Ben-Gurion University of the Negev
The Faculty of Engineering
The Department of **Electrical and Computer Engineering**

Domain Adaptation and Speaker Diarization for Speaker Recognition

Nave Algarici

Thesis submitted in partial fulfillment of the requirements
for the Master of Sciences degree

Under the supervision of **Prof. Haim Permuter**

February 2019

Ben-Gurion University of the Negev
The Faculty of Engineering
The Department of **Electrical and Computer Engineering**

Domain Adaptation and Speaker Diarization for Speaker Recognition

Nave Algarici

Thesis submitted in partial fulfillment of the requirements
for the Master of Sciences degree

Under the supervision of **Prof. Haim Permuter**

Signature of student: _____ Date: _____
Signature of supervisor: _____ Date: _____
Signature of chairperson of the
committee for graduate studies: _____ Date: _____

February 2019

Abstract

The task of speaker recognition is considered. In this task, the goal is, given two segments of speech, to decide whether the same speaker uttered both segments or different speakers did. In many cases, the segments domain (speaker's gender, origin, language, etc.) do not match the data the speaker recognition was trained on. Another common scenario is when a segment contains speech from multiple speakers (such as in YouTube videos), which makes the task more difficult.

In this thesis, both scenarios of domain mismatch and multiple-speaker segments are addressed. The suggested approaches are PLDA adaptation to decrease domain mismatch and an unsupervised X-vector based speaker diarization algorithm. These approaches were tested in the NIST 2018 speaker recognition evaluation, where they achieved a significant performance improvement, compared to the baseline algorithm, on both development and evaluation sets.

Acknowledgements

First, I would like to thank my adviser, Prof. Haim Permuter, for his input, support and dedication throughout our mutual work towards this thesis.

I would like to thank my research group, Ziv, Gal and Ron, for there support, both professionally and mentally. Couldn't have done it without them.

Last but not least, I would like to thank my parents, for there endless support and encouragement, for helping me get to where I am, and for making me who I am.

List of Publications

- Nave Algarici, Haim Permuter. "The BGU 2018 NIST Speaker Recognition Evaluation System". In Proceedings of the NIST 2018 speaker recognition evaluation workshop. 2018.

Contents

List of Figures

List of Tables

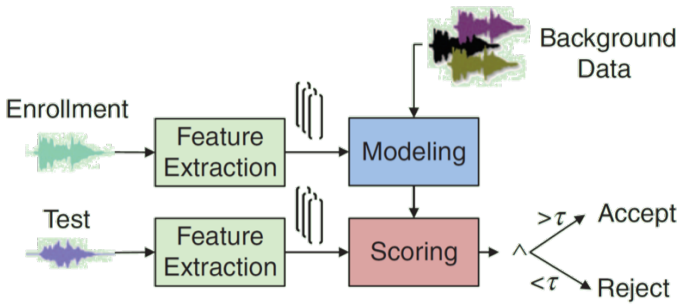
1 Introduction

1.1 Problem Definition

Acoustic speech signals contain large amounts of information to listeners. Some of it is related to the message of the speech it self, but speech also conveys information about the language being spoken, additionally information relating to the emotion, gender, and the identity of the speaker. The goal of a speaker recognition system is to take all the information contained in a speaker's voice to recognize their identity [1]. The task of speaker verification is usually differentiated from the task of speaker identification. While speaker identification attempts to identify an unknown speaker from a group of speaker models, speaker verification requires that the unknown speaker claims an identity and only requires that the identity be accepted or rejected. To put it simply, we can describe the speaker verification system as such:

Two pieces of speech are presented to the system. The system must decide whether the same speaker uttered both segments or different speakers did. This is often referred to Open-Set Identification [2].

Figure 1.1: Diagram of the generic speaker verification system.



Speaker verification can be used in security and access control applications, such as identification in personal devices.

1.2 Thesis Outline

In this introduction, the definition of the speaker recognition problem is presented.

In chapter 2, the background knowledge for understanding the speaker recognition problem is presented, the major components of a speaker recognition systems and the challenges of speaker recognition.

In chapters 3, 4 and 5, the algorithms for speaker diarization, speaker modeling and for scoring and adaptation, that were the focus of this thesis, are presented.

In chapter 6, the NIST speaker recognition evaluation is presented, along with the suggested systems and there performance on the 2018 development and evaluation sets.

Chapter 8, includes a summary of the work done, along with suggested future work.

2 Background

2.1 Components of a speaker recognition system

The speaker recognition system can be defined by combining three main components:

1. Front-end processing.
2. Speaker modeling.
3. Scoring.

2.1.1 Front-end processing

Front-end processing is where the audio input is processed to produce features that rare useful for speaker recognition. This process consists of three sub-processes: Voice Activity Detection (VAD), feature extraction and Speaker Diarization.

First, the input speech is processed by a VAD system that ensures that the verification is only performed when speech is occurring. For example in an energy-based VAD approach, time frames with higher energy are retained, and the remainder are removed from the feature set [3].

Next, feature extraction is used to convert the raw speech signal into a sequence of acoustic feature vectors, which contain information about the signal, and can be used to identify the speaker. There are a number of feature extraction techniques available, including mel-scale frequency cepstral coefficients (MFCC) and linear-scale frequency cepstral coefficients (LFCC). These features are based on the spectral information derived from a short time windowed segment of speech, and they mainly differ by the

detail in the power spectrum representation. The most commonly chosen feature extraction technique for the state of the art speaker verification systems is MFCC [4], as this feature representation has been shown to provide better performance than other approaches. The MFCC features are calculated through pre-emphasis filtering, framing and windowing, triangular filtering and discrete cosine transform (DCT). Time derivatives of the MFCC coefficients are used as additional features, and are generally appended to each feature to capture the dynamic properties of the speech signal.

The final stage of the front-end processing is speaker diarization which aims to partition the input stream into parts uttered by a single speaker and also determine the regions spoken by each individual speaker. This stage is one of the main focuses of this thesis, and will be discussed in-depth in section 3.

2.1.2 Speaker Modeling

Once the audio segments are converted to feature parameters, the next component of the speaker-recognition process is modeling. We can define modeling as a process of describing the feature properties for a given speaker [5]. The model must also provide means of its comparison with an unknown utterance. A modeling method is robust when its characterizing process of the features is not significantly affected by unwanted distortions, even though the features are. Most speaker-modeling techniques make various mathematical assumptions on the features (Gaussian distributed, for example). These models are trained using large quantities of background data. Consequently, mathematical models are forced to fit the features and recognition scores are derived based on these models and test data.

In Section 4 we will discuss a few common approaches, and the X-vector state of the art approach.

2.1.3 Scoring

Every speaker recognition system, produces a probabilistic score for speaker recognition. Formally, given an observation O and a hypothesized speaker

s, the task of speaker verification can be stated as a hypothesis test between

$$H_0 : \text{O is from speaker s,} \tag{2.1}$$
$$H_1 : \text{O is not from speaker s.} \tag{2.2}$$

For the set of observed feature vectors $X = \{x_n\}_{n=1}^{N}$, the likelihood-ratio (LR) test is performed by evaluating the following ratio:

$$LR = \frac{\mathrm{P}\left(X|H_s\right)}{\mathrm{P}\left(X|H\right)}. \tag{2.3}$$

In sections 4 and 5 we will discuss how speaker modeling and scoring approaches come together to construct a speaker recognition system.

2.2 Speaker recognition challenges

Unlike other forms of biometrics (e.g., fingerprints, irises, facial features, and hand geometry), human speech is a performance biometric. Simply put, the identity information of the speaker is primarily embedded in how speech is spoken, not necessarily in what is being said. This makes speech signals prone to a large degree of variability. It is important to note that even the same person does not say the same words in exactly the same way every time (this is known as inter-speaker variability). For example, we may find it difficult to recognize someone's voice through a telephone or maybe when the person is not healthy (i.e., has a cold) or is performing another task or speaking with a different level of vocal effort (i.e., whispering or shouting).

Also, various recording devices and transmission methods commonly used emphasize the problem. Recording speech with different devices and in different acoustic environments, cause high levels of variability that are not caused by the speaker's voice.

Further more, when training a speaker recognition system, we usually don't have large amounts of speech recordings of every target speaker we wish to identify, but only a handful of utterance. This makes the task of capturing the speaker variability very difficult.

This work addresses these challenges discusses how we deal with them, overcome them, and achieve high accuracy in identification.

3 Speaker Diarization

3.1 Overview

Speaker diarization is the task that answers the question ”who spoken when?”. Recognition tasks mainly depend on being fed single speaker speeches and would fail if more than one speaker’s speech is present. Many sound streams are multi-speaker streams, such as conference calls and audio from YouTube videos. The speaker diarization task aims to partition the input stream into parts uttered by a single speaker and also determine the regions spoken by each individual speaker [6].

Speaker diarization is thought of as a two-step task:

1. **Speaker segmentation**: the task of determining turn points in a stream. That is where the change of the speaker occurs. The output is a stream of segments, each uttered by a single speaker, and no two adjacent segments uttered by the same speaker.

2. **Speaker clustering**: the task of assigning each segment to its speaker, so that it can be determined what parts were spoken by each speaker.

3.2 Algorithm

3.2.1 Voice Activity Detection for Speaker Segmentation

Voice Activity Detection (VAD) is used to separate speech parts from non-speech parts. Some research tries to handle non-speech as an extra speaker, and therefore eliminates the need for an explicit VAD process. However, experiments have shown that having a separate VAD improves the overall performance of the system [7].

Energy Based VAD

This method assumes that non-speech frames have lower energy than speech frames. This assumption is true in case of silences. This approach is very simple to implement and fast to run, however, it fails to detect other types of non-speech, especially overlapping speech.

A window is slid across the stream and the energy for each window is computed. Either an energy threshold is set for decision making. Or, if the speech portion of the whole stream is known a priori (as a percentage of the stream length), then windows are sorted in a decreasing order according to their energies. Later, the top percentage of the sorted windows are classified as speech and thus retained, while the rest of the windows are discarded as being non-speech.

This method is parametric (needs a threshold parameter to be set manually). Even if the best parameters were set, other non-speech activity (laughter, overlapping speech, etc...) will still be audible on the output stream. Another issue is that the classified frames may be interleaved, and the end stream would miss parts in the middle of the speech (e.g. unvoiced segments) if they were classified as non-speech. For that, it is possible to perform a smoothing step for the filter.

3.2.2 Segment Modeling

Each speech segment detected using the VAD algorithm, must be presented as a speaker model, for further comparison and clustering. For segment modeling, one of the methods presented in chapter 4 are well suited for this task.

Given the speaker models for each speech segment, a Distance measure can be calculated between every pair of segments. In the case of using a GMM model, a log-likelihood ratio can be calculated from equation 4.20. In the more recent methods of i-vector and x-vector embeddings, a PLDA score can be used as detailed in chapter 5.

3.2.3 Speaker Clustering

Given similarity scores for each pair of segments, modern clustering algorithms can be applied to cluster segments together. For the diarization task, the well-known Agglomerative Hierarchical Clustering (AHC) algo-

rithm is commonly used [8].

AHC in a simple and greedy algorithm that works in a bottom-up fashion:

1. Each segment embedding is set as its own cluster.
2. Iteratively, pairs of clusters are merged, one pair at a time, by a cluster similarity metric, until a stopping criterion is met (e.g. number of clusters, maximum distance, etc.).

4 Speaker Modeling

4.1 GMM

4.1.1 Overview

Gaussian mixture models (GMMs) were proposed by Reynolds et al. [9] to model the speaker, and they perform very effectively in speaker verification systems. A GMM is a weighted sum of M component Gaussian densities as given by the equation,

$$P\left(\mathbf{x}|\lambda\right) = \sum_{k=1}^{M} w_k \cdot \mathcal{N}\left(x|\mu_k, \Sigma_k\right), \tag{4.1}$$

where $\mathbf{x}$ is a $\mathbf{D}$-dimensional feature vector, w_k, μ_k, Σ_k, $k = 1, 2, \ldots, M$, are the mixture weights, means and covariances respectively. $\mathcal{N}\left(x|\mu_k, \Sigma_k\right)$, $k = 1, 2, \ldots, M$, are the component Gaussian densities.

Each component density is a $\mathbf{D}$-variate Gaussian function of the form,

$$\mathcal{N}\left(x|\mu_k, \Sigma_k\right) = \frac{1}{\left(2\pi\right)^{\frac{D}{2}} \left|\Sigma_k\right|^{\frac{1}{2}}} \exp\left(-\frac{\left(x-\mu_k\right)^T \Sigma_k^{-1} \left(x-\mu_k\right)}{2}\right), \tag{4.2}$$

where the mixture weights satisfy the constraint $\sum_{k=1}^{M} w_k = 1$. The complete GMM is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities, and these parameters are collectively represented by $\lambda = \{w_k, \mu_k, \Sigma_k\}$, $k = 1, 2, \ldots, M$.

4.1.2 Expectation maximization for GMM

An expectation-maximization (EM) algorithm is an iterative method for finding maximum likelihood estimates of parameters in statistical models. The EM algorithm is used to learn the GMM parameters, based on

maximizing the expected log-likelihood of the training data. The motivation of the EM algorithm is to estimate a new and improved model λ from the current model λ_0 using the training utterance features $\{x_n\}_{n=1}^N$ such that the probability

$$\Pi_{n=1}^N \mathrm{P}\left(x_n|\lambda\right) \geq \Pi_{n=1}^N \mathrm{P}\left(x_n|\lambda_0\right) \tag{4.3}$$

This is an iterative technique where the new model becomes the current model for the following iteration.

The initial GMM parameters are typically defined using the k-means algorithm. The k-means algorithm is also based on an iterative approach in which the mixture of training feature vectors is performed through the estimation of mixture means.

The EM algorithm attempts to maximize the auxiliary function $Q\left(\lambda;\lambda_0\right)$. This is implemented using Jensen's inequality ensuring 4.3. The auxiliary function can be formulated as

$$Q\left(\lambda;\lambda_0\right) = \sum_{n=1}^{N}\sum_{k=1}^{M} \mathrm{P}(k|x_n,\lambda_o)\log\left(w_k \cdot \mathcal{N}\left(x|\mu_k,\Sigma_k\right)\right), \tag{4.4}$$

where $\mathrm{P}(k|x_n,\lambda_o)$ forms the E step for producing observation x using

$$\mathrm{P}(k|x,\lambda_o) = \frac{w_k \cdot \mathcal{N}\left(x|\mu_k,\Sigma_k\right)}{\mathrm{P}\left(x|\lambda_0\right)}. \tag{4.5}$$

The M step then sees the auxiliary function is maximized using Equation 4.4 This maximization results in the GMM parameters being estimated as

$$w_k = \frac{1}{N}\sum_{n=1}^{N} \mathrm{P}(k|x_n,\lambda_o), \tag{4.6}$$

$$\mu_k = \frac{\sum_{n=1}^{N} \mathrm{P}(k|x_n,\lambda_o)x_n}{\sum_{n=1}^{N} \mathrm{P}(k|x_n,\lambda_o)}, \tag{4.7}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \mathrm{P}(k|x_n,\lambda_o)\left(x_n-\mu_k\right)\left(x_n-\mu_k\right)^T}{\sum_{n=1}^{N} \mathrm{P}(k|x_n,\lambda_o)}. \tag{4.8}$$

4.1.3 Universal background model (UBM)

In typical speaker verification tasks, there is a limited amount of data available to train each speaker model, thus the speaker models can't be directly estimated reliably with the EM algorithm. For this reason, *maximum*

a posteriori (MAP) adaptation [10] is often used to train the speaker models for speaker verification systems. This approach estimates the speaker model from the universal background model (UBM). A UBM is a high-order GMM, trained on a large quantity of speech obtained from a wide sample of the speaker population of interest, and is designed to capture the general form of a speaker model and represents the speaker-independent distribution of features. The UBM parameters are estimated using the EM algorithm described in the previous section.

4.1.4 MAP adaptation speaker enrollment

In MAP adaptation, the speaker model is derived from the UBM by considering specific speaker vectors. As a variant of the EM algorithm, first initializing the speaker models with the parameters of UBM iteratively updates the parameters of the GMM $\lambda = \{w_k, \mu_k, \Sigma_k\}$ such that total likelihood for an enrolment utterance $x_1, x_2, \ldots, x_N$. is maximized.

The updated model is then used as the initial model for the next iteration. The process is repeated until some convergence threshold is reached. For each iteration of the EM algorithm, the expressions of the maximum likelihood (ML) estimates of the GMM parameters, which guarantee a monotonic increase of the model's likelihood, are as described in the previous section.

Given a GMM-UBM and enrollment speaker data $x_1, x_2, \ldots, x_N$, at first the probabilistic alignment of the feature vectors with respect to the UBM components is calculated as

$$\mathrm{P}(k|x, \lambda_o) = \frac{w_k \cdot \mathcal{N}(x|\mu_k, \Sigma_k)}{\mathrm{P}(x|\lambda_0)} = \gamma_n(k). \tag{4.9}$$

Next, the values of $\gamma_n(k)$ are used to calculate the sufficient statistics fro the weight, mean and covariance parameters as

$$N_s(k) = \Sigma_{n=1}^{N} \gamma_n(k), \tag{4.10}$$

$$F_s(k) = \Sigma_{n=1}^{N} \gamma_n(k) x_n, \tag{4.11}$$

$$S_s(k) = \Sigma_{n=1}^{N} \gamma_n(k) x_n x_n^T. \tag{4.12}$$

These quantities are known as the zero-, first-, and second-order Baum–Welch statistics, respectively. Using these parameters, the posterior mean and co-

variance matrix of the features given the data vectors **X** can be found as

$$\mathrm{E}_k\left[x_n|X\right] = \frac{F_s(k)}{N_s(k)} \tag{4.13}$$

$$\mathrm{E}_k\left[x_n x_n^T|X\right] = \frac{S_s(k)}{N_s(k)} \tag{4.14}$$

The maximum a posteriori (MAP) adaptation update equations for weight, mean, and covariance are

$$\hat{w}_k = \left[\alpha_k N_s(k)/N + (1-\alpha_k)\, w_k\right]\beta, \tag{4.15}$$

$$\hat{\mu}_k = \alpha_k \mathrm{E}_k\left[x_n|X\right] + (1-\alpha_k)\,\mu_k, \tag{4.16}$$

$$\hat{\Sigma}_k = \alpha_k \mathrm{E}_k\left[x_n x_n^T|X\right] + (1-\alpha_k)\left(\Sigma_k + \mu_k\mu_k^T\right) - \hat{\mu}_k\hat{\mu}_k^T. \tag{4.17}$$

The scaling factor β in 4.15 is computed from all the adapted mixture weights to ensure that they sum to unity. The variable α_k is defined as

$$\alpha_k = \frac{N_s(k)}{N_s(k)+r}, \tag{4.18}$$

where r is known as the relevance factor. This parameter controls how the adapted GMM parameter will be affected by the observed speaker data.

4.1.5 GMM based speaker verification

In the GMM–UBM approach, the hypothesis H_0 and H_1 from 2.1 and 2.2 are represented by a speaker-dependent GMM λ_s and the UBM λ_0. Thus, for the set of observed feature vectors $X = \{x_n\}_{n=1}^N$, the likelihood-ratio (LR) test is performed by evaluating the following ratio:

$$\frac{\mathrm{P}\left(X|\lambda_s\right)}{\mathrm{P}\left(X|\lambda_0\right)} \begin{cases} \geq \tau & \text{accept } H_0 \\ < \tau & \text{reject } H_0 \end{cases}, \tag{4.19}$$

where τ is the decision threshold. Usually, the LR test is performed in the logarithmic scale, providing the log-LR

$$\Lambda(X) = \log \mathrm{P}\left(X|\lambda_s\right) - \log \mathrm{P}\left(X|\lambda_0\right). \tag{4.20}$$

4.1.6 GMM supervectors

One of the issues with speaker recognition is that the training and test speech data can be of different durations. Obtaining a fixed-dimensional utterance-level features is beneficial, for further use of machine learning classifiers. One effective solution is the formulation of a GMM supervector which is essentially a large vector obtained by concatenating the parameters of a GMM model. Generally, a GMM supervector is obtained by concatenating the GMM mean vectors of a MAP-adapted speaker model.

4.2 I-Vector

4.2.1 Overview

Revisiting the MAP adaptation technique discussed previously in the GMM–UBM system. Examining the adaptation equation 4.16, which is used to update the mean vectors, it is clear that this is a linear combination of two components: one is speaker dependent and the other is independent. In a more generalized way, MAP adaptation can be represented as an operation on the GMM mean supervector as:

$$m_s = m_0 + Uz_s, \tag{4.21}$$

where U is $(MD \times MD)$ a diagonal matrix and z_s is a $MD \times 1$ standard normal random vector. In order to provide more robustness to the session and inter-speaker variability than the traditional MAP approach, a Factor Analysis (FA) of the GMM supervectors approach rose, aiming at describing the variability in high-dimensional observable data vectors using a lower number of hidden variables [11]. Many variants of FA methods were employed since then, which finally led to the i-vector approach [12].

In the i-vector model, observing the fact that the channel factors also contain speaker-dependent information, the speaker and channel factors were combined into a single space termed the *total variability space*. In this FA model, a speaker and session dependent GMM supervector is represented by

$$m_s = m_0 + Tw_s. \tag{4.22}$$

The hidden variables $w_s \sim N(0, I)$ in this case are called *total factors*. The hidden variables are not observable but can be estimated by their posterior expectation. The estimates of the total factors, which can be used as

features, came to be known as the i-vectors. Unlike other FA methods, the i-vector approach does not make a distinction between speaker and channel. It is simply a dimensionality reduction method of the GMM supervector. In essence, 4.23 is very similar to a PCA model on the GMM supervectors.

4.2.2 Expectation maximization for i-vector

Suppose we have a sequence y_s of L frames $\{y_{s,1}, y_{s,2}, \ldots, y_{s,L}\}$ and an UBM with parameters λ composed of M mixture components defined in a D-dimensional feature space. Dividing equation 4.23 into M mixture components we get

$$m_{s,k} = m_{0,k} + T_k w_{s,k}. \tag{4.23}$$

Assuming the latent variable w's prior follows a Gaussian distribution with a 0-mean and identity covariance, each frame is aligned to a fixed mixture component, and the factor loading submatrix T_k is known, the posterior distribution can be estimated as

$$\mathrm{P}\left(w|y_s\right) = \mathcal{N}\left(w; L_s^{-1} \sum_{k=1}^{M} T_k^* \Sigma_k^{-1} F_s(k), L_s^{-1}\right), \tag{4.24}$$

where the precition Matrix $L_s \in \mathbf{R}^{D\times D}$ is

$$L_s = I + \sum_{k=1}^{M} N_s(k) T_k^* \Sigma_k^{-1} T_k, \tag{4.25}$$

and the Baum–Welch statistics needed are

$$N_s(k) = \sum_{n=1}^{L} \gamma_n(k), \tag{4.26}$$

$$F_s(k) = \sum_{n=1}^{L} \gamma_n(k) y_{s,n}, \tag{4.27}$$

with $\gamma_n(k)$ is the same as in 4.9. The i-vector is simply the MAP point estimate (the mean point of the posterior distribution 4.24 of the variable w

$$w = L_s^{-1} \sum_{k=1}^{M} T_k^* \Sigma_k^{-1} F_s(k). \tag{4.28}$$

To estimate $\{T_k\}_{k=1}^{M}$, the EM algorithm is used to maximize the ML training criterion

$$Q\left(T_1,\ldots,T_k\right) = -\frac{1}{2}\sum_{s,n,k}\gamma_{s,n}(k)\left[\log|L_s| + (y_{s,n}-\mu_s(k))^T\Sigma_k^{-1}(y_{s,n}-\mu_s(k))\right], \tag{4.29}$$

or equivalently

$$Q\left(T_1,\ldots,T_k\right) = -\frac{1}{2}\sum_{s,k}[\gamma_s(k)\log|L_s| + \gamma_s(k)\,Tr\{\Sigma_k^{-1}T_k w_s w_s^T T_k^T\} \tag{4.30}$$

$$- 2\,Tr\{\Sigma_k^{-1}T_k w_s F_s(k)^T\}] + C. \tag{4.31}$$

The result of maximizing equation 4.31 is the M-step

$$T_k = C_k A_k^{-1}, \quad 1 \le k \le M \tag{4.32}$$

where

$$C_k = \sum_s F_s(k) w_s^T, \tag{4.33}$$

$$A_k = \sum_s N_s(k)\left[L_s^- 1 + w_s w_s^t\right] \tag{4.34}$$

that are computed in the E-step.

4.2.3 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a commonly employed technique in statistical pattern recognition that aims at finding linear combinations of feature coefficients to facilitate discrimination of multiple classes. It finds orthogonal directions in the feature space that are more effective in discriminating the classes. Projecting the original features in these directions improve classification accuracy.

Let D indicate the set of all development utterances, $w_{s,i}$ indicates an utterance feature (or i-vector) obtained from the ith utterance of speaker s, n_s denotes the total number of utterances belonging to speaker s, and S is the total number of speakers in D. The between-and within-class covariance matrices are given by

$$S_b = \frac{1}{S}\sum_{s=1}^{S}\left(\bar{w}_s - \bar{w}\right)\left(\bar{w}_s - \bar{w}\right)^T \quad \text{and} \tag{4.35}$$

$$S_w = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{n_s}\sum_{i=1}^{n_s}\left(\bar{w}_{s,i} - \bar{w}_s\right)\left(\bar{w}_{s,i} - \bar{w}_s\right)^T, \tag{4.36}$$

where the speaker-dependent and speaker-independent mean vectors are given by

$$\bar{w}_s = \frac{1}{n_s} \sum_{i=1}^{n_s} w_{s,i} \tag{4.37}$$

$$\bar{w} = \frac{1}{S} \sum_{s=1}^{S} \frac{1}{n_s} \sum_{i=1}^{n_s} w_{s,i}, \tag{4.38}$$

respectively. The LDA optimization aims at maximizing the between class variance while minimizing the within-class variance (due to channel variability). The projections obtained from this optimization are found by the solution of the following generalized eigenvalue problem:

$$S_b v = \Lambda S_w v. \tag{4.39}$$

Here, Λ is the diagonal matrix containing the eigenvalues. If the matrix S_w is invertable, this solution can be found by finding the eigen-values of the matrix $S_w^{-1} S_b$. Generally,the first $k < R$ eigen-values are used to prepare a matrix A_{LDA} of dimension $R \times k$ given by

$$A_{LDA} = [v_1 \dots v_k], \tag{4.40}$$

where $v_1 \dots v_k$ denote the first k eigen-vectors obtained by solving 4.39. The LDA transformation of the utterance feature w is thus obtained by

$$\hat{w}_{LDA} = A_{LDA}^T w. \tag{4.41}$$

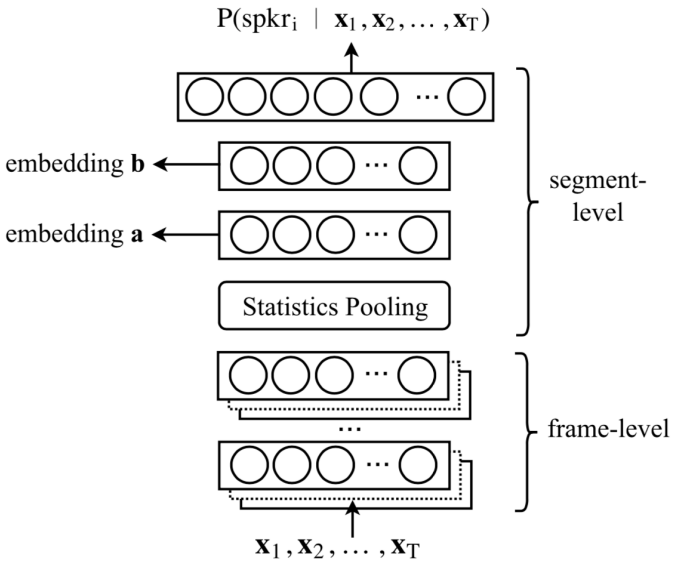
4.3 X-Vector

4.3.1 Overview

Given recent success in other areas of speech technology, utilizing deep neural networks (DNN) for speaker recognition was a natural progression. For the purpose of replacing i-vectors with embeddings produced by a DNN, the x-vector method was developed [13].

This method uses a feed-forward DNN that computes speaker embeddings from variable-length acoustic segments. An end-to-end approach requires a large amount of in-domain data to be effective. Replacing the end-to-end loss with a multi-class cross entropy objective.

Figure 4.1: Diagram of the x-vector DNN.



4.3.2 Architecture

The DNN configuration is outlined in Table 4.1. Suppose an input segment has T frames. The first 5 layers of the network work at the frame level, with a time-delay architecture [26]. Suppose t is the current time step. At the input, we splice together frames at $\{t-2, t-1, t, t+1, t+2\}$. The next two layers splice together the output of the previous layer at times $\{t-2, t, t+2\}$ and $\{t-3, t, t+3\}$, respectively. The next two layers also operate at the frame-level, but without any added temporal context. In total, the frame- level portion of the network has a temporal context of $t-8$ to $t+8$ frames. The statistics pooling layer aggregates all T outputs from the final frame-level layer and computes its mean and standard deviation. The statistics are 1500 dimensional vectors, computed once for each input segment. This process aggregates information across the time dimension so that subsequent layers operate on the entire segment. In Table 4.1, segment level layers are denoted by a layer context of 0 and a total context of T. These segment- level statistics are concatenated together and passed to the segment level layers, and finally the soft-max output layer. The nonlinearities are all rectified linear units (ReLUs).

The goal of training the network is to produce embeddings that generalize well to speakers that have not been seen in the training data. We would

like embeddings to capture speaker characteristics over the entire utterance, rather than at the frame-level. The pre-softmax affine layer is not used because of its large size and dependence on the number of speakers. The other two affine layers are those from which the embeddings can be extracted. These are depicted in Figure 4.3.1 as embeddings **a** and **b**. Embedding **a** is the output of an affine layer directly on top of the statistics. Embedding **b** is extracted from the next affine layer after a ReLU, and so it is a nonlinear function of the statistics.

Table 4.1: The x-vector embedding DNN architecture

Layer	Layer context	Total context	Input x output
frame1	[t-2,t+2]	5	120x512
frame2	{t-2,t,t+2}	9	1536x512
frame3	{t-3,t,t+3}	15	1536x512
frame4	{t}	15	512x512
frame5	{t}	15	512x1500
stats pooling	[0,T)	T	1500Tx3000
segment6	{0}	T	3000x512
segment7	{0}	T	512x512
softmax	{0}	T	512xN

4.3.3 Training

The network is trained to classify training speakers using a multi-class cross entropy objective function. The primary difference of this system is that it is trained to predict speakers from variable-length segments, rather than frames. Suppose there are K speakers in N training segments. Then $\mathrm{P}\left(spkr_k|\{x_t^{(n)}\}_{t=1}^T\right)$ is the probability of speaker k given T input frames $\{x_1^{(n)}, x_2^{(n)}, \ldots, x_T^{(n)}\}$. The quantity d_{nk} is 1 if the speaker label for segment n is k, otherwise it's 0.

$$\mathrm{Q} = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} \log\left(\mathrm{P}\left(spkr_k|\{x_t^{(n)}\}_{t=1}^T\right)\right) \tag{4.42}$$

5 PLDA Scoring and Domain Adaptation

5.1 Probabilistic Linear Discriminant Analysis

5.1.1 Overview

Probabilistic Linear Discriminant Analysis (PLDA) is a probabilistic approach that models both between-class and inter-class variance as multi-dimensional Gaussians. It seeks directions in space that have maximum discriminability and are hence most suitable for supporting class recognition tasks. We assume that the training data consists of J utterances each of I speaker. We denote the j'th utterance of the i'th speaker by x_{ij} . We model data generation by the process:

$$x_{ij} = \mu + Fh_i + Gw_{ij} + \epsilon_{ij}. \tag{5.1}$$

This model comprises two parts:

1. The signal component - $\mu + Fh_i$ which depends only on the identity of the speaker but not the particular utterance (there is no dependence on j). This describes between-speaker variation.

2. The noise component - $Gw_{ij} + \epsilon_{ij}$ which is different for every utterance of the speaker and represents within-speaker noise.

The term μ represents the overall mean of the training dataset. The columns of the matrix F contain a basis for the between-speaker subspace and the term h_i represents the position in that subspace. The matrix G contains a basis for the within-speaker subspace and w_{ij} represents the position in this subspace. Remaining unexplained data variation is explained by

the residual noise term ϵ_{ij} which is defined to be Gaussian with diagonal covariance Σ. These parameters are collectively represented by $\theta = \{\mu, F, G, \Sigma\}$. The term h_i is particularly important as this represents the identity of speaker i. This is the *latent speaker variable*: in recognition we will consider the likelihood that two utterances were generated from the same underlying h_i. Formally, the model in Equation 5.1 can be described in terms of conditional probabilities:

$$\mathrm{P}\left(x_{ij}|h_i, w_{ij}, \theta\right) = \mathcal{N}_x\left[\mu + Fh_i + Gw_{ij}, \Sigma\right], \tag{5.2}$$

$$\mathrm{P}\left(h_i\right) = \mathcal{N}_x\left[0, 1\right], \tag{5.3}$$

$$\mathrm{P}\left(w_{ij}\right) = \mathcal{N}_x\left[0, 1\right]. \tag{5.4}$$

Gaussian PLDA

Since the speaker embeddings are of sufficiently low dimension, the modification proposed in [14] assumes that Σ is a full covariance matrix and remove the within-speaker values from equation 5.1. The modified Gaussian PLDA (G-PLDA) model follows:

$$x_{ij} = \mu + Fh_i + \epsilon_{ij}. \tag{5.5}$$

5.1.2 Training

The goal in training the PLDA model, is to find the parameters θ under which the data is most likely. There is a well known solution for finding these parameters. The EM algorithm in [14] estimates the the parameters in such way that the likelihood is guaranteed to increase at each iteration. In the E-step, the full posterior distribution over the latent variables h_i and w_{ij} is calculated. In the M-step, the point estimates of parameters θ are optimized.

5.1.3 Scoring

For a speaker verification task, given the speaker embeddings η_1 and eta_2 involved in a trial, two alternative hypotheses are tested: $\mathcal{H}_s$ that both and share the same speaker identity latent variable h, or $\mathcal{H}_d$ that the speaker embeddings were generated using different identity variables h_1 and h_2.The verification score can now be computed as the log- likelihood ratio for this

hypothesis test as

$$score = \log \frac{\mathrm{P}\left(\eta_1, \eta_2 | \mathcal{H}_s\right)}{\mathrm{P}\left(\eta_1 | \mathcal{H}_d\right) \mathrm{P}\left(\eta_2 | \mathcal{H}_d\right)}. \tag{5.6}$$

For the G-PLDA case this log-likelihood ratio is easily computed in closed-form solution since the marginal likelihoods are Gaussian.

$$score = \log \mathcal{N}\left(\begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}; \begin{bmatrix} \mu \\ \mu \end{bmatrix}, \begin{bmatrix} \Sigma_{tot} & \Sigma_{ac} \\ \Sigma_{ac} & \Sigma_{tot} \end{bmatrix}\right) - \log \mathcal{N}\left(\begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}; \begin{bmatrix} \mu \\ \mu \end{bmatrix}, \begin{bmatrix} \Sigma_{tot} & 0 \\ 0 & \Sigma_{tot} \end{bmatrix}\right), \tag{5.7}$$

where

$$\Sigma_{tot} = FF^T + \Sigma \quad \text{and} \tag{5.8}$$

$$\Sigma_{ac} = FF^T. \tag{5.9}$$

By pre-computing the global offset and removing it from all speaker embeddings, μ is set to 0, and the computation comes down to

$$score = \eta_1^T Q \eta_1 + \eta_2^T Q \eta_2 + 2\eta_1^T P \eta_2 + const, \tag{5.10}$$

with

$$Q = \Sigma_{tot}^{-1} - \left(\Sigma_{tot} - \Sigma_{ac}\Sigma_{tot}^{-1}\Sigma_{ac}\right)^{-1}, \tag{5.11}$$

$$P = \Sigma_{tot}^{-1}\Sigma_{ac}\left(\Sigma_{tot} - \Sigma_{ac}\Sigma_{tot}^{-1}\Sigma_{ac}\right)^{-1}. \tag{5.12}$$

$$\tag{5.13}$$

5.2 Domain Adaptation

5.2.1 Overview

A systems speaker recognition performance is highly influenced by the data of which it is trained on. Using a trained system for speaker recognition of utterances from a specific domain (referred to as in-domain data), that differs from the data it was trained on (out-of-domain data), leads to poor performance on the in-domain data. To deal with this shortcoming, the original out-of-domain system must be adapted to the new domain.

When porting a system to a new domain, there are three options:

1. Assume the new domain data is sufficiently close to the data used to train the hyper-parameters that the system will work well.

2. Collect a large amount of unlabeled data from the new domain and adapt the hyper-parameters using unsupervised techniques.

3. Collect and label sufficient amounts of new domain data to allow retraining or supervised adaptation of the hyper-parameters.

A PLDA scoring function between a model and test speaker embedding, requires a within-class (WC) matrix G, characterizing how speaker embeddings from a single speaker vary, and an across class (AC) matrix F, characterizing how speaker embeddings between different speakers vary. The WC and AC matrices, require labeled data to learn within speaker and across speaker variabilities. While all hyper-parameters are susceptible to mismatch, those requiring labeled data to train are more difficult to handle.

5.2.2 PLDA Adaptation

Given a PLDA scoring function with WC and AC hyper-parameters G_{out} and F_{out}, trained on out-of domain data, and a unlabeled of in-domain speaker embeddings, the goal is to adapt the PLDA hyper-parameters, to better represent the in-domain data variabilities, and improve speaker recognition performance. Thus, the following approach is followed:

1. Use G_{out} and F_{out} to compute a pairwise affinity matrix A, on the unlabeled in-domain data. Element A_{ij} in this matrix is a PLDA score as detailed in section 5.1.3 between speaker embeddings i and j, representing the log likelihood ratio between the hypothesis that they are from the same speaker and the hypothesis that they come from different speakers.

2. Use A to obtain a hypothesized speaker clustering of the in-domain data. Current best practice is to use the *agglomerative hierarchical clustering* algorithm [8].

3. These estimated speaker clusters are used to train a new PLDA scoring function hyper-parameters G_{in} and F_{in}.

4. Linearly interpolate between the prior out-of-domain and new in-domain covariance matrices to obtain the final hyper-parameters:

$$G_{final} = \alpha_{WC} \cdot G_{out} + (1 - \alpha_{WC}) \cdot G_{in} \tag{5.14}$$

$$F_{final} = \alpha_{AC} \cdot F_{out} + (1 - \alpha_{AC}) \cdot F_{in} \tag{5.15}$$

It also possible to iterate this procedure, where G_{final} and F_{final} obtained in step 4 respectively replace the G_{out} and F_{out} of step 1, and the process is repeated until conversion.

6 Results

6.1 NIST Speaker Recognition Evaluation 2018

6.1.1 Overview

The 2018 speaker recognition evaluation (SRE18) is the next in an ongoing series of speaker recognition evaluations conducted by the US National Institute of Standards and Technology (NIST) since 1996. The objectives of the evaluation series are (1) to explore promising new ideas in speaker recognition, (2) to support the development of advanced technology incorporating these ideas, and (3) to measure and calibrate the performance of the current state of technology.

SRE18 was focusing on speaker detection over conversational telephone speech (CTS) collected outside North America. In addition to CTS recorded over a variety of handsets (PSTN), voice over IP (VOIP) data, which is also collected outside North America, as well as audio from video (AfV) was included as development and test material in SRE18.

6.1.2 Task Description

Task Definition

The task for SRE18 is speaker detection: given a segment of speech and the target speaker enrollment data, automatically determine whether the target speaker is speaking in the segment. A segment of speech (test segment) along with the enrollment speech segment(s) from a designated target speaker constitute a trial. The system is required to process each trial independently and to output a log-likelihood ratio (LLR), defined at 4.20, for that trial.

Training conditions

The training condition is defined as the amount of data/resources used to build a Speaker Recognition (SR) system. The task described above can be evaluated over a *fixed* (required) or *open* (optional) training condition.

We chose to focus on the *fixed* training condition, which limits the system training to specific common data sets. For the *fixed* training condition, only specified speech data may be used for system training and development, to include all sub-systems, e.g., voice activity detection (VAD), and auxiliary systems used for automatic labels/processing. Publicly available, non- speech audio and data (e.g., noise samples, impulse responses, filters) may be used.

6.1.3 Data Description

The data collected by the LDC as part of the Call My Net 2 (CMN2) and Video Annotation for Speech Technology (VAST) corpora.

The CMN2 data are composed of PSTN and VOIP data collected outside North America, spoken in Tunisian Arabic. Recruited speakers (called claque speakers) made multiple calls to people in their social network (e.g., family, friends). Claque speakers were encouraged to use different telephone instruments (e.g., cell phone, landline) in a variety of settings (e.g., noisy cafe, quiet office) for their initiated calls and were instructed to talk for at least 8 minutes on a topic of their choice. All CMN2 segments will be encoded as a-law sampled at 8 kHz in SPHERE formatted files.

The VAST data are composed of audio extracted from YouTube videos that vary in duration from a few seconds to several minutes and include speech spoken in English. Each audio recording may contain speech from multiple talkers, therefore manually produced diarization labels (i.e., speaker time marks) will be provided for the enrollment cuts (but not for the test cuts). All VAST data will be encoded as 16-bit FLAC files sampled at 44 kHz.

6.1.4 Scoring Mechanism

A basic cost model is used to measure the speaker detection performance and is defined as a weighted sum of false-reject (missed detection) and

false-alarm error probabilities for some decision threshold θ as follows

$$C_{Det}(\theta) = C_{Miss} \cdot P_{Target} \cdot P_{Miss}(\theta) + C_{FalseAlarm} \cdot \left(1 - P_{Target}\right) \cdot P_{FalseAlarm}(\theta), \tag{6.1}$$

where the parameters of the cost function are C_{Miss} (cost of a missed detection) and $C_{FalseAlarm}$ (cost of a spurious detection), and P_{Target} (a priori probability of the specified target speaker) and are defined to have the following values for CTS and AfV source types:

Table 6.1: SRE18 cost parameters

Source Type	**Parameter ID**	C_{Miss}	$C_{FalseAlarm}$	P_{Target}
CTS	1	1	1	0.01
	2	1	1	0.005
AfV	3	1	1	0.05

To improve the interpretability of the cost function C_{Det} in 6.1, it will be normalized by the best cost that could be obtained without processing the input data. After normalization, the normalized cost is

$$C_{Norm}(\theta) = P_{Miss}(\theta) + \beta \cdot P_{FalseAlarm}(\theta), \tag{6.2}$$

where β is defined as

$$\beta = \frac{C_{FalseAlarm}}{C_{Miss}} \cdot \frac{1 - P_{Target}}{P_{Target}}. \tag{6.3}$$

Actual detection costs will be computed from the trial scores by applying detection thresholds of $\log(\beta)$. For trials involving the CTS (i.e., PSTN and VOIP) source type, threshold swill be computed for two values of β ,with β_1 for $P_{Target_1} = 0.01$ and β_2 for $P_{Target_2} = 0.005$,while for AfV trials a single threshold, $\log(3)$, will be computed for $P_{Target_3} = 0.05$. The primary cost measure for SRE18 is then defined as

$$C_{Primary} = \frac{1}{2}\left[\frac{C_{Norm_{\beta_1}} + C_{Norm_{\beta_2}}}{2} + C_{Norm_{\beta_3}}\right] \tag{6.4}$$

Also, a minimum detection cost will be computed by using the detection thresholds that minimize the detection cost.

6.2 SRE18 Submission

In our submitted speaker recognition, we exploited the advances of Deep Neural Networks in the field of speaker recognition. We use the framework of the X-vector extractor Detailed in 6.2.1 for speaker embedding, and for diarization. In addition, we adapt the PLDA using in-domain data to better fit it to the task. We have improved the EERs of the Call My Net 2 (CMN2) and the Video Annotation for speech Technology (VAST) data sets by 16% and 28% respectively, in comparison to the best results published in the 2018 baseline systems.

6.2.1 System Description

Acoustic features

The features are 23 MFCCs with a frame length of 25ms every 10ms using a 23 channel mel-scale filterbank spanning the frequency range 20Hz-3700Hz. These feature vectors are mean-normalized over a sliding window of up to 3 seconds. Delta and acceleration are appended to create 60 dimension feature vectors. After that, energy based speech activity detection (SAD) is applied to select only features that correspond to speech frames.

Speaker diarization

In the case of the VAST data set, we apply speaker diarization on the test segments. For diarization we use a system based on the work in [15]. first each utterance is segmented using the same SAD mentioned in 6.2.1. For each speech segment in the utterance a feature embedding (detailed in 6.2.1), and a PLDA score is given to each pair of segments. Next, we cluster the segments using agglomerative hierarchical clustering (AHC). In our system we choose the most dominant cluster in the recording, and concatenate the segments belonging to it, to replace the original recording in the evaluation.

Speaker embedding

For extracting speaker embeddings, we used the pre-trained X-vector extractor model from [16]. Unlike in the original algorithm, we do not split

the input of the extractor in to chunks and average the X-vectors, instead we use all the feature vectors (after VAD or diarization) to extract a single X-vector. These 512-dimentional speaker embeddings are centered, whitened, and unit-length normalized.

LDA

Dimentionality reduction from 512 to 150 is performed using linear descriminant analysis (LDA).

PLDA

For scoring, a Gaussian PLDA model with a full-rank Eigenvoice subspace is used. In the case of the CMN2 data set, we train an adapted version of the PLDA scorer using in-domain unlabelled data [17].

6.2.2 Data Description

We focused on the fixed training condition, where system training is limited to specific common data sets. The x-vector extractor is trained using conversational telephone and microphone speech data extracted from the NIST 2004-2010 SRE datasets, as well as MIXER 6, Switchboard Cellular (SWBCELL) Parts I and II, and Switchboard (SWB) Phases I, II, and III corpora. In order to increase the diversity of the acoustic conditions in the training set, a 3-fold augmentation strategy is used that adds two corrupted copies of the original recordings to the training list. The recordings are corrupted by either digitally adding noise (i.e., babble, general noise, music) or convolving with simulated room impulse responses (RIR). The Gaussian PLDA is trained using the x-vectors extracted from all speech segments from the SRE and MIXER 6 sets. The PLDA adaptation is performed using the unlabelled portion of the development set.

6.2.3 Evaluation Results

We report the performance of our models on the NIST SRE 2018 development data set. All results are presented in Tables 6.2 and 6.3. The matching DET curves are also presented. As we see, utilizing the unlabelled data to train the PLDA adaptation has improved the performance significantly. Applying diarization on the VAST test set helped remove other speakers

and background audio, so the speaker embedding is more relevant and helped the overall performance of the system. In both cases (PLDA adaptation and speaker diarization), the gap between the actual DCF and the minimal DCF is significantly smaller, which adds the robustness of the system to different types of inputs.

Dataset	System	CMN2		
		EER(%)	min DCF	actual DCF
Development	X-Vector (baseline)	10.05	0.614	1.340
	X-Vector + PLDA adaptation	8.64	0.544	0.557
Evaluation	X-Vector (baseline)	11.06	0.671	0.958
	X-Vector + PLDA adaptation	10.02	0.593	0.602

Table 6.2: Results on the CMN2 portion of the development and evaluation sets of NIST SRE 2018.

Dataset	System	VAST		
		EER(%)	min DCF	actual DCF
Development	X-Vector (baseline)	7.41	0.572	0.704
	X-Vector + diarization (VAST)	5.35	0.457	0.457
Evaluation	X-Vector (baseline)	15.24	0.656	0.677
	X-Vector + diarization (VAST)	13.65	0.654	0.663

Table 6.3: Results on the VAST portion of the development and evaluation sets of NIST SRE 2018.

Figure 6.1: Performance on CMN2 development set.

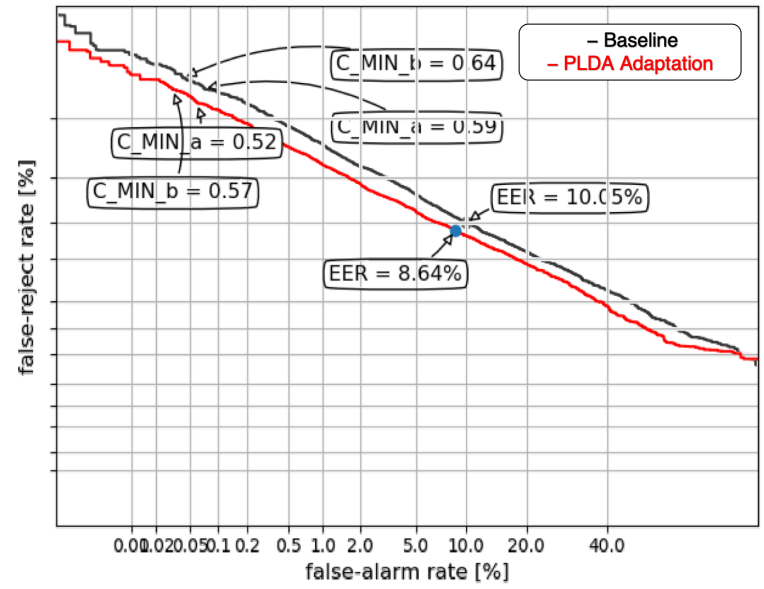


Figure 6.2: Performance on CMN2 evaluation set.

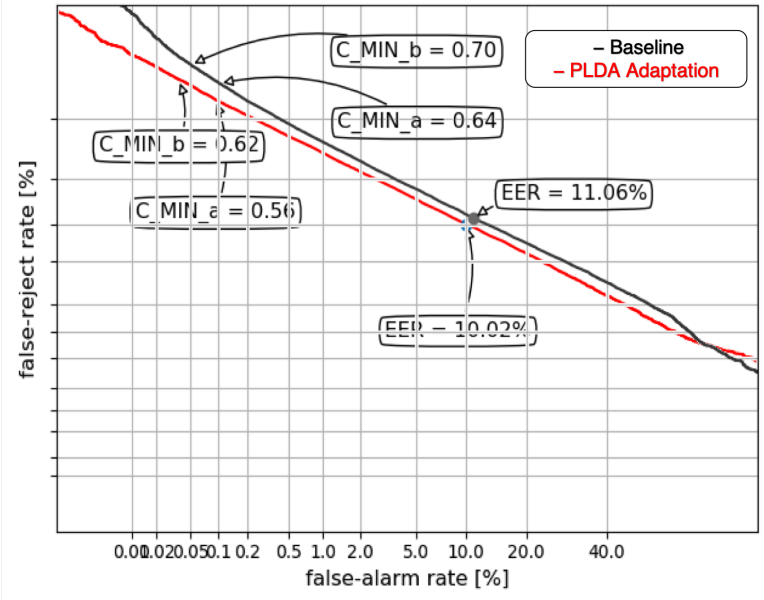


Figure 6.3: Performance on VAST development set.

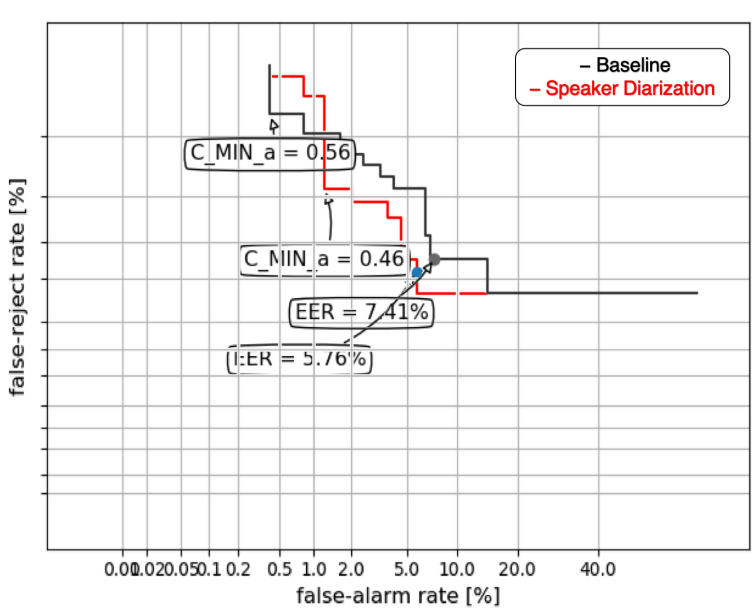
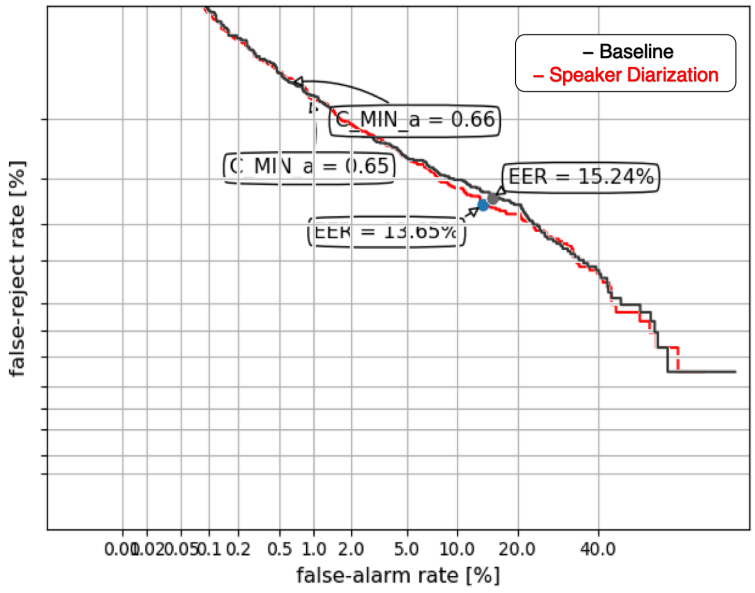


Figure 6.4: Performance on VAST evaluation dataset.



7 Conclusions

In this work, the aim was to improve speaker recognition in scenarios of insufficient in-domain data, and of multi-speaker utterances.

In the scenario of insufficient in-domain data, the algorithm's performance was improved by adapting the PLDA scoring mechanism using the available unlabelled in-domain data, and iteratively retraining the PLDA.

In the scenario of multi-speaker utterances, the algorithm's performance was improved by applying unsupervised speaker diarization using state-of-the-art X-vector speaker modeling and Agglomerative Hierarchical Clustering.

In both scenarios the performance improvement was significant, as well as the robustness of the systems due to the normalization nature of the algorithms used.

Future suggestions for further improving speaker recognition in these scenarios are end-to-end domain adaptation, especially in the speaker modeling component. This can decrease domain mismatch. For the multi-speaker scenario, a neural-network based speaker diarization model can be a promising path to research.

8 Appendix - Code Description

All the code for reproducing the experiments is located in:

https://github.com/navealg/SRE18_BGU.git

This is a repository contains the BGU team speaker recognition system and participation in the 2018 NIST speaker recognition evaluation.

8.1 Requirements

1. Install the kaldi tooklit using the instructions in http://kaldi-asr.org/doc/install.html.
2. Install Sox package ‘sudo apt-get install sox‘.

8.2 Instructions

1. Copy directories into the ‘egs‘ subfolder in the kaldi directory.
2. For each experiment, make sure that the data path in ‘run_without_training.sh‘ matches the path to the data on your machine.
3. Run ‘run_without_training.sh‘ for each experiment.
4. Use the diarization and scoring tools inside the ‘tools‘ directory.

Bibliography

[1] Douglas A Reynolds. An overview of automatic speaker recognition technology. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages IV–4072. IEEE, 2002.

[2] John HL Hansen and Taufiq Hasan. Speaker recognition by machines and humans: A tutorial review. *IEEE Signal processing magazine*, 32(6):74–99, 2015.

[3] JA Haigh and JS Mason. Robust voice activity detection using cepstral features. In *Proceedings of TENCon'93. IEEE Region 10 International Conference on Computers, Communications and Automation*, volume 3, pages 321–324. IEEE, 1993.

[4] Satoshi Imai. Cepstral analysis synthesis on the mel frequency scale. In *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 93–96. IEEE, 1983.

[5] Douglas A Reynolds. Automatic speaker recognition using gaussian mixture speaker models. In *The Lincoln Laboratory Journal*. Citeseer, 1995.

[6] Margarita Kotti, Vassiliki Moschou, and Constantine Kotropoulos. Speaker segmentation and clustering. *Signal processing*, 88(5):1091–1124, 2008.

[7] Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370, 2012.

[8] William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

[9] Douglas A Reynolds. Speaker identification and verification using gaussian mixture speaker models. *Speech communication*, 17(1-2):91–108, 1995.

[10] Douglas A Reynolds and Richard C Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, 3(1):72–83, 1995.

[11] Patrick Kenny. Joint factor analysis of speaker and session variability: Theory and algorithms. *CRIM, Montreal,(Report) CRIM-06/08-13*, 14:28–29, 2005.

[12] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

[13] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur. Deep neural network-based speaker embeddings for end-to-end speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 165–170. IEEE, 2016.

[14] Patrick Kenny. Bayesian speaker verification with heavy-tailed priors. In *Odyssey*, volume 14, 2010.

[15] Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree. Speaker diarization using deep neural network embeddings. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4930–4934. IEEE, 2017.

[16] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE, 2018.

[17] Stephen H Shum, Douglas A Reynolds, Daniel Garcia-Romero, and Alan McCree. Unsupervised clustering approaches for domain adaptation in speaker recognition systems. 2014.

תקציר

בעיית זיהוי דובר נחקרת כאן. בבעיה זו, המטרה היא, בהינתן שתי קטעי קול, להחליט האם שני הקטעים מגיעים מאותו הדובר, או מדוברים שונים. במקרים רבים, הקטעים אינם תואמים את המידע עליו אומנה המערכת. מקרה שכיח נוסף, הוא כאשר בקטע קיים קול המגיע מכמה דוברים שונים.

תזה זו פונה אל שני המקרים, של אי־התאמת תחום הדוברים ומספר רב של דוברים בקטע יחיד. השיטות המוצעות הן התאמה של אלגוריתם ה־PLDA להקטנת חוסר ההתאמה בין התחומים, ושיטת הפרדת דוברים בלתי מבוקרת, מבוססת X־VECTOR שיטות אלו נבחנו באבלואציה של זיהוי דובר שאורגנה על ידי מכון התקנים האמריקאי בשנת 2018, שם הושג שיפור משמעותי בשני המקרים, ביחס למערכת הבסיס, גם על סט הפיתוח וגם על סט האבלואציה.

אוניברסיטת בן־גוריון בנגב
הפקולטה להנדסה
המחלקה להנדסת חשמל ומחשבים

התאמת תחום והפרדת דוברים עבור זיהוי דובר

נוה אלגריסי

חיבור לשם קבלת התואר "מגיסטר" בפקולטה להנדסה

בהנחיית פרופ. חיים פרמוטר

פברואר 2019