Instruments. Its catalogues consists of 8 kHz sampled recordings of English telephone conversations in the US, where participants were automatically matched with one another to have a conversation on a randomly selected topic [27, p. 1442–1555].

Similar to the Switchboard series, the Fisher and Mixer corpora, recorded by the LDC, mainly consists of 8 kHz telephone conversations on prompted topics. Mixer also contains some read speech and some speech in other languages than English.

2.10.3 NIST datasets

Since 1996, the National Institute of Standards in Technology (NIST) have held annual or biannual competitions on text-independent speaker recognition, the latest held fall 2016 under the name NIST 2016 Speaker Recognition Evaluation (SRE16) [34]. The evaluations of NIST have been a driving force for the speaker speaker recognition community, contributing with setting standard system evaluation metrics such as the DCF, and with each evaluation increasing levels of difficulty as system performances increase, focusing on specific areas for each evaluation.

The data used in the NIST SRE evaluations have resulted in datasets of training and test data named after each evaluation, of which a subset is part of the standard evaluation protocols in speaker recognition research of today. Most of this data is recorded over noisy telephone channels and sampled at 8 kHz.

2.10.4 The PRISM evaluation set

A standardized compilation of essential datasets was in 2011 published as the PRISM Evaluation Set [12]. This is a large-sized publicly available dataset consisting of NIST SRE 2004–2010 evaluations, Mixer, Fisher and Switchboard datasets, for the purpose of evaluating systems under varying conditions such as language, channel and speaker.

2.10.5 RSR2015

The RSR2015 database [24] was collected with the purpose of providing a complete corpus for text-dependent speaker recognition system evaluation, and contains speech from 300 speakers (143 female, 157 male) recorded on mobile devices, sampled at 16 kHz. It differs from other datasets not only by being text-dependent and recorded on modern mobile devices, but also by including utterances of different durations; short commands, phonetically rich TIMIT sentences and digits. The corpus is in English, with speakers of three Singaporean ethnicities. Each speaker have recorded 30 sentences, 30 commands and 13 digit strings, all in a total of 9 different sessions on 3 different devices.

2.10.6 RedDots

In addition to the RSR2015 database, the RedDots corpus [25] is another recent effort of creating a database with more control of sessions. Similar to RSR2015, RedDots is

recorded using mobile devices, but focuses on not just session, but also speaker variability, capturing different modes (e.g. healthy, stressed, ill) of a speaker through having one session per week in one year, 52 in total. It thus differs from RSR2015 by having a large number of sessions, with only 24 utterances of short duration per session. As of November 2016, RedDots consists 89 speakers of English speech (72 male, 17 female) from native and non-native speakers from 21 countries [36].

2.11 Toolkits for speaker recognition

The process of setting up a framework for speech processing from scratch requires a substantial amount of work. Fortunately, various programming toolkits have been developed, so researchers need only speech data and a parameter scheme to perform experiments. This section mentions the existing toolkits for speaker recognition.

SPEAR [21], ALIZE [3] and SIDEKIT [23] are high-level integrated toolkits that is made with simplicity in mind, and include example setups[6]. They all have non-restrictive licenses that opens up for commercial applications.

Kaldi is a popular open-source toolkit developed for speech recognition, with has a large community of users and contributors. It contains modules for speaker recognition applications, and is a powerful tool very supportive of DNN approaches, however requiring some experience before setting up own experiments. It is fully developed in C++ under a non-restrictive license.

Finally there the MSR Identity Toolbox [39] of MATLAB, that provides a high-level interface for speaker recogntion experiments, with the wide and powerful functionality of MATLAB easily available. One might also mention the Hidden Markov Model Toolkit (HTK) [45], another high-level toolkit that includes some speaker verification functionality. A drawback with these two are the highly restrictive licences, making them unsuitable for commercial usage.

[6] Although *integrated*, they all rely on som external packages and tools, most notably ALIZE, that for instance uses SPRO and HTK for feature extraction. SIDEKIT is the most integrated solutions of these three [23].

Chapter 3

Speaker recognition in SIDEKIT

SIDEKIT (Speaker IDEntification ToolKIT) [23] is an open-source toolkit built exclusively for speaker and language recognition. With readability and comprehensiveness in mind, it is build for easy implementation of experiments for education and research. In [23], authors argue the need for such a toolkit by pointing out the fact that existing toolkits are 1): written partly or fully in languages that reduces readability and makes it harder to use for beginners in the field, and 2): dependent on other programs for certain stages of the toolchain, such as feature extraction. Sidekit contains functionality for the full toolchain from preprocessing of data and feature extraction to classification and system evaluation. All its code is written in Python, with the current version (SIDEKIT 1.2.1) only tested with Python versions 3.5 and above.

This chapter gives a short introduction to the classes and functionality of SIDEKIT, with examples of system implementation. Note, however, that the online SIDEKIT documentation [22] contains a complete API description and tutorials for baseline speaker recognition experiments. The summary in this chapter therefore be more of a guide to an actual implementation, focus on a given task through all steps of the toolchain. It is assumed that the reader has fundamental knowledge of Python and commonly used Python libraries (os, random).

3.1 Classes and data organization

A speaker recognition experiment in SIDEKIT involves five classes:

- **FeaturesExtractor**, for feature preprocessing and extraction.
- **FeaturesServer**, for feature processing and "feeding" lists of training segments to other modules.
- **Mixture**, for training and managing Gaussian mixture models.
- **FactorAnalyser**, for methods based on factor analysis (JFA, i-vectors, PLDA).

- **StatServer**, for storage and processing of first and second order statistics.

In addition, SIDEKIT includes classes for managing datasets. These are adopted from the MATLAB-based Bosaris framework [4], to be consistent with standards in speaker recognition research. The classes are:

- **IdMap**, for mapping segments to speakers, used in the enrollment phase.
- **Key**, for storing trial information. Maps which segments that are to be scored against which models.
- **Ndx**, stores information about which trials that are target trials and which that are nontarget trials.

Finally, SIDEKIT provides the class **Scores** to store trials scores.

Apart from audio input formats, which are wave, sph and raw-PCM, SIDEKIT stores and loads all its data using the HDF5-format [41], allowing for compact, binary storage of big amounts of data. For managing HDF5-encoded files, SIDEKIT uses the h5py library [6]. Other libraries of importance are the Theano library [42], used mainly for neural network training, and Multiprocessing, which allows for parallel computation, often useful for feature extraction and training procedures.

3.2 Demonstration code

It is chosen to include example code that can act as a supplementary example in addition to the existing tutorials provided in the SIDEKIT documentation.

3.2.1 Managing datasets

As mentioned in the pervious section, datasets are managed using IdMap, Key and Ndx. There are many ways of making lists of dataset, often defined by the structure of the chosen dataset. For simplicity, let us assume that all the data is pre-organized in designated folders for UBM training data, enrollment data, test data, and normalization data, respectively[1]. We further assume that each audio file has the following name structure. "*<gender><speaker>_<session>_<sentence>.ext*", where,

- ***gender*** is a letter indicating the gender of the speaker, *f* if the speaker is female, *m* if male.
- ***speaker*** is a ID-number of the speaker.
- ***session*** is the ID-number of the session.
- ***sentence*** is the ID-number of the sentence.
- ***ext*** is the file extension.

[1] Directory structures tend to vary between datasets. In this case, it might be wiser to read pre-defined lists of files (if provided), or use traversing functions like os.path.walk(), and follow the dataset protocol for defining sets.

With these assumptions, Listing 3.1 makes a list of audio files for UBM training, and an IdMap object from the data in the training data directory.

```
import sidekit
import os

# Make ubm file list
ubm_data_dir = "./data/audio/ubm_data" # UBM data directory
ubm_list = os.listdir(ubm_data_dir) # list files in directory
ubm_list = ["ubm_data/"+files.split('.')[0] for files in ubm_list] #
    remove extension and add path prefix

# Make enrollment (IdMap) file list
enroll_data_dir = "./data/audio/enroll_data" # enrollment data directory
enroll_list = os.listdir(enroll_data_dir)
enroll_models = [files.split('_')[0] for files in enroll_list] # list of
    model IDs
enroll_segments = ["enroll_data/"+files.split('.')[0] for files in
    enroll_list]

# Generate IdMap
enroll_idmap = sidekit.IdMap()
enroll_idmap.leftids = np.asarray(enroll_models)
enroll_idmap.rightids = np.asarray(enroll_segments)
enroll_idmap.start = np.empty(enroll_idmap.rightids.shape, '|O')
enroll_idmap.stop = np.empty(enroll_idmap.rightids.shape, '|O')
```

Listing 3.1: Preparing UBM training and enrollment data

As we shall see, adding path prefixes "ubm_data" and "enroll_data" allows us to use the same FeatureExtractor object for all audio files. So before feature extraction enrollment, let us in Listing 3.2 define the trials by defining the Key and Ndx-objects. To keep the example simple, we assume there are at least one utterance from each of the enrollment speakers, and that there are no utterances from speakers outside the enrollment set. With num_models enrollment speakers and $test\_size$ test utterances, there can be maximum $num\_models \cdot test\_size$ trials [2]. The Key and Ndx objects require two aligned arrays, one containing test segments, the other one containing the model that the corresponding test segment in the first.

```
# Make list of test segments
test_data_dir = "./data/audio/test_data" # test data directory
test_list = os.listdir(test_data_dir)
test_list = ["test_data/"+files.split('.')[0] for files in test_list]

# Make lists for trial definition, and write to file
test_models = []
test_segments = []
test_labels = []

for models in enroll_models:
    for segments in test_list:
        test_models.append(models)
        test_segments.append(segments)
```

[2] This is usually a relatively large number. Performing all possible trials should not be necessary for a thorough system evaluation.

```
        # Compare gender and speaker ID for each test file
        if files.split('_')[0] == models:
            test_label.append('target')
        else:
            test_label.append('nontarget')

with open("./task/trial_definition.txt","w") as fh:
    for i in range(len(test_models)):
        fh.write(test_models[i]+' '+test_segments[i]+' '+test_labels[i]+'\
    n')

# Define Key and Ndx from text file
key = Key("./task/trial_definition.txt")
ndx = key.to_ndx()
```

Listing 3.2: Trial definition with Key and Ndx

Now that all audio file list are ready, this is an appropriate stage to define the FeatureExtractor and FeaturesServer objects as exemplified in Listing 3.3. Note that the FeatureExtractor does not extract and save features before a function for saving, such as FeatureExtractor.save(), is explicitly called. In this example, features are extracted "on the fly" after being passed to the FeaturesServer and then to functions for UBM training.

In Listing 3.3, the FeaturesExtractor *fe* is being defined for audio of with a sampling frequency at 16 KHz, which, in 25 ms sized pre-emphasised windows with 10 ms frameshift, is filtered through a 300–3400 Hz band-pass limited filterbank of size 24, and processed into 19 MFCCs. In addition, energy-based VAD is applied, and VAD, log-energy and MFCCs are to be passed on to the next stage. The FeaturesServer *fs* is then initialized to read the parameters of *fe*, performing CMVN and calculating delta and delta-delta parameters. By including "vad" in *dataset_list*, only frames containing speech will be passed on to the next stages.

```
# Initialize FeaturesExtractor
fe = sidekit.FeaturesExtractor(audio_filename_structure="./data/{}.wav",
                               feature_filename_structure=None,
                               sampling_frequency=16000,
                               lower_frequency=300,
                               higher_frequency=3400,
                               filter_bank="log",
                               filter_bank_size=24,
                               window_size=0.025,
                               shift=0.01,
                               ceps_number=19,
                               vad="energy",
                               pre_emphasis=0.97,
                               save_param=["vad","energy","cep"],
                               keep_all_features=False)


# Initialize FeaturesServer
fs = sidekit.FeaturesServer(features_extractor=fe,
                            feature_filename_structure=None,
                            sources=None,
                            dataset_list=["vad","energy","cep"],
                            mask=None,
                            feat_norm="cmvn",
```

```
                                global_cmvn=None,
                                dct_pca=False,
                                dct_pca_config=None,
                                sdc=False,
                                sdc_config=None,
                                delta=True,
                                double_delta=True,
                                delta_filter=None,
                                context=None,
                                traps_dct_nb=None,
                                rasta=False,
                                keep_all_features=False)
```

Listing 3.3: Initialization of the FeatureExtractor and FeaturesServer objects

3.2.2 Model training and adaptation

With the groundwork done, the next step is to use our UBM data list and IdMap for UBM-training and speaker enrollment. SIDEKIT allows more than one implementation of the EM algorithm; this example uses the EM_split() algorithm. The EM_split() algorithm applies the expectation and maximization steps starting with one mixture component, then two, then four, until the specified number of distribution, *distrib_nb*, is reached. The number of EM iterations for each doubling is set by the parameter *iterations* (omitted in this example). EM_split() also allows for parallell computation, the number of processes set by *num_thread*. Listing 3.4 gives an example where a 512-mixture GMM using 8 parallel processes, producing the output *llk*, a list of log-likelihood values after each EM iteration.

```
# Initialize and train 512 mixture GMM
ubm = sidekit.Mixture()
llk = ubm.EM_split(features_server=fs,
                   feature_list=ubm_list,
                   distrib_nb=512,
                   num_thread=8,
                   save_partial=False)
ubm.write('ubm.h5') # save the UBM
```

Listing 3.4: UBM training

With the UBM is is now possible to adapt speaker models with the enrollment data defined in *enroll_idmap*. Computation of sufficient statistics and MAP adaptation is shown in Listing 3.5.

```
# Create StatServer for the enrollment data
enroll_stat = sidekit.StatServer(statserver_file_name=enroll_idmap,
                                 distrib_nb=512,
                                 feature_size=60)
# Compute the sufficient statistics
enroll_stat.accumulate_stat(ubm=ubm,
                            feature_server=fs,
                            seg_indices=range(enroll_stat.segset.shape[0]),
                            num_thread=8)
enroll_stat.write('data/enroll_stat.h5')

# MAP adaptation of enrollment speaker models
```

```
regulation_factor = 3  # MAP regulation factor
enroll_sv = enroll_stat.adapt_mean_map_multisession(ubm=ubm,
                                                    r=regulation_factor)
```

Listing 3.5: MAP adaptation

3.2.3 Scoring and evaluation

For a GMM-UBM framework, we now have all necessary components to score the test segments against enrolled models. Listing shows how SIDEKIT allows for simple scoring. The function gmm_scoring() returns a Scores object.

```
# Compute scores
scores_gmm_ubm = sidekit.gmm_scoring(ubm=ubm,
                                     enroll=enroll_sv,
                                     ndx=test_ndx,
                                     feature_server=features_server,
                                     num_thread=nbThread)
scores_gmm_ubm.write('gmm_ubm_scores.h5')
```

Listing 3.6: Scoring of test segment against enrolled models

Raw scores provides little insight, but SIDEKIT provides functionality for computing EER and minDCF, as well as DET-curve plotting, which is based on the Matplotlib library. The example in Listing 3.7 shows an example on how this could be done, inspired by the online SIDEKIT tutorial examples. The resulting figure will be similar to the DET-plot presented in Figure 2.7.

```
# Make DET plot
prior = sidekit.logit_effective_prior(0.01, 10, 1)
dp = sidekit.DetPlot(window_style='sre10', plot_title='Scores GMM-UBM')
dp.create_figure()
dp.set_system_from_scores(scores_gmm_ubm, key, sys_name='GMM-UBM')
dp.plot_rocch_det()
dp.plot_DR30_both(idx=0)
dp.plot_mindcf_point(prior, idx=0)

dp.show() # Display figure
dp.__figure__.savefig('DET_GMM_UBM.png') # Save figure
```

Listing 3.7: Plotting the DET curve with EER and minDCF points

3.2.4 I-vector system

With version 1.2, released early 2017, SIDEKIT included the FactorAnalyser()-class, simplifying implementation of the i-vector toolchain. By keeping the variables from the previous section, and assuming that we already have prepared IdMaps *tv_idmap* and *plda_idmap* for TV- and PLDA-matrix training, respectively, Listing 3.8 prepares the i-vector experiments by training a TV matrix of dimension 400 with 10 iterations.

```
# Create a joint StatServer for TV and PLDA training data
back_idmap = plda_idmap.merge(tv_idmap)
back_stat = sidekit.StatServer(statserver_file_name=back_idmap,
```

```
                                distrib_nb=512,
                                feature_size=60)

# Jointly compute the sufficient statistics of TV and PLDA data
# and write to 'data' directory
back_stat.accumulate_stat(ubm=ubm,
                          feature_server=fs,
                          seg_indices=range(back_stat.segset.shape[0]),
                          num_thread=8)
back_stat.write('data/stat_back.h5')

# Load the sufficient statistics from TV training data
tv_stat = sidekit.StatServer.read_subset('data/stat_back.h5', tv_idmap)
tv_stat.write('data/tv_stat.h5')

# Train TV matrix using FactorAnalyser
fa = sidekit.FactorAnalyser()
fa.total_variability('data/tv_stat.h5',
                     ubm=ubm,
                     tv_rank=400,
                     nb_iter=10,
                     output_file_name='data/TV_matrix',
                     num_thread=8)
tv = fa.F # TV matrix
tv_mean = fa.mean # Mean vector
tv_sigma = fa.Sigma # Residual covariance matrix
```

Listing 3.8: Compute sufficient statistics and train a 400-dimensional TV matrix training

With the TV-matrix and IdMaps, the next step is to extract the i-vectors from enrollment, test and PLDA data. Listing 3.9 shows the example code. It is assumed that the sufficient statistics of the test data have been computed in advance and written to 'data/test_stat.h5', as in Listing 3.5.

```
# Extract i-vectors from enrollment data
enroll_iv = fa.extract_ivectors(ubm=ubm,
                                'data/enroll_stat.h5',
                                num_thread=8)

# Extract i-vectors from test data
test_iv = fa.extract_ivectors(ubm=ubm,
                              'data/test_stat.h5',
                              num_thread=8)

# Load sufficient statistics and extract i-vectors from PLDA training data
plda_stat = sidekit.StatServer.read_subset('data/back_stat.h5',
                                           plda_idmap)
plda_stat.write('data/plda_stat.h5')
plda_iv = fa.extract_ivectors(ubm=ubm,
                              'data/plda_stat.h5',
                              num_thread=8)
```

Listing 3.9: Extraction of i-vectors from enrollment, test and PLDA training data

Now, cosine distance scoring is straightforward. An example implementation is shown in Listing 3.10.

```
# Do cosine distance scoring and write results
```

```
scores_ivector_cos = sidekit.iv_scoring.cosine_scoring(enroll_iv,
                                                       test_iv,
                                                       ndx,
                                                       wccn=None)

# Write scores
scores_cos.write('scores/scores_ivector_cos.h5')
```

Listing 3.10: Cosine distance scoring of extracted i-vectors from enrollment and training data.

Finally, Listing 3.11 shows the training of a 400-dimensional PLDA matrix, followed by PLDA scoring.

```
# Train PLDA matrix
fa_plda = sidekit.FactorAnalyser()
fa_plda.plda(stat_server=plda_iv,
             rank_f=400,
             nb_iter=10,
             scaling_factor=1,
             output_file_name=PLDA_matrix,
             save_partial=False)

# Perform PLDA scoring
scores_plda = sidekit.iv_scoring.fast_PLDA_scoring(enroll=enroll_iv,
                                                   test=test_iv,
                                                   ndx=ndx,
                                                   mu=fa_plda.mean,
                                                   F=fa_plda.F,
                                                   Sigma=fa_plda.Sigma)

# Write scores
scores_plda.write('scores/scores_ivector_plda.h5')
```

Listing 3.11: PLDA matrix training and PLDA scoring.

After scoring the extracted i-vectors, the scores can be evaluated the same way as shown in Listing 3.7.

Chapter 4

Experiments

This section presents the experiments conducted on GMM-UBM and i-vector systems. A setup for bottleneck feature extraction is included, although no experiments have been conducted on this system. The task scenario and global conditions are described first, followed by model-specific parameters, then finally the experimental results with discussion.

4.1 Task scenario

The training and test protocols are defined with a video conference scenario in mind. The system in use will be similar to the one presented in Figure 2.3, a group of people in a meeting room engaged in a video conference. The conference system has a digital assistant that remains inactive until evoked by a short predefined wake word. The assistant is to recognize and execute the command that shortly follows after the wake word. The action of the assistant may vary depending on the speaker, as if for instance the question "<Wake word>, how does my schedule look tomorrow" was to be asked. The assistant must therefore be able to distinguish between speakers; investigating techniques for such a feature has been the purpose of the experiments presented in this section.

It is assumed that the underlying system provides functionality for recording, speech recognition, transcription and wake word detection. Further, the available enrollment data limited to a few sentences, as the enrollment of new speakers must be easily done. There is apart from these constraints, unlimited computational resources to allow instant scoring of incoming utterances, as well as unlimited speech data from other meetings and speakers for background model training.

4.2 Global conditions

The protocol in the experiments has been inspired by the GMM-UBM and i-vector tutorials of the SIDEKIT documentation [22], as well as the papers [24] and [29]. All training and experiments have been conducted using the SIDEKIT toolkit, exept phonetic alignment

that was done in Kaldi, in addition to contributions by the author where needed. Essential parts of the code is found in the Appendix.

4.2.1 Enrollment and test data

For enrollment and test data, data from the RSR2015 database [24] were used. The speaker, session and utterance structure of RSR2015 is as described in the implementation example in Section 3.2.1. Of the 143 female and 157 male speakers, speakers with speaker ID 1–50 were used as evaluation speakers, speakers with ID 51–100 were used for estimation of normalization parameters for z-, t-, zt- and tz-normalization, while the remaining speakers were used for training a complementary model.

To be consistent with the mentioned task scenario, the enrollment data has been restricted, but varied between three, six and nine sentences with sentence IDs 1–3, 1–6 and 1–9, respectively. For each enrollment speakers these have been randomly picked from three RSR2015 sessions, session IDs $\{1,4,7\}$, that have been recorded with the same device for each session.

For system evaluation, the test data have been constructed by choosing utterance ID 32 as a global wake word[1], and concatenating it with randomly selected I) commands with ID 33–60, and II) sentences with ID 10–30. Test sets I) and II) of two different sizes were chosen to investigate the effects of increased amounts of test data on the different systems. To maximize the session mismatch between enrollment and test data, the test data was chosen from different sessions than the enrollment data, that is, from session IDs $\{2,3,5,6,8,9\}$.

Average durations of all sentences and commands used 3.20 s and 1.99 s, respectively [24].

4.2.2 Feature extraction parameters

All data used for model training, normalization and testing was subject to feature extraction with the same parameters. The data was segmented in window sizes of 20 ms, and frame shifts of 10 ms, then a pre-emphasis filtered as described in Equation 2.7 with $\alpha = 0.97$, followed by SNR-based VAD, then band-pass filtered with lower and upper frequencies 300 and 3400 Hz, respectively. From a filterbank of 24 filters, 19 MFCCs were extracted and normalized with CMVN and RASTA filtering. Delta and delta-delta parameters were estimated, resulting in a feature vector of size 57.

4.2.3 System evaluation

The systems have been evaluated both as speaker identification systems and verification systems. The identification task was done as an open set task, where 100 groups of 10 speakers were randomly drawn from the set of the 50 male and 50 female enrollment speakers. Then, for each set, 8 of the speakers were enrolled to the system and labeled as *known* speakers, and the remaining 2 speakers were labeled *unknown*. An utterance from each speaker in the set was then scored against all enrolled models and the complementary

[1] "Watch cartoon"

model, and after normalization techniques subject to decision with thresholding and score difference criterions described in Section 2.9.3.

Speaker verification results were obtained by simply using the "raw" and normalized scores from the speaker identification experiments, but regarding each trial as an independent speaker verification trial. Only scores from enrollment speakers and models, in addition to scores against the complementary model, were included in the evaluations. That is, no results involving the normalization speakers and models were included in the speaker verification evaluation, only *normalized* scores.

4.3 Modeling

This section presents the individual parameters of the GMM-UBM-, i-vector- and bottleneck-feature-based systems.

4.3.1 GMM-UBM system

A 1024 mixture UBM was trained on a 1000 segments of duration 5 minutes, randomly selected from the datasets Switchboard Cellular 1 and 2, NIST SRE 2004, 2005, 2006 (training set only) and 2008. Test-data for estimation of z-normalization parameters were of the same amount as test set sizes I) and II), but was randomly drawn from all 9 sessions. Models for t-normalization was enrolled the same way as the enrollment speakers. All normalization was *gender-dependent*. The evaluated sets of scores include "raw" scores (abbreviated to '*raw*'), z-normalized ('*z-norm*'), t-normalized ('*t-norm*'), zt-normalized ('*zt-norm*') and tz-normalized ('*tz-norm*') scores.

4.3.2 I-vector-based system

The same UBM as in the GMM-UBM system was used A TV-matrix of dimension 400 was trained on the complete datasets Switchboard Cellular 1 and 2, NIST SRE 2004, 2005, 2006 (training set only) and 2008. Scoring was done with cosine distance and PLDA, for which a PLDA matrix of dimension 400 was been trained on the same data as the TV-matrix, but not including Switchboard Cellular 1 and 2. Further, scores after cosine-distance scoring were normalized with WCCN and LDA, and the two combined. The LDA normalization reduced the 400-dimensional i-vectors to 150-dimensions.

Summarized, the evaluated sets of scores include cosine distance scores (abbreviated to '*cos*'), cosine distance with WCCN ('*cos+wccn*'), cosine distance with LDA ('*cos+lda*'), cosine distance with WCCN and LDA combined ('*cos+lda+wccn*'), and finally, PLDA-scoring ('*plda*').

4.3.3 Bottleneck features

A phonetically aligned neural network was configured to train on the Fisher English part 1 dataset. The phonetical alignment was done using the Fisher English tutorial example of Kaldi[2].

[2]In the latest version of Kaldi, this is located in Kaldi/egs/fisher_english.

4.4 Results and discussion

This section presents and discusses the results from the GMM-UBM and i-vector system experiments.

4.4.1 Speaker identification results

Figure 4.1 shows the maximal recognition rates of all systems for the different amounts of enrollment data and test sets I) and II). It shows that the GMM-UBM system gives the best performance when for all sets of scores, most notably with the tz-normalized scores, that has a maximum of 81.3 % for 6 enrollment sentences and test set II), significantly better than all other score sets. The other GMM-UBM-based scores behave more synchronously, suggesting that the other normalization techniques fail to do fulfill their purposes. Adding more test data increases the gap between the tz-normalized scores and the other sets, a sign that the enrolled models lack robustness when facing increased phonetic and session variability, but that tz-normalization does a good job to alleviate the problem.

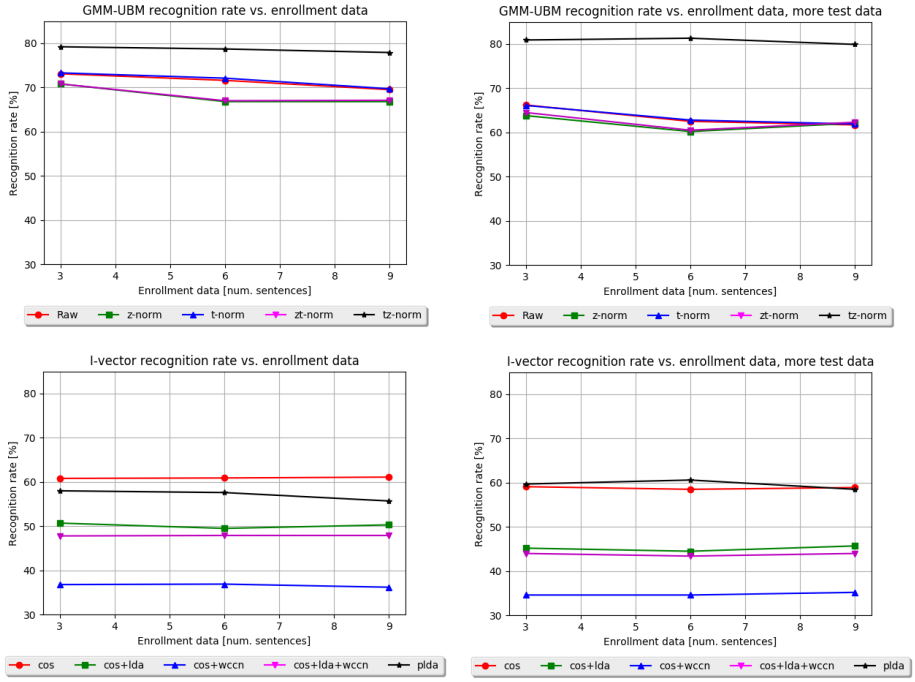

Figure 4.1: Recognition rates for GMM-UBM and i-vector systems versus enrollment and test data. Upper left: GMM-UBM recognition rates for test set I). Upper right: GMM-UBM recognition rates for test set II). Bottom left: I-vector recognition rates for test set I). Bottom right: I-vector recognition rates for test set II).

The i-vector-based scores seems less affected by the increased test data, except for a slight increase in the PLDA-set recognition rates. However, contrary to what might be

expected, the GMM-UBM-based system clearly performs better than the i-vector-based system. This motivates a control of the system parameters.

One reason for the difference in results might be the mismatch between enrollment/test and training data. Recall from Section 2.10 that the UBM, TV and PLDA training data only consist of 8kHz, noisy telephone data with conversational American English speech, while the enrollment and test data is trained on RSR2015, 16 kHz, cleaner, text-dependent Singaporean English. Despite preprosessing and normalization techniques, this mismatch might play a role in the explanation of the result mismatch. As the TV and PLDA matrices require significantly more training data than the UBM, the i-vector-based system is more affected by this mismatch, strengthening this theory.

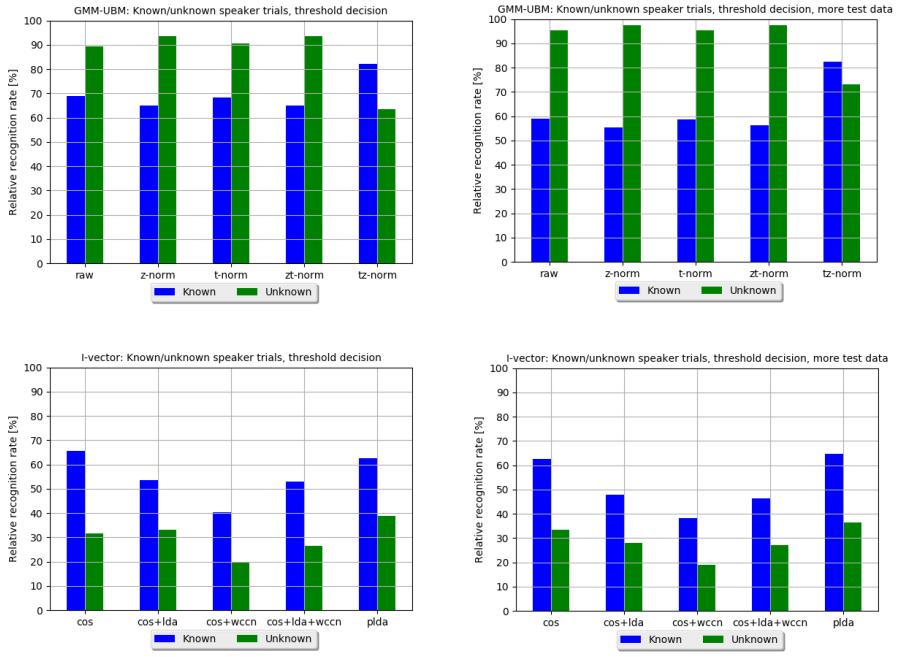

Figure 4.2: Maximum relative recognition rates of the GMM-UBM and i-vector systems, using 3 enrollment sentences. Upper left: GMM-UBM relative recognition rates for test set I). Upper right: GMM-UBM relative recognition rates for test set II). Bottom left: I-vector relative recognition rates for test set I). Bottom right: I-vector relative recognition rates for test set II).

As explained in Section 2.9.3, the recognition rate is a sum of the correctly recognized target trials and correctly rejected nontarget trials. To provide insight on which of these two categories that contribute to the recognition rates, the relative recognition rates with respect to known/unknown speaker trials are presented in Figure 4.3, for 3 enrollment sentencs and test data sets I) and II). The blue bars are the number of correctly accepted known (enrolled) speaker trials, divided by the total amount of known speaker trials, while the green bars are the number of correctly rejected unknown speaker trials, divided by the

total amount for known speaker trials. In numbers, the total amount of known speaker trials are $8\ \text{known speakers} \cdot 100\ \text{sets} = 800\ \text{trials}$, and the number of unknown speaker trials are $2\ \text{unknown speakers} \cdot 100\ \text{sets} = 200\ \text{trials}$.

An initial observation is how the GMM-UBM-based system not only performs better, but also have higher levels of security than the i-vector-based system by successfully rejecting significantly more utterances from unknown speakers. Next, note how the GMM-UBM-based system is more affected by the increased test data. For all normalization methods, only the tz-normalized score set seems to fully benefit from increasing to test set II), its known speaker recognition rate staying constant while its unknown speaker recognition rate increases.

Now, one might expect both the unknown and known recognition rates to increase with more enrollment and test data – why is not this the case? After all, more data equals more evidence that the system could use to make more correct decisions. The i-vector-based system seems unaffected by this increase in data, suggesting that the system has already reached its full potential in terms of speaker modeling. And while the tz-normalized GMM-UBM-based score set benefits from the inceased data, the rest of the sets do not, meaning that we might draw the same conclusion for the GMM-UBM-based system, and that the tz-normalization indeed does a good job, albeit failing to reject as many unknown speakers as the other GMM-UBM normalization methods. In the following section, the system scores are evaluated from a speaker verification, confirming the speaker identification results.

4.4.2 Speaker verification results

Speaker verification tests using 3 enrollment sentences and test set I) are summarized in Figure 4.3 [3], with DET-curves, EER and minDCF values. Reflecting what was found in the identification tests, the GMM-UBM system clearly performs better than the i-vector system, again with the tz-normalized score set far better than the other sets, with an EER of 7.8 %.

Not to repeat the discussion of the speaker identification results, let us briefly move our attention to the minDCF values. The i-vector system minDCF values are much higher, reflecting the poor rejection performances shown in Figure 4.2. For the GMM-UBM-scores, the minDCF values are lower for raw scores and z-, t- and zt-normalized scores, again reflecting the rejection performances in Figure 4.2.

4.4.3 Discussion

As the previous sections have shown, there is little doubt that the GMM-UBM system performs better that the i-vector based system for all normalization and scoring methods, contrary to what would be expected, as the i-vector is considered the state-of-the-art speaker recognition model.

As a comparison with another study, see Figure 7 b) in [24], where an i-vector system comparable to the one used in this task has been tested on short commands from RSR2015,

[3] DET curves for increased amounts of enrollment and test data are omitted due to similarity.

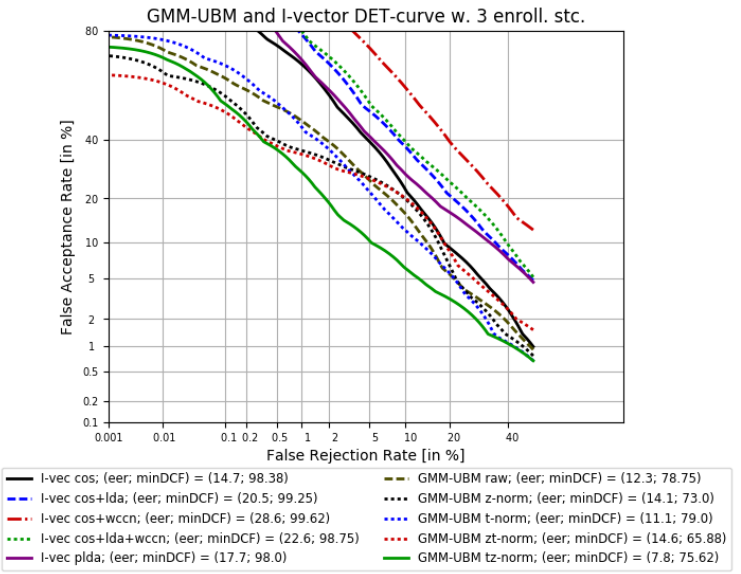

Figure 4.3: DET-curves of all GMM-UBM and i-vector score sets, using 3 enrollment sentences and test set I).

giving an EER of 11.26 % and a minDCF of 46.93. This is better than the i-vector system results in this task, although not better than the best GMM-UBM-based system. However, Figure 7 a) in the same study shows an improvement when the i-vector-based system is tested on the RSR2015 sentences, with an EER of 4.03 % and minDCF of 21.39. As our i-vector system shows no improvement on similar conditions, i.e. on test set II), this suggests that our i-vector system should be subject to further investigation, for instance by using UBM and/or TV-matrix training data of another character. Not forgetting the lack of improvement despite increased enrollment data, the UBM-GMM-system should possibly be evaluated on a different UBM – this to confirm/reject the results, due to their character.

In addition, an evaluation would not be complete without a study of how well recently successful approaches, such as bottleneck features, would apply to the same experimental setup.

Chapter 5

Conclusion

In recent years, smart, speech-driven applications have grown in number and with increasing performance. However, despite their recent introduction to smart speakers, speaker identification techniques have yet to enter the market with a strong presence.

This thesis has investigated speaker identification of utterances of the form *wake word + command* in a video conference scenario. A task description and assumptions have been made, and relevant theory and possible approaches have been summarized. Implementations of GMM-UBM-, i-vector- and bottleneck feature-based systems have been done using the SIDEKIT toolkit for speaker identification, for which an introductory guide has been included in the thesis. Speaker identification and verification experiments have been run on the GMM-UBM- and i-vector-based systems with varying amounts of enrollment and test data. With different normalization and scoring techniques, a total of 10 different approaches have been evaluated.

The GMM-UBM-system have given the best results, with the tz-normalized score set giving a recognition rate of 81.3 % and an EER of 7.8 %. The systems have displayed little improvements for increased enrollment and test data, and mismatch between training and enrollment/test data might be a possible explanation for this. However, it is suggested that these results are reconfirmed with more carefully selected and processed UBM and TV-matrix training data.

For future work, it is encouraged to include the implementation of bottleneck features-based systems, also extending with tandem features (as described in Section 2.6.4), as they have shown to be the most promising among the newly proposed deep-features. Finally, for optimization, it is possible to use the wake word to view the task as *semi-text-dependent*, thus including a temporal aspect in the modeling, as presented in [24]. Hopefully, this thesis has laid the groundwork for further development within the task scenario.

Bibliography

[1] V. R. Apsingekar and P. L. De Leon. Support vector machine based speaker identification systems using gmm parameters. In *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, pages 1766–1769. IEEE, 2009.

[2] H. Beigi. *Fundamentals of speaker recognition*. Springer US, 2011.

[3] J. Bonastre, F. Wils, and S. Meignier. ALIZE, a free toolkit for speaker recognition. *ICASSP '05*, pages 737–740, 2005.

[4] N. Brümmer and E. de Villiers. Bosaris Toolkit. `https://sites.google.com/site/bosaristoolkit/home`. Accessed: 2017-06-03.

[5] W. M. Campbell, D. E. Sturim, and D. A. Reynolds. Support vector machines using gmm supervectors for speaker verification. *IEEE signal processing letters*, 13(5):308–311, 2006.

[6] A. Collette. HDF5 for Python, 2008. `http://h5py.alfven.org`.

[7] L. D. Consortium. Obtaining data. `https://www.ldc.upenn.edu/language-resources/data/obtaining`, 2017. Accessed: 2017-06-05.

[8] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Signal Process. Lett.*, 20(4):357–366, 1980.

[9] N. Dehak. *Discriminative and Generative Approaches for Long-and Short-term Speaker Characteristics Modeling: Application to Speaker Verification*. Thèse de doctorat en génie. École de technologie supérieure, 2009.

[10] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, , and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Trans. Audio, Speech, Lang. Process.*, 19(4):788–798, 2011.

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

[12] L. Ferrer, H. Bratt, L. Burget, H. Cernocky, O. Glembek, M. Graciarena, A. Lawson, Y. Lei, P. Matejka, O. Plchot, et al. Promoting robustness for speaker modeling in the community: the prism evaluation set. In *Proceedings of NIST 2011 workshop*. Citeseer, 2011.

[13] S. Furui. An overview of speaker recognition technology. In *Automatic speech and speaker recognition*, pages 31–56. Springer, 1996.

[14] D. Garcia-Romero and A. McCree. Insights into deep neural networks for speaker recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[15] S. Ghalehjegh and R. C. Rose. Deep Bottleneck features for i-vector based text-independent speaker verification. *IEEE Workshop on ASRU '15*, 2015.

[16] C. Greenberg, A. Martin, D. Graff, L. Brandschain, and K. Walker. 2010 NIST Speaker Recognition Evaluation Test Set, LDC2017S06. Hard Drive, 2017.

[17] J. H. Hansen and T. Hasan. Speaker recognition by machines and humans, a tutorial review. *IEEE Signal Processing Magazine*, pages 74–99, 2015.

[18] H. Hermansky and N. Morgan. Rasta processing of speech. *IEEE transactions on speech and audio processing*, 2(4):578–589, 1994.

[19] P. Kenny. Joint factor analysis of speaker and session variability: Theory and algorithms. Technical report, CRIM, 2005.

[20] L. G. Kersta. Voiceprint identification. *The Journal of the Acoustical Society of America*, 34(5):725–725, 1962.

[21] E. Khoury, L. E. Shafey, and S. Marcel. Spear: An open source toolbox for speaker recognition based on Bob. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[22] A. Larcher, K. Aik Lee, and S. Meignier. SIDEKIT Documentation - home. `http://lium.univ-lemans.fr/sidekit/`. Accessed: 2017-06-03.

[23] A. Larcher, K. Aik Lee, and S. Meignier. An extensible speaker identification sidekit in Python. *ICASSP '16*, pages 5095–5099, 2016.

[24] A. Larcher, K. A. Lee, B. Ma, and H. Li. Text-dependent speaker verification: Classifiers, databases and rsr2015. *Speech Communication*, 60:56–77, 2014.

[25] K.-A. Lee, A. Larcher, G. Wang, P. Kenny, N. Brümmer, D. A. van Leeuwen, H. Aronowitz, M. Kockmann, C. Vaquero, B. Ma, et al. The reddots data collection for speaker recognition. In *INTERSPEECH*, pages 2996–3000, 2015.

[26] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1695–1699. IEEE, 2014.

[27] S. Z. Li and A. Jain. *Encyclopedia of biometrics*. Springer Publishing Company, Incorporated, 2 edition, 2015.

[28] J. Lindh and G. S. Morrison. Humans versus machine: forensic voice comparison on a small database of swedish voice recordings. In *Proceedings of ICPhS*, volume 17, page 4. Citeseer, 2011.

[29] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu. Deep feature for text-dependent speaker verification. *Speech Communication*, 73:1–13, 2015.

[30] R. J. Mammone, X. Zhang, and R. P. Ramachandran. Robust Speaker Recognition: a feature-based approach. *IEEE Signal Processing Magazine*, 13(5):58, 1996.

[31] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The det curve in assessment of detection task performance. Technical report, DTIC Document, 1997.

[32] P. Matějka, O. Glembek, O. Novotnỳ, O. Plchot, F. Grézl, L. Burget, and J. H. Cernockỳ. Analysis of dnn approaches to speaker identification. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5100–5104. IEEE, 2016.

[33] Merriam-Webster.com. Voiceprint. `https://www.merriam-webster.com/dictionary/voiceprint`, 2017. Accessed: 2017-05-23.

[34] N. I. of Standards and Technology. The NIST 2016 speaker recognition evaluation plan. `https://www.nist.gov/sites/default/files/documents/2016/10/07/sre16_eval_plan_v1.3.pdf`. Accessed: 2017-06-05.

[35] Prince, Simon JD and Elder, James H. Probabilistic linear discriminant analysis for inferences about identity. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[36] T. R. Project. Weekly update, week 90. `https://sites.google.com/site/thereddotsproject/data-collection-weekly-update`, 2017. Accessed: 2017-06-05.

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

[38] F. Richardson, D. Reynolds, and N. Dehak. Deep neural network approaches to speaker and language recognition. *IEEE Signal Processing Letters*, 22(10):1671–1675, 2015.

[39] S. O. Sadjadi, M. Slaney, and L. Heck. MSR Identity Toolbox v1.0: A MATLAB Toolbox for Speaker Recognition Research. Technical Report MSR-TR-2013-133, Microsoft Research Technical Report, September 2013.

[40] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[41] The HDF Group. Hierarchical Data Format, version 5, 1997-2017. `http://www.hdfgroup.org/HDF5/`.

[42] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[43] R. Vogt and S. Sridharan. Explicit modelling of session variability for speaker verification. *Computer Speech & Language*, 22(1):17–38, 2008.

[44] S.-C. Yin, R. Rose, and P. Kenny. Adaptive score normalization for progressive model adaptation in text independent speaker verification. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4857–4860. IEEE, 2008.

[45] S. Young and S. Young. The HTK Hidden Markov Model Toolkit: Design and Philosophy. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44, 1994.

[46] X.-L. Zhang and J. Wu. Denoising deep neural networks based voice activity detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 853–857. IEEE, 2013.