Domain adaptation for speaker diarisation in low-resource environments

Dissertation submitted in fulfilment of the requirements for the degree *Master of Engineering* in Computer Engineering at the Potchefstroom campus of the North-West University

Lucas van Wyk

27174786


0000-0001-8254-4850

Supervisor: Prof. M.H. Davel

Co-Supervisor: Dr. C.J. van Heerden

June 6, 2022

Declaration

I, Lucas van Wyk, hereby declare that the dissertation entitled "Domain adaptation for speaker diarisation in low-resource environments" is my own original work and has not already been submitted to any other university or institution for examination.

L. van Wyk

Student number: 27174786

Signed on the 10_{th} day of December 2021 at Potchefstroom.

Acknowledgements

This research was conducted within the Multilingual Speech Technologies (MuST) group, a research group at the North-West University which specialises in the broader field of deep learning. This research was also conducted in partnership with SAIGEN, who provided the data and various systems used in this work. I would like to thank the following individuals for their contributions:

- Prof. Marelie Davel and Dr. Charl van Heerden - Supervisor and co-supervisor. Thank you for your world-class mentorship. I stand on the shoulders of giants.
- Ulrike Janke - Project manager at MuST. You kept my wheels from falling off on more than one occasion. Thank you for your patience.
- Felix McGregor - Speech engineer at SAIGEN. For all the helpful comments, feedback and Kaldi scripts. Thank you.
- Teapot - Our deceased server. Beep boop boop beep.
- Centre of High Performance Computing (CHPC), South Africa - Computational resources. 104,180 CPU hours to be exact.
- MuST students - Great minds, good laughs and fellow admirers of Douglas Adams's work.
- Prof. Barend van Wyk and Mrs. Elsa van Wyk - Parents. Thank you for your support, encouragement and genes. You are amazing human beings and outstanding role models.
- Mrs. Lizelle van Wyk - Wife and primary caregiver. You love me even when I am a lot and you support all my dreams and aspirations, no matter how wild. I love you.

Finally, I would like to thank all my friends not mentioned here. Your value is beyond measure.

Abstract

Speaker diarisation systems aim to answer the question "who spoke when?" and are useful in providing valuable metadata to downstream applications, such as automatic speech recognition systems. However, speaker diarisation systems, like most applications in the field of speech recognition, are especially challenged by domain-mismatch conditions.

In this study, we investigate methods with which to adapt a pre-trained diarisation system to a new target domain when only a small in-domain corpus is available and retraining is therefore not an option. We also develop a method for fine-tuning the adaptation process of a pre-trained speaker diarisation system using cluster analysis. Our domain adaptation process focuses on retraining and adapting the statistical components in a speaker diarisation pipeline, which are inherently domain specific, to the target domain. Lastly, we demonstrate this domain adaptation process in a real-world scenario by adapting a pre-trained diarisation system using a small in-domain dataset consisting of telephonic speech from South African call centres. We show that the adapted system can be used to provide metadata which aids the performance of automatic speech recognition systems through speaker-specific adaptations.

Keywords: *speaker diarisation, automatic speech recognition, time-delay neural networks, cluster analysis, domain adaptation, statistical speaker modelling, speaker embedding*

Contents

List of Figures

List of Tables

List of Acronyms

ASR Automatic Speech Recognition

DNN Deep Neural Network

ReLU Rectified Linear Unit

SAD Speech Activity Detection

TDNN Time Delay Neural Network

HMMs Hidden Markov Models

VBHMM Variational Bayes Hidden Markov Model

GMM Gaussian Mixture Model

LDA Linear Discriminant Analysis

PLDA Probabilistic Linear Discriminant Analysis

MLP Multi-Layer Perceptron

MFCCs Mel Frequency Cepstral Coefficients

FBanks Filter Banks

CNN Convolutional Neural Network

VAD Voice Activity Detection

DCT Discrete Cosine Transform

HMM Hidden Markov Model

SCD Speaker Change Detection

LSTM Long Short Term Memory

PCA Principal Component Analysis

RG Radial Gaussinization

SSD Spherical Symmetric Density

AHC Agglomerative Hierarchical Clustering

OOD out-of-domain

ID in-domain

DER Diarisation Error Rate

JER Jaccard Error Rate

SGD Stochastic Gradient Descent

SC Silhouette Coefficient

Chapter 1

Introduction

In this chapter we discuss the utility and relevance of this speaker diarisation study. We delineate the scope and objectives, which provides the context for the rest of this work.

1.1 Background

The goal of speaker diarisation is to identify different speakers in an audio stream and to segment the audio stream into speaker-homogeneous segments, where each segment-level utterance belongs to a single speaker. Put another way: speaker diarisation is the answer to the question "who spoke when?".

Speaker diarisation has two main applications: Automatic Speech Recognition (ASR) systems that make use of speaker-homogeneous segments for adapting acoustic models [1] and speaker indexing, which provides metadata for downstream audio processing applications [2].

Historically, most speaker diarisation and recognition systems were based on i-vectors [3] which are obtained by projecting high-dimensional supervectors, or speaker representations, onto a lower-dimensional subspace using factor analysis. However, since the advent of deep learning, i-vectors have since been replaced by techniques based on Deep Neural Networks (DNNs) which can be trained as end-to-end systems, omitting i-vectors entirely [4], [5]. In early versions of such systems, a Multi-Layer Perceptron (MLP) is trained for frame-level speaker recognition [6]–[8] by extracting a lower-dimensional speaker embedding from frame-level speech features and performing classification. These frame-level speaker embeddings are averaged to form utterance-level speaker embeddings, known as d-vectors and can be used for speaker diarisation and recognition.

More recently, an alternative to d-vectors was proposed by Snyder et al. [5] where a Time Delay Neural Network (TDNN) is used to extract fixed-dimensional speaker embeddings, known as x-vectors, from variable length input features. It has since been shown that x-vectors outperform i-vectors and d-vectors, especially when trained on large amounts of data [5].

In speaker diarisation, speaker embeddings such as i-vectors, d-vectors and x-vectors are used to identify segments of speech belonging to the same or different speakers. These same-or-different speaker decisions can be made based on a pre-defined similarity measurement such as euclidean distance or cosine similarity taken between embeddings. In practice, however, Probabilistic Linear Discriminant Analysis (PLDA) [9] models are used to compare speaker embeddings and make inferences on identity.

The success of DNNs and PLDA models is largely dependant on one factor – the availability of in-domain (ID) training data. X-vector systems and PLDA models generally do not perform well when a significant domain mismatch exists between the training and test datasets [5], [10]. However, the assumption that the training data stretches across enough domains for the test data to be sufficiently represented is not a realistic nor practical one.

In this work, we focus on the problem of domain mismatch and investigate ways to adapt existing speaker diarisation systems, trained on vast amounts of out-of-domain (OOD)

data, to a low-resource environment (ID data), specifically South African call centres. The ID data used in this work is a small collection of calls from 9 different South African call centres. As this small proprietary corpus is not large enough to sufficiently train a speaker diarisation system from scratch, the main research objective is to investigate techniques and methods with which a pre-trained speaker diarisation system can be adapted to new, low-resource domains. It is important to note the distinction between ID and OOD data in this work: ID data refers to the data we are adapting to, i.e., our target domain. OOD data refers to the data on which the pre-trained system had been trained on and is therefore outside of the target domain.

The end goal of this dissertation is to develop a speaker diarisation system which has been sufficiently adapted to function on South African telephonic speech such that the speaker-level indexing provided by the system can be used to isolate a specific speaker in a conversation before the audio is transcribed. Additionally, the speaker indexes can be used by downstream ASR components for speaker-specific adaptations.

1.2 Problem statement

The use of DNNs and PLDA models for extracting discriminatory information from speech features has been widely successful, given that the target data belongs to a similar domain as the training data. In instances were a large domain mismatch exists between the target and training data, retraining the entire system on the target data is not always an option, especially not in low-resource environments where the available target data is insufficient for training purposes. Our goal is to adapt a pre-trained speaker diarisation system to a low-resource target domain. Specifically, we aim to adapt a pre-trained system for use in South African call centres where audio is characterised by low-quality recordings and excessive background noise.

1.3 Project scope

Given the problem statement, we restrict our study to a limited number of data sets and speaker embedding architectures:

- **Data sets**: To investigate domain adaptation techniques for low-resource environments, we limit our study to two datasets: the EMRAI and SAIGEN corpora. The SAIGEN corpus, as described in Section 3.3, is a small, proprietary dataset containing a collection of hand-labelled calls from different South African call centres. In contrast, the EMRAI corpus (Section 3.2) is a publicly available collection of synthetic two- and three-person dialogues created from the Librispeech corpus. The EMRAI corpus is included to perform a preliminary evaluation on state-of-the-art, pre-trained systems, so that a suitable baseline can be chosen for use in the SAIGEN corpus.
- **Speaker embedding models**: As our goal is to investigate domain adaptation, we will not retrain any speaker embedding models but rather use publicly available pre-trained models. Furthermore, we limit our design choices to x-vectors as they are the current de facto for speaker embedding models.

1.4 Research questions

With our problem statement and project scope in mind, we formulate the following research questions:

- How can an existing speaker diarisation system be adapted to a new domain, specifically South African call centres, without retraining the entire system?
- Does the SAIGEN corpus contain sufficient data for the domain adaptation of a pre-trained system?

- Can an adapted speaker diarisation system be used to improve the performance of an ID downstream ASR system? If yes, then to what extent?

1.5 Objectives

To address the research questions, we define the following objectives:

- Analyse the SAIGEN corpus to understand its content and recording environment and verify the quality and correctness of the hand-labelled data.
- Determine the current state-of-the-art for speaker diarisation:
  - Identify and compare the top pre-trained, state-of-the-art Speech Activity Detection (SAD) model architectures.
  - Identify and compare the top pre-trained, state-of-the-art x-vector architectures.
  - Identify and compare methods of domain adaptation.
- Evaluate the chosen state-of-the-art systems on the EMRAI corpus and identify the most suitable baseline system(s).
- Adapt the baseline system to the SAIGEN corpus and investigate the extent to which the baseline performance can be improved.

1.6 Research methodology

In this work, we approach our research objectives in the following way:

- **Literature Review:** Conduct a review of the current state of speaker diarisation and obtain an understanding of the various subcomponents in such systems with

an emphasis on DNN-based techniques and statistical speaker modelling. Review related work on domain adaptation in low-resource environments.

- **Data harvesting and pre-processing:** Review the method with which the SAIGEN and EMRAI corpora were gathered, annotated and pre-processed. Obtain an understanding of the environment in which the data was captured.
- **Experimental Procedure:**
  - Identify pre-trained and publicly available state-of-the-art diarisation systems.
  - Evaluate and compare state-of-the-art systems and adaptations on the synthetic EMRAI corpus to determine suitable approaches. Apply these approaches to the SAIGEN corpus to evaluate the efficacy of domain adaptation in a real-world scenario.
  - Evaluate the utility of the adapted speaker diarisation system for improving ASR and speaker indexing.
- **Development Environment and Toolkits:** For our experimentation, we will mainly focus on two widely-used toolkits (see Section 2.8 for more information):
  - Kaldi is used for all x-vector speaker embedding and PLDA models.
  - For all scoring and evaluation metrics we use Pyannote-audio, a Python-based toolkit for evaluating speaker diarisation systems.
- **Analysis:** Analyse the effects, benefits and pitfalls of domain adaptation in real-world scenarios as well as the effect of the adapted systems on downstream applications, such as speaker indexing and ASR.

1.7 Dissertation overview

The rest of this dissertation is structured as follows:

- Chapter 2 provides an introduction to speaker diarisation and explains necessary concepts and systems on which the rest of the work builds.
- Chapter 3 explains our datasets, data harvesting protocols and pre-processing.
- Chapter 4 describes the state-of-the-art systems that we investigate in this study and the distinctions among them. We also include an evaluation of each pre-trained system on the EMRAI corpus and identify the most suitable baseline for our real-world experiments on the SAIGEN corpus. Furthermore, we evaluate and establish our procedures for domain adaptation using the EMRAI corpus.
- Chapter 5 evaluates our chosen pre-trained, state-of-the-art systems and domain adaptation techniques on our target domain – the SAIGEN corpus. We also introduce the use of cluster analysis to fine-tune the adaptation process and assess the use of our adapted system(s) in ASR.
- Chapter 6 concludes this study with the key innovations and findings, and suggestions for future work.

1.8 Publications

- An article titled "Robust Diarisation of Telephone Conversations Using Target Speaker Verification" at the SACAIR 2020 Unconference[1].
- "Unsupervised Fine-Tuning of Speaker Diarisation Systems Using Silhouette Coefficients" [11], published at the South African Conference for Artificial Intelligence Research (SACAIR) 2021. The paper contain sections from Chapters 3, 4 and 5.

[1]Available at: `https://rb.gy/vfvb1e`

Chapter 2

Literature review

This chapter discusses the inner workings of speaker diarisation systems. We show how different components in a generic speaker diarisation pipeline operate and are trained, and how these components – feature extraction, segmentation, speaker modelling and clustering – function as a single entity. Lastly, we discuss and compare possible approaches to domain adaptation.

2.1 Introduction

In general, modern speaker diarisation systems are either end-to-end systems, where the entire system is comprised of a single component [12], [13], or multi-module systems wherein the system is comprised of different components acting in unison. In the case of an end-to-end system, the entire system is trained in a single step. However, since the focus of this dissertation is on domain adaptation, we rather focus on multi-module systems where the entire system does not have to be retrained for domain adaptation. Figure 2.1 illustrates the traditional, multi-module approach to speaker diarisation. This chapter describes each component in detail, especially the speaker modelling and clustering

components.

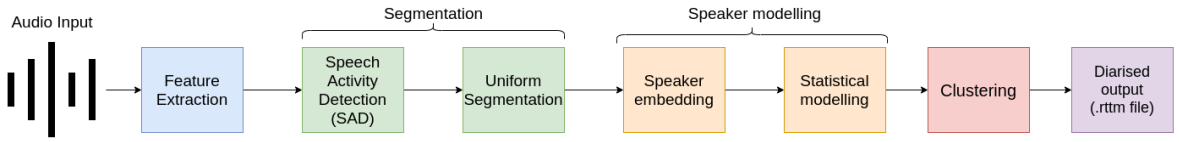

Figure 2.1: A traditional speaker diarisation system

The first step in diarisation is feature extraction (Section 2.2), which accepts raw audio as input and extracts hand-crafted features such as Mel Frequency Cepstral Coefficients (MFCCs) or Filter Banks (FBanks) [14] from the audio. Some feature extraction modules contain an optional speech enhancement step, which takes the hand-crafted features as input and outputs a new, denoised set of features [15]–[17]. However, we omit the speech enhancement step, as it is outside the scope of this research.

After feature extraction, the input audio is segmented. The purpose of the segmentation step is to remove non-speech frames from the input features and to pass uniform chunks of speech frames, known as speech segments, to the speaker modelling step (Section 2.4). Speaker modelling is a two-step process wherein fixed-dimensional representations, or speaker embeddings, are assigned to input segments. Thereafter, statistical models are used to extract discriminatory information from the speaker embeddings such that speech segments from the same speakers can be more easily identified in unseen data.

Speaker modelling provides the input to the clustering (Section 2.6) step which uses the speaker embeddings to form groups of speech segments containing the same speakers. These segment-level clusters are then combined to form the diarised output in the '.rttm' file format [18].

Lastly, we discuss approaches to domain adaptation (Section 2.5) which are primarily focused on the adaptation of the statistical components. This chapter is concluded with an overview of the evaluation metrics with which diarisation performance is measured(Section 2.7) and the frameworks and toolkits used in this research (Section 2.8).

2.2 Feature extraction

Feature extraction plays an important role in speaker diarisation and acts as a pre-processing step wherein raw audio is transformed into a feature space that is more representative of its content. Although recent speaker diarisation systems have been proposed that can use raw audio as input, such as SincNet [19], extracted features are still most commonly used. Therefore, in this study, we focus on the two most widely used speech features in speaker diarisation: MFCCs and FBanks.

To compute FBanks raw audio is first passed through a pre-emphasis stage which amplifies high frequencies such that low and high frequencies have equal magnitudes. The audio is then divided into fixed-length frames and a Hamming window is applied to each frame; the most common framing choice is a frame length of 25ms with a 10ms step between frames. After framing, a power spectrum is computed for the input audio and a mel-scale filter bank is applied to the power spectrum to obtain the FBanks.

To compute MFCCs, the FBanks are first computed and thereafter a Discrete Cosine Transform (DCT) is applied to obtain the MFCCs. The point of applying a DCT to FBanks is to decorrelate the FBank coefficients, which are highly correlated. Figure 2.2 illustrates the process of computing FBanks and MFCCs.

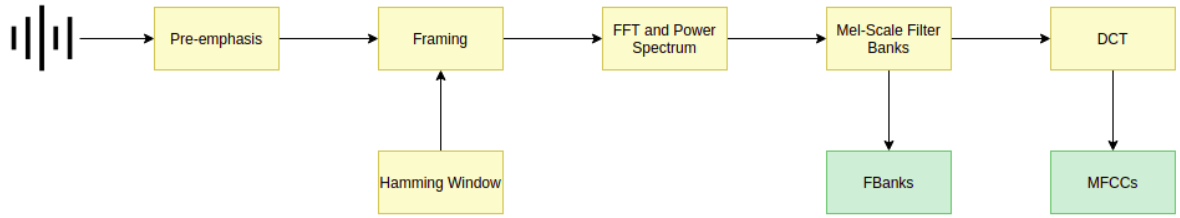

Figure 2.2: An illustration of the process for computing MFCCs and FBanks. Notice that MFCCs are just FBanks which had been passed through a DCT.

2.3 Segmentation

Ideally, speaker embeddings should only be extracted on audio containing speech, and each speaker embedding should preferably contain a single speaker. The goal of segmentation, therefore, is to detect and remove non-speech from audio (using SAD) and to subdivide long speech segments into shorter segments which are more likely to contain a single speaker (uniform segmentation). In this section we describe how SAD and uniform segmentation works and how they are used in this research.

2.3.1 Speech activity detection

The main challenge for SAD models, also referred to as Voice Activity Detection (VAD) models, is distinguishing human speech from non-speech events such as background noise or speaker noise like laughter or clapping. To do this, SAD models are composed of a feature extraction step that extracts speech features such as MFCCs and passes them on to the classification step, which performs frame-level speech/non-speech classification.

SAD models differ in complexity ranging from energy-based systems, where the classification between speech/non-speech is based on the amount of energy in the signal, to more complex methods that use statistical models such as a Gaussian Mixture Model (GMM)s [20] and Hidden Markov Model (HMM)s [21] for classification. More recently, however, DNNs had also been successfully applied to the problem of SAD [22].

As previously mentioned, the performance of the SAD model has a pronounced effect on the overall speaker diarisation performance because it can create false positive or missed speech segments. Therefore, in order to evaluate the performance of our speaker diarisation system, *we will report our results in two ways*: 'oracle SAD', which uses the SAD marks from the ground truth annotations and 'system SAD' which uses SAD marks obtained via a SAD model.

Lastly, since the focus of our study is on adapting speaker diarisation systems for low-resource environments, the SAD models in this research are restricted to publicly available,

pre-trained models as described in more detail in Section 4.2.1.

2.3.2 Uniform segmentation

For earlier speaker diarisation systems, the modus operandi, in terms of segmentation, was to use Speaker Change Detection (SCD) models. An SCD model tries to find the points in speech segments where the active speaker changes and segments the audio accordingly. Chen et al. [23] proposed a method wherein speech features are modelled as a Guassian process such that the probability of a speaker change between each frame-level input can be inferred. The speaker change points are then placed between all frames that have a speaker change probability higher than a predefined threshold. Similarly, Kemp et al. [24] performed SCD by placing speaker change points between frame-level features where the two adjacent frames have a Kullback-Leibler distance which is larger than a predefined threshold.

Modern speaker diarisation system use another form of segmentation known as uniform segmentation, where speech segments are split into shorter, fixed-length sequences. These uniform segments are typically 1.5 seconds in length but many different sizes are used in practice. Uniform segmentation subdivides speech segments in a sliding window fashion with a step sizes typically ranging between 250ms and 750ms. Since the advent of DNN-based speaker embeddings, SCD was mainly replaced by uniform segmentation since the varying-length segments provided by SCD introduces additional variability in the speaker embeddings that could deteriorate the discriminatory power of speaker representations [25].

2.4 Speaker modelling

The primary purpose of speaker modelling is to represent speech features belonging to a specific speaker such that the speaker in question can be recognised or identified amongst other representations. In this section we describe the two main aspects of speaker mod-

elling: speaker embedding and statistical modelling. We first describe how DNNs are used for speaker embedding and why x-vector systems are well suited for this task. We then describe how the speaker embeddings, obtained from an x-vector system, are statistically modelled using Radial Gaussinization (RG), Linear Discriminant Analysis (LDA) and PLDA models.

2.4.1 Speaker embedding

In its essence, speaker embedding can be viewed as a pattern recognition task: information from speech features have to be extracted to the point where a recognisable pattern emerges with which the speaker of an utterance can be inferred – a problem that DNNs are well suited for. In speaker diarisation, a DNN is used to extract a fixed dimensional pattern, referred to as a speaker embedding, from each audio segment given as input such that speaker embeddings from the same speaker are more similar compared to speakers embeddings from different speakers.

In Section 1.3 we restrict ourselves to only using x-vector systems for speaker embedding. However, many different DNNs have been applied to speaker diarisation, the most notable of which is the MLP and TDNN. This subsection first describes how an MLP and TDNN work and why the TDNN-based x-vector systems are preferred over the d-vector alternatives.

Multilayer perceptron

DNNs had been successfully applied to a wide array of problems and tasks and have dramatically increased in popularity with the accessibility of more and faster computational power. The simplest form of a DNN is the MLP.

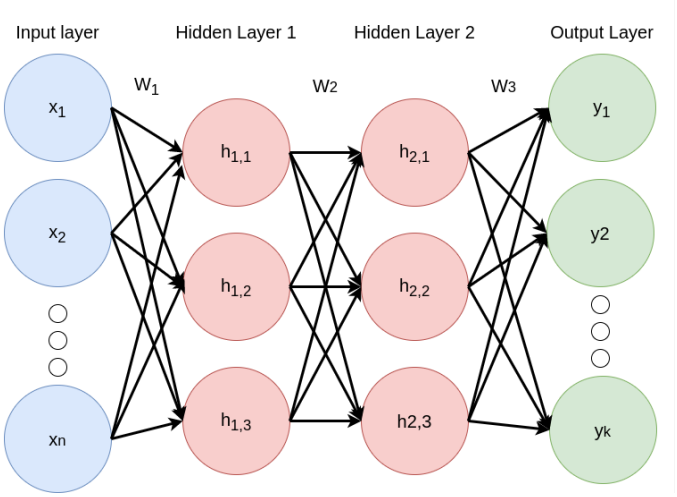

Figure 2.3: An MLP with n inputs, two hidden layers and an output layer with k outputs.

An MLP, as illustrated in Figure 2.3, is nothing more than a linear combination of non-linear functions known as activation functions. In an MLP, the input to every node (except the input nodes) is the weighted sum of the output of every other node in the preceding layer. For the network in Figure 2.3 the output can be described as:

$$\begin{aligned} y &= \sigma(W_3^T \mathbf{h}_2) \\ h_2 &= \sigma(W_2^T \mathbf{h}_1) \\ h_1 &= \sigma(W_1^T \mathbf{x}) \end{aligned} \tag{2.1}$$

Where W_i is the weight matrix of layer i and each row of W_i is a weight vector $\mathbf{w}_{ij}$ which contains the weights between node j in layer i and each node in the preceding layer $i-1$, $\mathbf{h}_1$ and $\mathbf{h}_2$ are the output vectors of the first and second hidden layers and $\mathbf{x}$ is the vector of inputs. The weights are trainable parameters and the optimal weights will map the inputs $\{x_1, x_2, ..., x_n\}$ to the desired outputs $\{y_1, y_2, .., y_n\}$. The activation function σ is a pre-defined non-linear function. A popular activation function for DNNs is the Rectified Linear Unit (ReLU) [26] given by $max(0; z)$ where z is the output of the node before activation. Although other activation functions exist we primarily focus on ReLUs as they are used in all our speaker embedding networks as discussed in Section 4.2.2 and are generally faster to train with compared to other activation functions.

To find the optimal weights for a network, the loss function $E(\mathbf{w})$ is minimised, with $\mathbf{w}$ being the collection of all weight vectors. A network's loss function measures the difference between a network's current output, $\mathbf{y_{predicted}}$ and the desired output, $\mathbf{y_{true}}$ for each sequence of inputs $\mathbf{x}$ and is minimised using an iterative process known as stochastic gradient descent [27]. The main task of stochastic gradient descent is to find the weight vectors $\mathbf{w}$ that minimises the chosen loss function $E(\mathbf{w})$. To do this a small step is taken from $\mathbf{w}$ to $\mathbf{w} + \delta$ and the change in $E(\mathbf{w})$, given as $\delta E \simeq \delta \mathbf{w} \nabla E(\mathbf{w})$ is calculated. $\nabla E(\mathbf{w})$ indicates the direction of greatest increase for $E(\mathbf{w})$ and since the error function is a function of $\mathbf{w}$, the optimal $\mathbf{w}$ is obtained when $E(\mathbf{w}) = 0$. The weight vectors $\mathbf{w}$ are repeatedly moved in the direction of $-\nabla E(\mathbf{w})$ to decrease the error until $E(\mathbf{w}) = 0$ or $E(\mathbf{w})$ reaches some minimum. During stochastic gradient descent, propagation is used to calculate the gradient of the loss function $\nabla E(\mathbf{w})$ with respect to each parameter in the weight vector $\mathbf{w}$ [28].

Time delay neural networks

A TDNN is a type of DNN architecture used for mapping some temporal input sequence $\{x_t, x_{t+1}, x_{t+2}...x_{t+n}\}$ to a corresponding sequence of outputs $\{y_t, y_{t+1}, y_{t+2}...y_{t+n}\}$ [29]. TDNNs use a feed-forward architecture to handle the context-dependant information of speech signals through its hierarchical structure [30].

In a TDNN, the first layer processes input from narrow contexts of speech and the deeper layers processes information by taking the spliced activations from the previous layers to learn patterns from a wider temporal context.

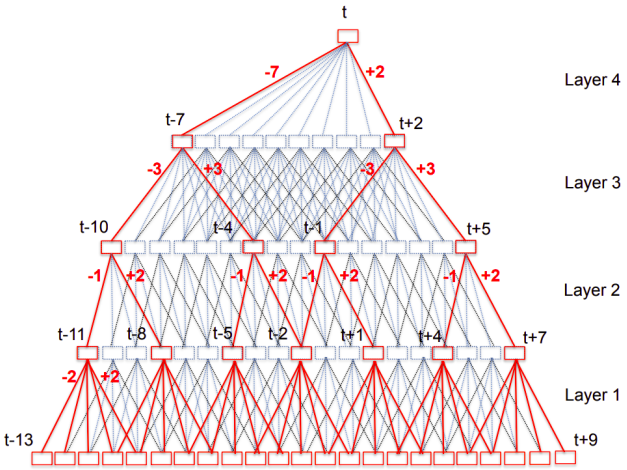

Figure 2.4: Computation in TDNN with sub-sampling (red) and without sub-sampling (blue). [31]

.

Figure 2.4 illustrates how a TDNN processes long temporal inputs, which are either processed using sub-sampling or not. In the context of speech processing, the temporal inputs to a TDNN are MFCCs or FBanks. Sub-sampling (shown in red) in TDNNs was proposed by Peddinti et al. [31] and involves only passing every other frame to deeper layers. Sub-sampling had been shown to make TDNNs more computationally efficient as splicing a continuous window of frames over the entire temporal context (shown in blue) leads to overlap, and redundancy [30]. The weight vectors of each layer in a TDNN are tied across time-steps which makes the TDNN similar to Convolutional Neural Network (CNN) architectures in this regard. Since the weight vectors are tied, TDNN layers are trained using the gradients accumulated over all time steps [31].

D-vectors

A d-vector system, as first proposed by Variani et al. [7], is a fully connected feed-forward network that accepts stacked FBanks as input and outputs a fixed-dimensional output vector. D-vector systems are trained for speaker classification and the speaker embedding or d-vector is obtained by simply taking the output of the last hidden layer. A simple

d-vector system is illustrated in Figure 2.5.

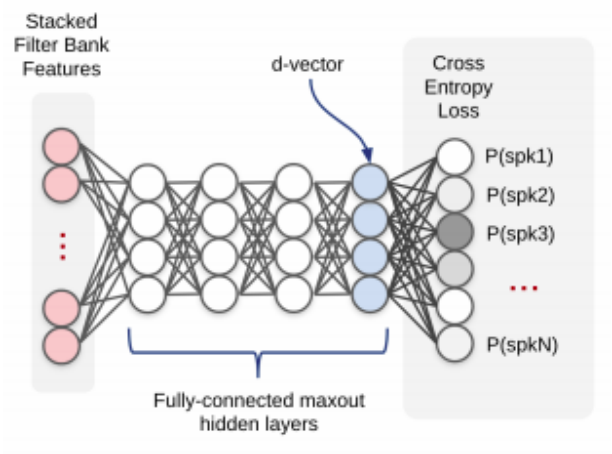

Figure 2.5: An illustration of a d-vector speaker embedding model. Illustration by Park et al. [25].

Notice that d-vector systems operate on fixed-length input utterances and can therefore only operate on utterances with a fixed length. More recently, Long Short Term Memory (LSTM) based d-vector systems have shown promising results in speaker diarisation [32], [33]. The utility of LSTM models in the context of speech recognition makes sense as they are better able to capture the sequential nature of speech features compared to stationary fully-connected networks [33].

X-vectors

X-vectors [5] are similar to d-vectors in the sense that both are DNN-based methods for speaker embedding. X-vectors, however, are distinct from d-vectors in two ways: x-vector systems are based on TDNNs and secondly, x-vector systems can extract speaker embeddings on variable-length input utterances while d-vector systems require fixed-length input utterances. X-vector systems are currently the de facto method for speaker embedding and considered state-of-the-art as they are used as benchmarks in major speaker recognition challenges such as the DIHARD [22], Chime6 [34] and VoxCeleb challenges [35]. We therefore only use x-vector systems in this study, as they are currently the most

widely used and can function on variable-length input sequences, which d-vector systems cannot do.

A basic x-vector system is illustrated in Figure 2.6. The frame-level layers represent a standard TDNN (as shown in Figure 2.4) which operates on frame-level features, such as MFCCs or FBanks, followed by a statistics pooling layer which computes the mean and standard deviation of each dimension of the frame-level representations and stacks them to form a fixed dimensional output vector. The utility of the statistics pooling layer is to provide a fixed dimensional output for any number of frame-level inputs. The output of the statistics pooling layer is then used as input to the segment-level layers, which are standard, fully-connected feed-forward layers. The output of the last segment-level layer is passed through a softmax layer [36], and all non-linearities are ReLUs. The x-vector or speaker embedding is extracted from a fully-connected layer between the statistical pooling and softmax layers and typically has a fixed dimension of 256 or 512. The dimensionality of an x-vector can be anything in practice but is defined during training and kept constant.

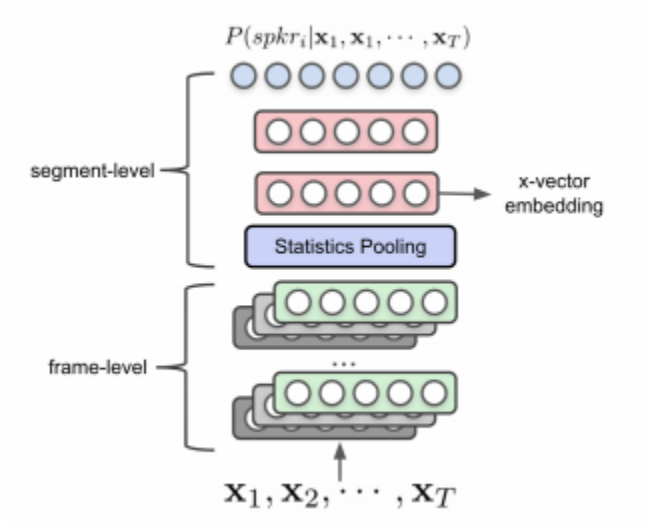

Figure 2.6: An illustration of an x-vector speaker embedding model. Illustration by Park et al. [25].

X-vector systems are trained for speaker classification and the ultimate goal is to train the network to generalise well to speakers that have not been seen in the training data

[5]. The extent to which an x-vector system can generalise also serves as our motivation to use pre-trained x-vector systems trained on large corpora, as systems trained on larger datasets tend to generalise better across domains.

2.4.2 Statistical modeling

Statistical modelling has long played an important role in speaker diarisation and ASR in general. In speaker diarisation, statistical models are used to probabilistically determine the active speaker in an audio segment, given the x-vector extracted on that segment.

Figure 2.7 shows how statistical modelling is applied to x-vectors [5]. In this subsection each component of the statistical modelling process is described. First, a RG transformation is used to transform the x-vectors into a space where the x-vectors can be assumed to follow a Gaussian distribution. Secondly, LDA is applied to the Gaussianised x-vectors as a method of dimensionality reduction. Finally, a PLDA model is trained using the output of the LDA model with which the speaker of an utterance can be inferred.

Figure 2.7: The process and order of the components of statistical modelling, in the context of speaker diarisation.

Radial Gaussianisation

When training a probabilistic model on multi-dimensional data it is often assumed that the data follows a Gaussian distribution and that the dimensions of the data are statistically independant. By making this assumption, multi-dimensional data can be modelled as a factor of numerous simpler distributions, such as Gaussians. In speaker diarisation, LDA and PLDA models assume that speaker embeddings (x-vectors) follow a Gaussian distribution and that speaker and channel components are statistically independent. However,

it had been empirically shown that speaker and channel behaviours are non-Gaussian [37].

Since the Gaussian assumption does not hold for speaker embeddings, it is common practice to project speaker embeddings onto a space where their distributions are more representative of a Gaussian. Dehak et al. [38] showed that the performance of speaker recognition systems can be significantly increased when the speaker embeddings are projected onto a subspace where the Gaussian assumption holds.

In order to transform speaker embeddings into a Gaussian subspace, Romero et al. [39] proposed a type of non-linear transformation called RG [40] which transforms inputs into a Gaussian subspace by extracting independent components. Firstly, speaker embeddings are transformed into a Spherical Symmetric Density (SSD) via a linear whitening transform learned from data samples [40]. Secondly, speaker embeddings are normalised (using l2-norm) to constrain the embeddings to a d-dimensional hypersphere. The RG-transformation is commonly used in x-vector speaker embedding networks [5], [41], and colloquially referred to as whitening and centering.

A centering and whitening or RG transformation is usually trained on the same data as the speaker embedding models [39]. The training steps are as follows:

1. Centering and whitening:
    - Compute mean and sample covariance ($\mathbf{m}$, θ) of the x-vectors extracted on the development data.
    - Obtain a whitening transformation $A = D^{-\frac{1}{2}}U^T$ with $\theta = UDU^T$.
    - Center (i.e. mean normalise) and whiten (i.e. normalise to have identity covariance matrix) x-vectors: $\mathbf{x}_{wht} = A(\mathbf{x} - \hat{\mathbf{m}})$
2. Scaling:
    - Project whitened x-vectors onto unit hypersphere (i.e. length normalisation) $\mathbf{x}_{wht} = \frac{\mathbf{x}_{wht}}{||\mathbf{x}_{wht}||}$

By applying an RG transformation to the x-vectors the covariance matrix of the whitened x-vectors is diagonalised ($\theta = I$) and the dimensions of the x-vectors are statistically independent. Additionally, length normalisation constrains the x-vectors to a unit hypersphere, which is where most of the mass of a standard, high-dimensional Gaussian is distributed [42]. Following the RG transformation, the x-vectors now represent a Gaussian distribution and an LDA model can be trained to reduce the dimensionality of x-vectors and to project them onto a subspace where similar x-vectors are grouped together.

Linear discriminant analysis

LDA is a statistical method that attempts to find a linear transformation such that the input data is transformed to a space that maximises the separation between data points of different classes. LDA achieves this by defining a new spatial axis wherein the variance between classes is maximised, and the variance within classes is minimised [42].

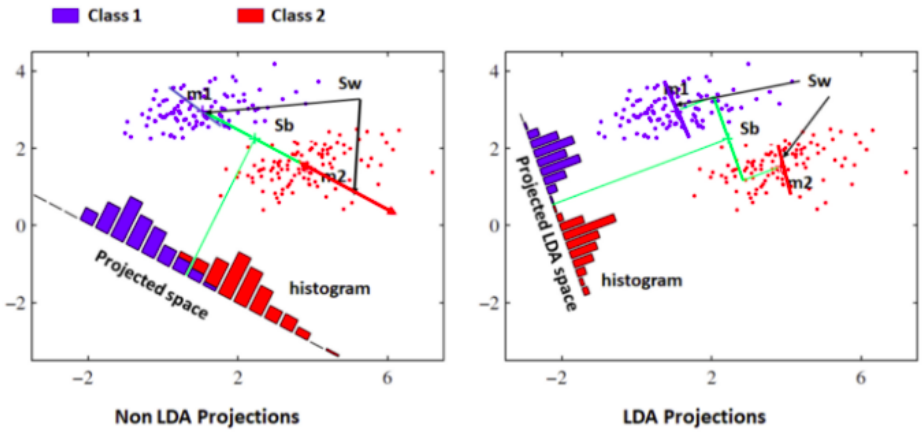

Figure 2.8: Data sampled from two classes (blue and red) before and after LDA projection. Notice how LDA linearly transforms the data to a space where the within-class separation (S_w) is minimised and the between-class separation (S_b) is maximised. Illustration by Bishop et al. [42].

In speaker diarisation, LDA is commonly used as a dimensionality reduction technique where higher-dimensional speaker embeddings are transformed into a lower-dimensional

subspace with the desired class separation. The reason for applying dimensionality reduction is to reduce the computational cost of the downstream PLDA model while still retaining the desired data distribution.

LDA is closely related to Principal Component Analysis (PCA), which is another common dimensionality reduction technique [42]. PCA, however, does not take into account any difference in class while LDA seeks to maximise the ratio of between-class and within-class variance – known as the 'Fisher ratio' [43]. The linear transformation which maximises the aforementioned ratio, as shown by Bishop et al. [42], is found as follows:

1. Consider a set of n samples defined as $\{\mathbf{x}_1 ... \mathbf{x}_n\}$ where each column $\mathbf{x}_i$ is a vector of length d. Assume that $\mathbf{x}_i$ belongs to one of K classes and let C_k be the subset of samples in class k and let n_k be the number of samples in each subset C_k.

2. Compute the 'within-class' (S_w) and the 'between-class' (S_b) scatter matrices as follows:
$$S_w = \frac{\Sigma_k \Sigma_{i \in C_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T}{N} \tag{2.2}$$
and
$$S_b = \frac{\Sigma_k n_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T}{N} \tag{2.3}$$
where $\mathbf{m}_k = \frac{1}{n_k} \Sigma_{i \in C_k} x_i$ is mean of the kth class and $\mathbf{m} = \frac{1}{N} \Sigma_k \mathbf{x}_i$ is the mean of the entire sample set.

3. Find the $dXd^{'}$ matrix W, with $d^{'}$ the desired number of dimensions such that the linear transformation given by $\mathbf{x} = W^T \mathbf{x}$ projects the sample $\mathbf{x}$ onto a $d^{'}$ dimensional latent space such that the Fisher ratio is maximised:
$$\underset{W}{\text{argmax}} \frac{W^T S_B W}{W^T S_W W} \tag{2.4}$$
The above maximisation problem can be solved by solving the generalised eigenvalue problem:
$$S_b \mathbf{w} = \lambda S_w \mathbf{w} \tag{2.5}$$
Where $\mathbf{w}$, the columns of W, are the generalised eigenvectors corresponding to the $d^{'}$ largest eigenvalues. It can also be shown that W simultaneously diagonalises the

scatter matrices $W^T S_b W$ and $W^T S_w W$ which implies that projected samples in the latent space will be decorrelated both between and within classes.

LDA can also be performed using GMMs, allowing us to probabilistically determine the class membership of a data point [44]. This is done by defining a latent variable $\mathbf{y}$ that represents the mean of a specific class and the center of the mixture component. The conditional probability of generating a sample $\mathbf{x}$ given the class variable $\mathbf{y}$ is then given by:

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\mathbf{y}, \theta_W) \tag{2.6}$$

With θ_W being the within-class covariance matrix and $\mathcal{N}$ being a normal distribution representing the class samples $\mathbf{x}$, centred around $\mathbf{y}$. Equation 2.6 implies that the samples of a class can be generated once the parameters of the class-specific Gaussian are known. Similarly, a sample $\mathbf{x}$ can be classified by comparing the probability of observing $\mathbf{x}$ given each class variable $\mathbf{y}$.

Additionally, a prior is imposed on each class variable which assigns a probability mass to each of the finite values of $\mathbf{y}$:

$$P(\mathbf{y}) = \sum_{k=1}^{K} \pi_k \delta(\mathbf{y} - \mu_k) \tag{2.7}$$

Where, K is the number of classes, π_k is the mixing coefficient of each Gaussian, μ_k is the mean of class k and δ is the Dirac delta function. Notice that the probability of observing the discrete variable $\mathbf{y}$ is equal to the mixture component of its associated Gaussian (denoted as π_k). The standard LDA projections are recovered by maximising the likelihood of the observed data with respect to the parameters π_k, μ_k and θ_W [9].

Since the prior probability $P(\mathbf{y})$ is discrete, LDA cannot be used for classification on unseen classes. Therefore, LDA is only used for dimensionality reduction and PLDA is used for making inferences on unseen classes.

Probabilistic linear discriminant analysis

The greatest utility of PLDA, for use in speaker diarisation, is that it can be used to infer the probability of two samples originating from the same underlying class [5]. Different variations of PLDA exist which differ in the way the data is modelled such as the heavy-tailed PLDA proposed by Kenny et al. [37] which models the data as a Student's t distribution. In this work, however, we use a Gaussian PLDA which models the data as a Gaussian and for the remainder of this dissertation we use Ioeffe's implementation of a Gaussian PLDA [9].

PLDA, unlike LDA can apply to unseen data by modelling the prior probability $P(\mathbf{y})$ as a continuous variable given by:

$$P(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{m}, \theta_B) \tag{2.8}$$

Where $\mathbf{m}$ is the mean of the entire dataset and θ_B is the between-class covariance matrix. As in LDA, the variable $\mathbf{y}$ still represents the class-specific mean. Given the continuous latent variable $\mathbf{y}$ a datapoint $\mathbf{x}$ is generated in the same manner as LDA, as shown in Equation 2.6. For visual representation of PLDA, we refer the reader to Figure 2.9.

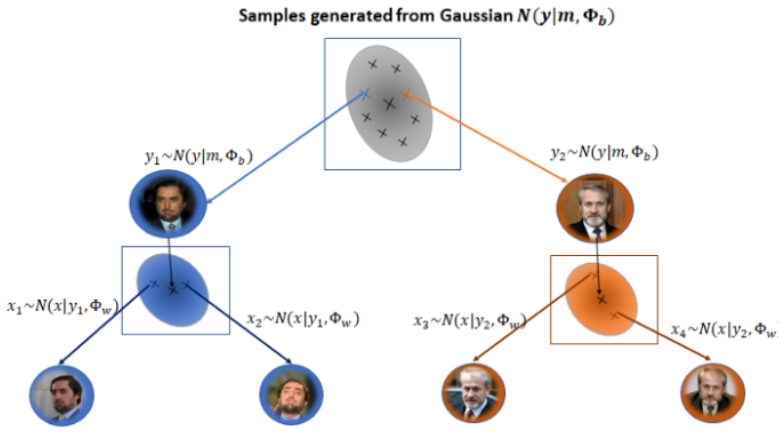

Figure 2.9: An example of samples generated by a PLDA model: A class (in this case an individual's face) is represented by the continuous latent variable $\mathbf{y}$ sampled from the distribution in Equation 2.8 and the data samples $\mathbf{x}$ are instances of class $\mathbf{y}$,i.e., the individual's face at different angles and lightning. Illustration by Singh, Prachi [45].

As illustrated in Figure 2.9, the goal of PLDA is to project data samples to a latent space such that the same distributions model samples from the same classes. To achieve this, a transformation matrix V is defined such that the within-class and between-class covariance matrices, θ_W and θ_B, are diagonalised by[1] $V^T\theta_B V = \Psi$ and $V^T\theta_W V = I$ where Ψ is a diagonal matrix and I is the identity matrix. By taking $A = V^{-T}$ we have $\theta_W = AA^T$ and $\theta_B = A\Psi A^T$ and the covariance matrices, now diagonalised, decorrelate data samples both between and within classes. Two Gaussian random variables $\mathbf{u}$ and $\mathbf{v}$ are then defined that represent the data samples $\mathbf{x}$ and class variables $\mathbf{y}$ in the latent space. The relationship between the latent and feature space of a PLDA model is depicted in Figure 2.10. The PLDA model is now given by[2]:

$$\begin{aligned} \mathbf{x} &= \mathbf{m} + A\mathbf{u} \\ \mathbf{y} &= \mathbf{m} + A\mathbf{v} \\ \mathbf{u} &\sim \mathcal{N}(\cdot|\mathbf{v}, \mathbf{I}) \\ \mathbf{v} &\sim \mathcal{N}(\cdot|0, \Psi) \end{aligned} \tag{2.9}$$

[1] A detailed derivation of the linear transform V is available at: `https://tinyurl.com/4w9krmuh`
[2] The derivation of Equation 2.9 is available at: `https://tinyurl.com/nn39pze7`

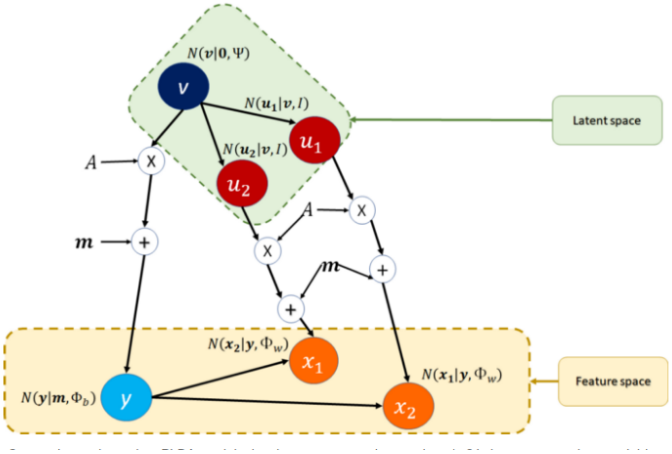

Figure 2.10: The relationship between latent space and feature space variables in a PLDA model. Illustration by Singh, Prachi [45].

The mean $\mathbf{m}$, covariance matrix Ψ and the loading matrix A are learned using an expectation maximisation algorithm (EM). For training, labelled data is required to calculate the mean and covariance matrices. The details of the PLDA training process are rather involved and not relevant with regard to the way we utilise PLDA in Section 4.3.3. For an in-depth explanation we refer the reader to Ioeffe's method [9].

In speaker diarisation, the first use of a PLDA model is to determine if two unseen samples originate from the same underlying class:

1. Consider a gallery of examples $\mathbf{x}_g \in \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ which each belong to one of M classes and a probe example $\mathbf{x}_p$. We would like to determine which class $\mathbf{x}_p$ is a member of.

2. Start by projecting all the data onto the latent space with $\mathbf{u} = A^{-1}(\mathbf{x} - \mathbf{m})$ which will decorrelate the data as the covariance of $\mathbf{u}$ is $\mathbf{I}$.

3. The class, K of our probe example $\mathbf{x}_p$ can be found by maximising the likelihood $K = \underset{g}{\operatorname{argmax}}\, P(\mathbf{u}_p|\mathbf{u}_g)$

Secondly, the class variable $\mathbf{y}$ from a single example $\mathbf{x}$ can be inferred by computing the posterior probability $P(\mathbf{y}|\mathbf{x})$ which will itself be a Gaussian. The estimate of $\mathbf{y}$ can now

be estimated by maximising the posterior probability with respect to the class variable which is simply the mean of Gaussian $P(\mathbf{y}|\mathbf{x})$. Hence, by using equation 2.9, the class variable $\mathbf{y}$ is given by:

$$\mathbf{y} = \mathbf{m} + A\mathbf{v} = \mathbf{m} + A(\Psi + \mathbf{I})^{-1}\Psi A^{-1}(\mathbf{x} - \mathbf{m}) \tag{2.10}$$

Lastly, a PLDA model can be used to decide if two examples originate from the same class by computing a 'similarity score' between two datapoints:

1. Define two hypotheses h_o and h_1 with h_o being the same class hypothesis and h_1 being the different class hypothesis.

2. Compute the likelihood ratio R based on h_0 and h_1:

$$\begin{aligned} R(\mathbf{u}_1, \mathbf{u}_2) &= \frac{likelihood(h_0)}{likelihood(h_1)} = \frac{P(\mathbf{u}_1, \mathbf{u}_2)}{P(\mathbf{u}_1)P(\mathbf{u}_2)} \text{ with} \\ P(\mathbf{u}_1, \mathbf{u}_2) &= \int P(\mathbf{u}_1|\mathbf{v})P(\mathbf{u}_2|\mathbf{v})P(\mathbf{v})d\mathbf{v} \text{ and} \\ \text{Decision} &= \begin{cases} \text{same class if } log(R(\mathbf{u}_1, \mathbf{u}_2)) > \theta \\ \text{different class otherwise} \end{cases} \end{aligned} \tag{2.11}$$

In Equation 2.11 $log(R)$ is defined as the similarity score, or PLDA score between two data points and θ is some arbitrary decision threshold.

In speaker diarisation, PLDA scores between embeddings are used to construct a PLDA score matrix, which is a square matrix with each column representing the PLDA scores between a specific speaker embedding and every other speaker embedding. The PLDA score matrix is used for determining the similarity between speaker embeddings and clustering them accordingly [5].

2.5 Domain adaptation

As mentioned in Section 1.3, the goal of this research is to adapt component(s) of a pre-trained speaker diarisation system for a new domain. Since the goal of an x-vector system is to extract embeddings that are separable between speakers, it is nonsensical to retrain an x-vector system on a small ID dataset since a system that was pre-trained on a much larger OOD corpus is trained on many speakers across many domains leading to better generalisation as the large number of parameters in an x-vector system cannot be sufficiently trained on the small ID dataset.

However, as mentioned in Section 2.4.2, speaker diarisation systems contain domain-specific components, such as the RG and LDA transforms and PLDA models that are trained to model the speaker distributions it was trained on. For this reason, we focus on the adaptation of the RG, LDA and PLDA models for new domains. In this section we describe how each of the above mentioned components can be adapted or retrained for a new domain.

2.5.1 Adaptation of the RG and LDA transforms

The combination of the RG and LDA transforms, as shown in Section 2.4.2, is a proven technique for projecting the output embeddings of an x-vector system to a Gaussian subspace where classes are separated. When applying pre-trained diarisation systems to new domains, it is standard practice to retrain the RG and LDA transforms on ID data [5], [46].

To train the RG and LDA transforms, we require labelled ID data containing utterances and their associated speaker identities to estimate the within-class and between-class covariance matrices. In cases where labelled ID data is unavailable, it is also possible to use a pre-trained speaker diarisation system to label ID utterances for training [46]. However, the efficacy of this unsupervised transform is reliant on the size of the domain mismatch between the ID and OOD dataset: a large mismatch would lead to poor performance of

the pre-trained system, which would in turn provide inaccurate labels for training the transform.

2.5.2 PLDA interpolation

As the goal of a PLDA model is to represent the speaker embeddings in a given domain, it makes logical sense to retrain PLDA models when working on a new domain. However, the performance of a PLDA model hinges on the availability of ID training data since more training data would result in a PLDA model that can better represent the target domain. Thus, the assumption that the available training data would contain enough training examples such that the target domain can be sufficiently modelled is perhaps unrealistic [10].

To create a PLDA model that can sufficiently represent a low-resource target domain Romero et al. [47] proposed a simple linear interpolation wherein the parameters of an OOD PLDA model, trained on large amounts of data, is interpolated with the parameters of an ID PLDA model. This interpolation process will result in an adapted PLDA model that will be more suited to the target domain. This method of PLDA adaptation has proven very useful in practice and was used in the winning submission of the second DIHARD diarisation challenge [48].

The adaptation method works as follows:

1. Extract speaker embeddings from the labelled ID corpus using the retrained LDA and RG transforms.
2. Train a new PLDA model on the ID embeddings; use the ID labels when assigning speaker labels to embeddings.
3. Derive an adapted PLDA model by interpolating the parameters of the OOD and newly trained ID PLDA models (the OOD PLDA model is not retrained but used as-is):

$$\begin{aligned} \theta_{W-adapt} &= \alpha\theta_{W-in} + (1-\alpha)\theta_{W-out} \quad \text{and} \\ \theta_{B-adapt} &= \alpha\theta_{B-in} + (1-\alpha)\theta_{B-out} \quad \text{with} \\ \alpha &\in [0,1] \end{aligned} \tag{2.12}$$

Where $\Theta_{W-adapt}$ and $\Theta_{B-adapt}$ are the adapted within-class and between-class covariance matrices, Θ_{W-in} and Θ_{W-out} are the ID and OOD within-class covariance matrices and Θ_{B-in} and Θ_{B-out} are the ID and OOD between-class covariance matrices. The interpolation parameter $\alpha \in [0,1]$, is proportional to the contribution that the ID data has on the adaptation and should be fine-tuned on the ID data. The optimal α parameter is usually estimated using a small held-out set [47].

Just as the RG and LDA transforms can be trained in a supervised or unsupervised way, an ID PLDA can also be trained in an unsupervised way: by first using a clustering algorithm to cluster speaker embeddings according to speaker identity and then using the resulting speaker labels to train the PLDA model. When training a PLDA model in an unsupervised way, Agglomerative Hierarchical Clustering (AHC) is usually used to cluster speaker embeddings using the PLDA-scores obtained from the OOD PLDA model as a similarity measurement. The AHC merging process is then stopped when the similarity measurement between clusters falls below a certain threshold.

2.6 Clustering

After speaker modelling, clustering algorithms are used to sort speaker embeddings according to speaker identities: each speaker embedding is assigned to a cluster such that speaker embeddings originating from the same speaker are grouped together. In speaker diarisation, the speaker embeddings are most commonly clustered after applying an RG and LDA transformation (as defined in Section 2.4.2) as these subspaces are more likely to exhibit the desired class separation. The most common clustering algorithms currently used in speaker diarisation are AHC [49], K-Means clustering [50] and Variational Bayes Hidden Markov Model (VBHMM) Clustering [51].

It is worth noting that other clustering methods have been used for speaker diarisation, such as spectral clustering, to give only one example. In this research, however, we will limit clustering methods to AHC, K-Means, and VBHMM as these feature the most in literature and have proven to consistently deliver the best results in the context of speaker diarisation [5], [25], [34], [46], [48]. In this section we show how AHC, K-Means and VBHMM clustering is applied to speaker diarisation.

2.6.1 K-Means clustering

K-Means clustering clusters data points by finding k centroids such that examples from the k_{th} class are closer to the k_{th} centroid than it is to any other centroid. A significant drawback of the K-Means Algorithm is that it requires the number of centroids, or equivalently, clusters to be known.

The K-Means Algorithm has many variants, but the concept remains the same and works as follows:

1. Initialise k-centroids: a centroid is a point in the feature space with the same dimensionality as the input examples. Centroids can be initialised randomly or by some initialisation protocol.

2. For each datapoint, compute the distance between itself and each centroid and assign each data point to the nearest centroid.

3. Move each centroid such that it is located in the centre of all data points assigned to it.

4. Repeat steps 2 and 3 until the centroids remain unchanged since the previous iteration; if the centroids do not change, the algorithm has converged.

Note that the K-Means algorithm as described above only describes the underlying algorithm, which remains unchanged for all variants. More complex methods differ in the

metrics used to measure the distance between centroids, the centroid initialisation scheme, and if outliers are considered when updating cluster assignments.

2.6.2 Agglomerative hierarchical clustering

AHC is an iterative process in which clusters are continually merged until a specified stopping criterion is met. AHC works by first assigning each data point to its own cluster. The pair of clusters with the highest similarity score, calculated using a pre-defined similarity metric, is then repeatedly merged until some stopping criterion is met. In speaker diarisation, the most commonly used similarity metric is the PLDA score [34], [48], [52]. The merging process of AHC can be stopped by either defining the number of desired clusters or by stopping the merge process when the similarity score between the two closest clusters are lower than a certain threshold.

2.6.3 Variational Bayes Hidden Markov Model clustering

Hidden Markov Models (HMMs) are very prominent in ASR and have been widely used for recognition of continuous speech since the 1970s [53]. More recently, however, HMMs has been applied to the problem of clustering in the context of speaker diarisation. Diez et al. [54] showed that HMMs can be used to infer speaker states and change probabilities and subsequently assign speaker labels to x-vectors – a process known as VBHMM clustering.

The underlying HMM is depicted in Figure 2.11 and the VBHMM clustering method works as follows:

1. Start by assuming that each x-vector is generated by an HMM with K states with each state corresponding to one of K possible speakers.
2. Let $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_T}\}$ be the observed sequence of x-vectors.
3. Let $Z = \{z_1, z_2, ..., z_T\}$ be a sequence of discrete latent variables that define the

alignment of x-vectors to HMM states, e.g., if $z_t = s$ then the x-vector $\mathbf{x_t}$ is generated by the HMM state s with $s \in K$.

4. At the same time, the speaker distributions, e.g., the distribution of $\mathbf{x_t}$ is modelled by a latent vector $\mathbf{y_s}$ which is of the same dimensionality as $\mathbf{x_t}$. A standard PLDA model, as shown in Section 2.4.2, is used to infer $\mathbf{y_s}$ and to model the within and between speaker variability[3].

5. The latent variables $\mathbf{y_s}$ and z_t are jointly estimated given the input sequence of x-vectors $\mathbf{X}$.

6. The solution to the speaker diarisation task is then given by the sequence Z, which encodes the alignment of x-vectors to speaker states s. Therefore, the diarisation problem consists of finding the assignment of x-vectors to speakers. To do this, the posterior distribution $P(Z|\mathbf{X}) = \int P(Z, \mathbf{Y}|\mathbf{X})dY$ is approximated using Variational Bayes (VB) inference.

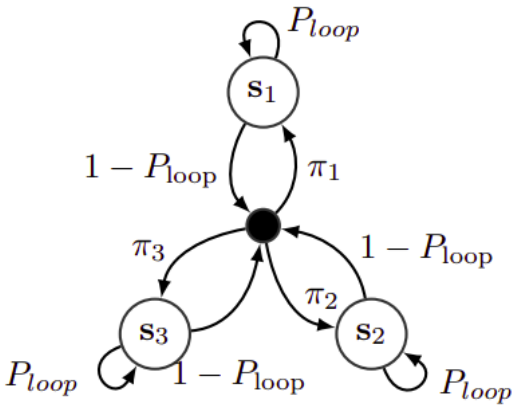

Figure 2.11: HMM model for 3 speakers with 1 state per speaker and a dummy non-emitting initial state.

Figure 2.11 shows the HMM topology used in VBHMM clustering. The probability of transitioning to the same speaker is denoted as P_{loop}, and the remaining probability $1 -$

[3]The derivation of $\mathbf{y_s}$ from a PLDA model is outside of the scope of this research. For more information we refer the reader to the original paper by Diez et al. [54]

P_{loop} is the probability of a speaker change. At initialisation, the HMM immediately transitions from the non-emitting node to one of the speaker states with probability π_s. The probability of leaving a speaker and entering another state s is $(1 - P_{loop})\pi_s$.

In the case of an unknown amount of speakers, the number of speakers or speaker states is estimated using AHC clustering. The stopping threshold for the AHC step is chosen such that the HMM is initialised with a larger number of states than actual speakers so that it may converge to the correct number of speakers. Alternatively, the minimum number of speaker states to which the HMM can converge may also be set.

Lastly, the VBHMM has three fine-tunable parameters:

1. P_{loop}: The probability of *not* changing speaker states between observations. Refer to Figure 2.11.

2. F_A: The acoustic scaling factor F_A is used to compensate for the incorrect assumption of statistical independence between observations, which results in overconfident posterior distributions of latent variables $\mathbf{y}_s$ and $\mathbf{z}_t$. F_A counteracts the independence assumption by scaling down the log-likelihood of observations.

3. F_B: The speaker regularisation constant F_B will make the model more or less aggressive when dropping redundant speakers. A high value of F_B will result in more speaker models being dropped during inference.

2.7 Evaluation metrics

Since modular speaker diarisation pipelines, such as the one we focus on in this research, consist of many components working in unison the poor performance of one component may negatively affect the entire system's performance. It is therefore vital to know how each component contributes to the system error. This section discusses the scoring and evaluation metrics for SAD and speaker diarisation systems that will be used to investigate the performance of various components in the speaker diarisation context.

2.7.1 Scoring metrics for speech activity detection

SAD is a crucial step in any speaker diarisation system as misclassified speech segments can harm the performance of downstream components such as the x-vector model and PLDA backend. By incorrectly classifying speech segments, the downstream components will not function on all speech segments or it will function on audio segments that are actually non-speech such as silence or noise. The performance of SAD models can be measured several ways:

- F1-score, which gives a holistic view of the accuracy of speech/non-speech labels.
- Coverage, which shows how much speech is included or 'covered' in the SAD labels.
- Boundary precision, which measures the precision or 'correctness' of speech/non-speech boundary placements.
- Purity, which measures the percentage of speech regions that only contain speech.
- Boundary recall, which compares the amount of hypothesised speech/non-speech boundaries to the actual speech/non-speech boundaries.

F1-score

F1-score, also known as the balanced F-score or F-measure, can be interpreted as a weighted average of the precision and recall. The F1-score is commonly used to evaluate binary classifiers such as SAD models, which only distinguish between two classes, speech and non-speech. The F1-score is defined as follows:

$$F1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2.13}$$

Where TP are the true positives or correctly classified frames, FP are the false positives or frames incorrectly labelled as speech. FN are the false negatives or frames incorrectly labelled as non-speech.

Boundary precision and recall

The boundary precision and recall metrics measure the 'correctness' of the placement of speech boundaries, within a given tolerance. In terms of speaker boundaries, recall is the ratio of reference speech boundaries that have a corresponding hypothesised speech boundary, i.e., a recall of 100% indicates that the reference and hypothesis labels have the same amount of speech boundaries. Similarly, precision is the ratio of hypothesised speech boundaries that have a corresponding reference speech boundary *within a certain tolerance*, i.e., a precision of 100% indicates that all of the hypothesised speech boundaries have a corresponding reference speech boundary.

Figure 2.12 visually depicts the boundary precision and recall metrics. The top 4 boundaries are the reference (oracle SAD) boundaries, and the bottom 3 are the hypothesis (system SAD) boundaries. In this example the recall is 75% because 3 out of 4 reference boundaries were correctly detected, and precision is 100% because all hypothesised boundaries are within an acceptable range of a reference boundary.

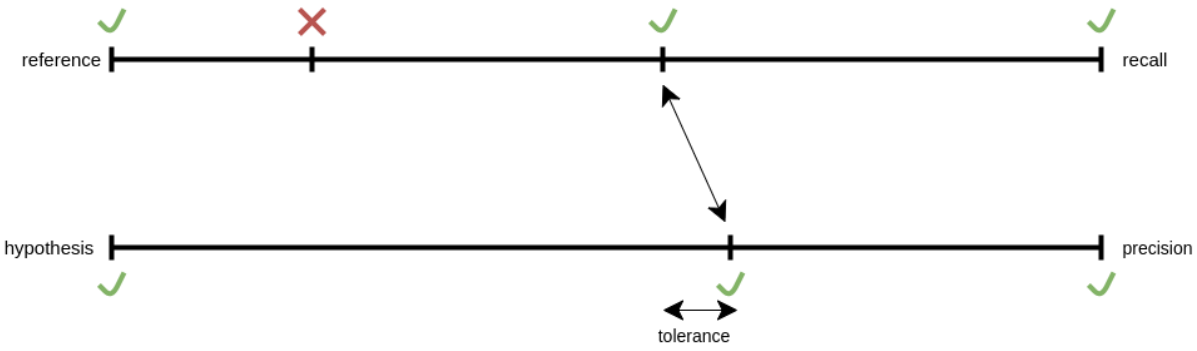

Figure 2.12: Boundary recall and precision

Segment coverage and purity

Coverage and purity are the dual metrics that describe how similar the labels of the hypothesised speech segments are to the reference speech segments. Coverage measures the ratio of reference segments that are correctly labelled or 'covered' in the hypothesised

segments and purity measures the ratio of hypothesis segments that contain a single reference event, e.g., a hypothesised segment that only contains speech is considered 'pure' but a hypothesised segment that mostly contained speech but also includes a small amount of non-speech is considered 'impure'.

Figure 2.13 visually depicts how the coverage and purity metrics are calculated. The top 3 segments are reference (oracle SAD) segments and the bottom three are hypothesised (system SAD) segments. In this example the first reference segment has a coverage of 100% because it is completely covered in the first hypothesis segment. Conversely, the second and third segments have a lower coverage because the corresponding hypothesised segments do not entirely cover them. Similarly, the third hypothesis segment has a purity of 100% because it only contains speech, and the first hypothesised segment has a lower purity of 50% since only 50% of the segment contains speech although the entire segment was labelled as such.

The overall coverage and purity is just the average segment-level coverage and purity, therefore, per the example in Figure 2.13 the overall coverage is 78.3% and the overall purity is 75%.

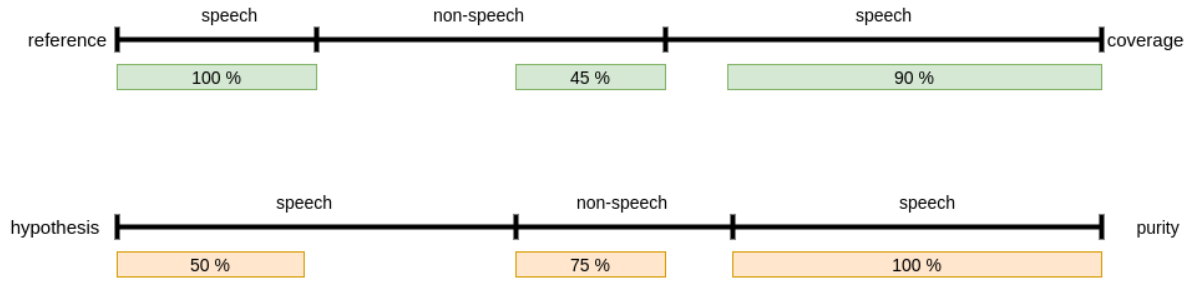

Figure 2.13: Segment coverage and purity

2.7.2 Scoring metrics for speaker diarisation

The modular nature of speaker diarisation systems necessitates the need for informative scoring metrics, such that the origin of the diarisation error can be determined. For-

tunately, standardised evaluation metrics for diarisation, as proposed by the the Rich Transcription Evaluation Series (NIST) [55]–[57], has been widely adopted.

The two most widely used diarisation scoring metrics, and the ones used in this research, are Diarisation Error Rate (DER) and Jaccard Error Rate (JER). Notice that DER does not take different speakers into account whereas JER is the weighted average of per-speaker errors. Therefore, DER is commonly used as an indication of overall diarisation performance while JER reflects the per-speaker diarisation performance. In scenarios where it is important that each speaker is correctly diarised one may opt for JER and in scenarios where the overall diarisation performance is given preference one may opt for DER.

Diarisation error rate

DER is most commonly used for evaluating diarisation performance and is the primary metric used in the international DIHARD speaker diarisation challenges [52], [58], [59]. DER is defined as:

$$DER = \frac{\text{False Alarm + Missed Detection + Speaker Confusion}}{\text{Total Duration of Time}} \tag{2.14}$$

In Equation 2.14 'False Alarm' refers to the total time labelled as speech which is non-speech, 'Missed Detection' refers to the total amount of time labelled as non-speech, which is speech and lastly, 'Speaker Confusion' refers to the total amount of time which has the wrong speaker label.

During scoring, a Hungarian Algorithm [57] is used to establish a one-to-one mapping between the hypothesis and reference outputs since the reference speech labels may be real names such as 'Carl' or 'Bob', but the hypothesis labels are just the identifiers used for the clusters such as 1 and 2 or 'a' and 'b'.

The DER metric also allows for the use of a collar which is a boundary set around each segment wherein the output will not be scored. A collar is used to account for inconsistent

annotation errors in the reference transcripts [56]. The collar is usually set to 250ms. Therefore, we use a standard collar of 250ms for all experiments when computing DER.

Jaccard error rate

The JER is similar to DER, but instead of measuring all speaker errors over the entire duration, it takes the weighted average of per-speaker errors. JER is defined as:

$$JER = \frac{1}{N_{ref}} \sum_{i}^{N_{ref}} \frac{\text{False Alarm}_i + \text{Missed Detection}_i}{\text{Duration}_i} \tag{2.15}$$

In Equation 2.15 False Alarm_i is the total amount of time incorrectly assigned to speaker i. $\text{Missed Detection}_i$ is the total amount of time belonging to speaker i which had not been assigned to speaker i. Duration_i is the total duration of time for which speaker i was speaking. Lastly, N_{ref} is the number of speakers in the reference annotation.

As with DER, the same Hungarian Algorithm is used to establish a one-to-one mapping between hypothesis and reference outputs. Additionally, a collar is used with JER in the same way as in DER and is typically set to 250ms. The collar will therefore be set to 250ms for all experiments when computing JER.

2.8 Toolkits

In this work the feature extraction, x-vector systems, statistical models and clustering backends are all employed using specialised and popular open-source toolkits. The toolkits used for speaker diarisation and scoring are described in this section.

2.8.1 PyAnnote-metrics

Pyannote-metrics is an open-source Python library used for evaluating speaker diarisation performance [60]. We will be using PyAnnote-metrics to calculate DER, JER, boundary recall and precision and segment coverage and purity.

2.8.2 Scikit-learn

Scikit-learn is an open-source Python library used for machine learning tasks [61] such as classification, regression and clustering. In this work we will be using Scikit-learn to compute the F1-score for SAD evaluation and for performing AHC and K-Means clustering.

2.8.3 Kaldi

Kaldi is an open-source C++ toolkit for speaker recognition [62]. Kaldi is amongst the most popular toolkits in the field of speech technology. In this work we use Kaldi for all feature extraction, e.g. the extraction of MFCCs and FBanks from raw audio. Furthermore, we will use Kaldi for the training of all RG and LDA transformations and PLDA models. Lastly, the original x-vector system [5] was created in Kaldi, and all our x-vector systems are derived from the original Kaldi x-vector training script known as a 'recipe'.

2.9 Conclusion

This chapter discusses the fundamental concepts of a speaker diarisation system: feature extraction, segmentation, speaker modelling, and clustering. We discussed the importance of domain adaptation in speaker diarisation systems and approaches to the problem of domain mismatch, which are focused on the adaptation and retraining of domain-specific components. Lastly, the various scoring metrics used to evaluate speaker diarisation systems are discussed. The following chapter introduces the data used in this research

and our methods of data harvesting, pre-processing and verification.

Chapter 3

Data and preprocessing

This work primarily uses two datasets: the EMRAI corpus which is a synthetic, open-source dataset and the SAIGEN corpus which consists of hand-labelled calls from several South African call centres. This chapter describes the two corpora in terms of their domain, composition and preprocessing and why these two datasets are used.

3.1 Introduction

This chapter gives an overview of the data used in this research, their formats, preprocessing, and augmentation. Two different corpora are used in this research: the EMRAI corpus, a synthetic, open-source corpus which is used for the dual purpose of evaluating domain adaptation techniques and identifying the a suitable baseline system (Chapter 4). The second corpus, referred to as the SAIGEN corpus, is a proprietary dataset used to evaluate our chosen baseline system for domain adaptation in a real-world scenario (Chapter 5).

The reason for the use of two corpora is that the SAIGEN corpus was incomplete at

the start of this study; the EMRAI corpus had therefore used as a proxy. Due to time constraints, the experiments on the EMRAI corpus could not all be replicated on the SAIGEN corpus. Therefore, when the SAIGEN corpus became available, the baseline systems were chosen based on the their performance on the EMRAI corpus.

3.2 EMRAI corpus

The EMRAI speech corpus [63] is a publicly available[1] corpus of synthesised dialogues. The dialogues were created using speech from the Librispeech speaker recognition corpus [64], which consists of over 100 hours read English speech. As with Librispeech, the EMRAI corpus is partitioned into a 'clean' and 'other' set, with the former consisting of higher quality audio recordings and accents closer to US English and the latter consisting of poorer quality recordings and foreign accents. Furthermore, the EMRAI corpus contains two- and three-person dialogues both with and without speaker overlap. However, as the purpose of this corpus is only to identify suitable pre-trained systems for the real-world experiments, only the two-person dialogues from the 'other' set are used without overlap. Overlapped speech is revisited in Section 3.3.2.

The EMRAI corpus was created using 16kHz audio. To mimic the conditions of narrow-band telephonic speech, we downsample the 16kHz audio to 8kHz and pass it through a telephone codec before upsampling it back to 16kHz. Lastly, to evaluate the performance of the baseline systems with different levels of background noise, we add randomly sampled noise from the 'free sound' subset of the MUSAN noise corpus [65]. Noise is added to the 16kHz audio *before* downsampling with uniform intensity ranging from -5dB to 5dB, measured in Signal-to-Noise-Ratio (SNR) such that for a single level of background noise, e.g., 5dB the subset will have an average SNR of 5dB.

Table 3.1 provides an overview of subsets within the EMRAI corpus. All subsets in Table 3.1 contain only 2-person dialogues without overlap. This research only uses the 'dev-other' and 'test-other' subsets.

[1] available at `github.com/EMRAI/emrai-synthetic-diarization-corpus`

Table 3.1: Synthetic 2-person corpus with no overlap

	Dialogs	Utterances (Turns)	Tokens	Hours
Train	292	28522	989715	98.15
Dev-clean	48	2673	53765	4.98
Dev-other	45	2822	50227	4.69
Test-clean	43	2605	52279	5.07
Test-other	45	2861	51305	4.85

3.3 SAIGEN corpus

The SAIGEN corpus is a proprietary corpus consisting of calls from various South African call centres and reflects a real-world scenario for speaker diarisation. After a baseline system had been established, the SAIGEN corpus is used to investigate the feasibility of domain adaptation for practical use cases.

3.3.1 Data collection

The corpus consists of 20 hours of hand-labelled telephonic conversations from 9 different South African call centres in the English, Zulu and isiSotho languages. The calls consist of 8kHz narrowband telephone speech, typically encoded using a GSM telephone codec and stored in a WAV49 format [66]. Furthermore, the audio contains large amounts of background and channel noise which, together with the low bandwidth and encodings, pose a challenge for speaker diarisation and ASR systems. Table 3.2 shows the number of calls and duration of calls for each call centre.

Table 3.2: Number and duration of calls per call centre in the SAIGEN corpus.

	Calls	Hours
Call centre 1	19	1.95
Call centre 2	9	1.96
Call centre 3	6	1.21
Call centre 4	5	1.43
Call centre 5	17	4.66
Call centre 6	4	2.41
Call centre 7	32	3.82
Call centre 8	17	1.25
Call centre 9	9	1.44
	118	20.13

Annotation protocol

The calls were hand labelled by a single annotator using the PRAAT[2] computer software package. Initially, an annotation protocol was developed and iteratively revised for the annotator to follow. As the annotations progressed guidelines were added to the protocol on how to label different types of salient events such as background speech or overlapped speech[3].

The annotations consist of labels and timestamps to show who spoke when, and other salient events such as silence, background speech, music and noise are also indicated. Background speech is classified as coherent speech from a speaker that is not partaking in the conversation. Annotations are initially saved in the .TextGrid file format, which is native to PRAAT, but converted to a .rttm file format which is the standard format for diarisation systems.

[2]Publicly available at: `https://praat.en.softonic.com/download`
[3]Annotation protocol available at: `https://rb.gy/y0glyf`

Verification

To verify the annotations, we randomly chose two calls from each call centre to verify that the speaker boundaries are correctly placed and to check for labelling errors. Furthermore, we applied our baseline system, as described in Section 5.3, on the entire corpus. Using the diarisation output from the baseline system, we identified calls with higher error rates than other calls in the same call centre and manually verified the annotations of the identified calls. In most cases, the calls with the highest error rates contained labelling errors or spelling errors within the annotations. To rectify this, all annotations were replaced with standardised speaker tags. Furthermore, no significant boundary placement errors were detected and the DER and JER error metrics include a collar during scoring to account for annotations errors, as described in Section 2.7.2.

3.3.2 Data analysis

The following is an analysis that investigates the recording conditions, composition and availability of the call centre data. The speakers in the SAIGEN corpus are split into two main groups: the 'agent', which refers to an employee or operator and the 'caller' or customer. While the agent and caller play distinct roles in each conversation, and most conversations contain only two speakers, some conversations may contain more than two speakers as the caller may speak to multiple agents during a single call or visa-versa. Additionally, it is possible that the same speaker or agent may appear in more than one call, but this has not been investigated in this study as the speakers in each call are not uniquely identified, i.e., only the speaker tag showing 'caller' or 'agent' is given. This will also be the case in practice.

We describe the composition of calls in the corpus, the amount and types of speech/non-speech, the recording environment, and the difference between agent and caller speakers in terms of utterance duration and frequency. Recall from Section 2.7 that the error metrics, DER and JER, calculate the amount of labelling error in terms of false alarm, missed detection and speaker confusion. Therefore, it is necessary to understand the

composition of the SAIGEN corpus to identify the most probable challenges.

First, we analysed the average length of speaker utterances. Figure 3.1 shows the respective probability density distributions of the utterance lengths for the caller and agent speakers. The dotted line represents the average utterance length for the caller (orange) and agent (blue) speakers. Notice that the agent has a higher number of long utterances (utterances longer than 2 seconds) and a lower number of short utterances (utterances shorter than 2 seconds) than the caller. The fact that the agent has longer utterances on average could mean that higher-quality speaker embeddings could potentially be extracted from agent utterances than caller utterances, since the utterances are longer.

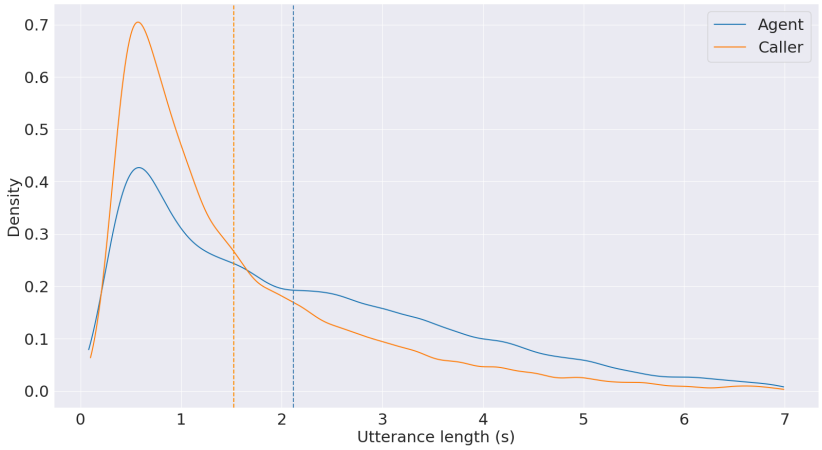

Figure 3.1: Probability density plot of utterance lengths for agent (blue) and caller (orange) speakers. The vertical lines represents the average utterance length for the caller and agents speakers.

Secondly, we analysed the frequency of speaker utterances, i.e., how often a speaker can be expected to speak. Figure 3.2 shows the probability distribution function of speaker frequencies, e.g., the number of utterances spoken by each speaker. The dotted line shows the average number of utterances for the caller (orange) and agent (blue) speakers. Notice that, in addition to having shorter utterances on average, the caller also has fewer utterances meaning that the caller is less active than the agent. Since the agent utterances are longer and more frequent than caller utterances, it is possible that a diarisation system may correctly label agent utterances and incorrectly label caller utterances and still have

a low error rate since the majority of the call is still correctly diarised.

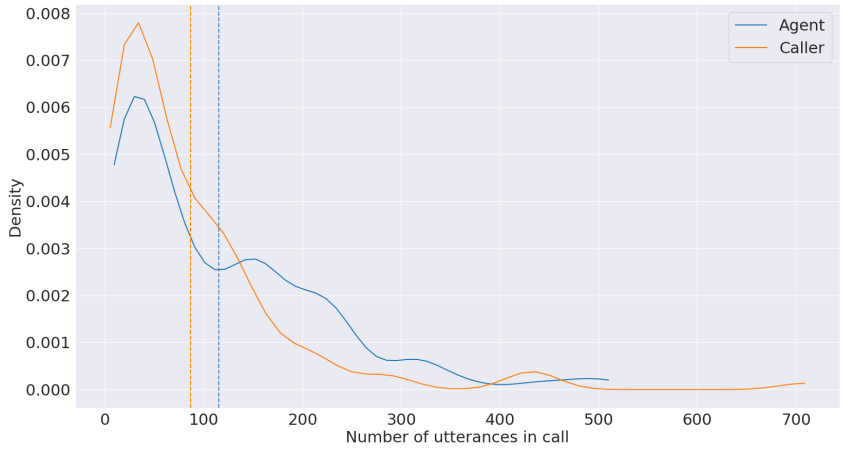

Figure 3.2: Probability density plot of the frequency of agent (blue) and caller (orange) utterances. The vertical lines represents the average number of utterances for the caller and agent speakers.

Lastly, Table 3.3 shows the composition of speech and non-speech events in the SAIGEN corpus as the average percentage of the total time in which a particular event occurs. Speech is divided into four categories: speech spoken by the agent, speech spoken by the caller, overlapped speech and background speech. Similarly, non-speech is divided into four categories: silence, music that plays when a caller is placed on hold, ringing that occurs at the start of calls, and miscellaneous events that refer to any unclassified noise, such as the sound of a car alarm. Notice that the majority of speech belongs to the agent and caller, and the agent is more than twice as active on average as the caller. Other speech events such as overlapped speech and background speech account for a relatively small portion of the speech. It is also apparent that most of the corpus is speech, as non-speech makes up a much smaller portion of the call. The most common non-speech event is silence.

Table 3.3: Composition of speech/non-speech events in the SAIGEN corpus.

Event type		Average percentage of time	
Speech	Agent	48.36	72.64
	Caller	20.73	
	Overlap	3.04	
	Background	0.51	
Non-speech	Silence	22.09	27.58
	Music	1.31	
	Ring	2.20	
	Miscellaneous	1.98	

3.4 Conclusion

This chapter introduced the EMRAI and SAIGEN corpora and gives an overview of their domains and how they are used in this study. The SAIGEN corpus is discussed in more detail, especially with regards to the annotation and verification protocols and the composition of the SAIGEN corpus is analysed in terms of the type and amount of speech and non-speech it contains.

In the next chapter, a preliminary evaluation is conducted on pre-trained x-vector and SAD systems using the EMRAI corpus and different methods of domain adaptation are compared and discussed. A suitable baseline system for the SAIGEN corpus is then chosen based on this evaluation.

Chapter 4

Evaluation of pre-trained systems

Various pre-trained, state-of-the-art x-vector and SAD models are described, evaluated and compared using the EMRAI corpus. Additionally, domain adaptation is investigated on the EMRAI corpus by retraining the RG and LDA transforms and performing PLDA interpolation.

4.1 Introduction

This chapter identifies several publicly available, pre-trained and state-of-the-art x-vector and SAD systems (Section 4.2). Section 4.3 compares the baseline performance of the pre-trained systems on the EMRAI corpus and contains a preliminary investigation on domain adaptation wherein we adapt the statistical components of the pre-trained pipelines to the EMRAI corpus.

As mentioned in Section 3.2, the EMRAI corpus served as a proxy while the SAIGEN corpus was not yet available. The aim of this chapter, therefore, is to familiarise the reader with the use and setup of x-vector based diarisation pipelines and domain adaptation and

not to directly compare adaptation methods. Direct comparisons of domain adaptation methods are found in Chapter 5.

4.2 Pre-trained systems

In this section the pre-trained SAD and x-vector models are investigated and we describe the architecture, training protocols and training datasets of each model. All the pre-trained models used in this study, with the exception of the TDNN-SAD, are open source and publicly available. It should be mentioned that no domain adaptation will be applied to the SAD models in any way, as that is beyond the scope of this research.

4.2.1 Pre-trained SAD baselines

WebrtcVAD

The first SAD model, known as WebrtcVAD[1], is widely used as a baseline in speaker diarisation systems [32], [33], [59], [67]. WebrtcVAD consists of a GMM trained to classify a given audio-frame as speech or non-speech using two full covariance Gaussians: one for speech and one for non-speech. Although DNN-based models generally perform better when trained on a specific domain, WebrtcVAD is sometimes preferred as it had been shown to generalise better to unseen domains [32].

TDNN-SAD

The second SAD model, which we will refer to as TDNN-SAD, is used by SAIGEN in ASR applications. The TDNN-SAD is a TDNN trained for voice classification and accepts 40-dimensional MFCCs with a window size of 30ms extracted every 25ms as input and consists of 5 time-delay layers followed by 2 statistical pooling layers. The TDNN was

[1]Python implementation available at: `https://pypi.org/project/webrtcvad/`

trained on narrow-band telephone speech from South African call centres and has two output classes: speech and non-speech.

LSTM-SAD

The last SAD model is an LSTM trained for speech activity detection[2] on the DIHARD speaker diarisation dataset [59]. The SAD consists of two bidirectional LSTM layers of size 128 followed by three feed-forward layers of the same size [68]. As with the TDNN-SAD, this model has two output classes: speech and non-speech.

4.2.2 Pre-trained x-vector models

In this section we investigate and explain the architectures of the pre-trained x-vector systems that are used for speaker embedding and diarisation. The usefulness of an x-vector model is determined by how much acoustic information the model needs to extract a meaningful representation [25], i.e., what is the shortest possible utterance length on which an embedding can be extracted which can still be used to discriminate among speakers?

In our case, because a large ID dataset is unavailable, we resort to using pre-trained x-vector models so that the generalisation capability afforded by large OOD datasets can be leveraged. We investigate three different pre-trained models: The Extended Time Delay Neural Network (E-TDNN) baseline [69]; the BUT ResNet101 baseline, which was used by the Brno University of Technology (BUT) in the second place submission to the VoxConverse Diarisation Challenge [70]; and lastly, the BUT DIHARD baseline, which was used in the winning submission of the second DIHARD diarisation challenge [48].

All the pre-trained x-vector systems are trained on the same data: the development sets of the Voxceleb1 [71], and Voxceleb2 [72] datasets which contain over 1.2 million utterances from 7,146 speakers. During pre-training, additional utterances were generated

[2]Model available at: `https://github.com/pyannote/pyannote-audio-hub/tree/master/models`

by following the data augmentation protocol proposed by Snyder et al. [5] in the original x-vector Kaldi recipe[3]. The data augmentation protocol creates multiple copies of the training data with different levels of background speech, noise, music and reverberation.

E-TDNN baseline

The Extended Time Delay Neural Network E-TDNN [69] is based on the standard x-vector architecture [5] but has 13 layers instead of 9. The E-TDNN network accepts 23-dimensional, mean-normalised MFCCs with a frame-length of 25ms extracted every 10ms. The architecture of the E-TDNN network is shown in Table 4.1. The first 10 layers are time-delay layers operating on speech frames followed by a single statistics pooling layer, a single fully connected layer from which the speaker embeddings are extracted and a softmax layer used only during training.

This particular E-TDNN network, and the accompanying PLDA model, was trained for speaker recognition on 16kHz audio from the Voxceleb1 and Voxceleb2 datasets with data augmentation, which provided an additional 5,000,000 segments. The architecture was trained for 6 epochs using categorical cross-entropy loss and Stochastic Gradient Descent (SGD).

[3]Available at: `https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2`

Table 4.1: The network architecture used in the E-TDNN system [69]. K represents the input feature dimensionality, T is the number of input frames and N is the number of speakers. The time-delay layers are centered around frame t.

Layer	Layer context	(Input) x output
frame 1	[t-2,t+2]	(2xK) x 512
frame 2	{t}	512 x 512
frame 3	{t-2,t,t+2}	(3x512) x (512)
frame 4	{t}	512 x 512
frame 5	{t-3,t,t+3}	(3x512) x 512
frame 6	{t}	512 x 512
frame 7	{t-4,t,t+4}	(3x512) x 512
frame 8	{t}	512 x 512
frame 9	{t}	512 x 512
frame 10	{t}	512 x 1500
stats pooling (mean + stddev)	[0,T]	(2x1500) x 512
segment1	{0}	512 x 512
softmax	{0}	512 x N

This x-vector system is used in a larger diarisation pipeline[4] which incorporates K-Means clustering and PLDA scores to perform diarisation. The E-TDNN system performs diarisation as follows:

1. Perform SAD on the input audio using either a SAD model or oracle SAD marks. Refer to Section 2.3.1.

2. Divide long speech segments into shorter, non-overlapping speech segments with each segment having a maximum length of 2 seconds. Segments shorter than 2 seconds are allowed when necessary.

3. Using the E-TDNN described in Table 4.1, extract an x-vector for each speech

[4]Available at: `https://github.com/Jamiroquai88/VBDiarization`

segment.

4. Train the RG, LDA and PLDA components on the extracted x-vectors. Before training the PLDA model, use the LDA transform to reduce the dimensionality of the x-vectors from 512 to 150.

5. Perform spherical K-Means clustering on the LDA-projected x-vectors[5]. Spherical K-Means is similar to vanilla K-Means, as described in Section 2.6.1, but uses cosine distance to cluster data points such that the data points in each cluster have a similar cosine similarity. Centroids are initialised randomly and the number of centroids is equal to the number of speakers present in the audio dialogue (2 in the case of EMRAI).

6. Perform vanilla K-Means (Section 2.6.1) but use the PLDA scores between embeddings as the distance metric. Use the centroids obtained from the previous clustering step as initialisation.

7. The timestamps of each speaker embedding and their associated clusters, or speaker labels, are then written to a .rttm file which constitutes the diarised output [18].

BUT DIHARD baseline

The BUT DIHARD baseline [48] is architecturally similar to the E-TDNN baseline but has 12 layers instead of 13, and the statistics pooling layer aggregates across two time-delay layers and not just the preceding layer. The TDNN operates on 40-dimensional, mean-normalised FBanks with a frame-length of 25ms extracted every 10ms. The architecture for the TDNN network is shown in Table 4.2. The first 9 layers are time-delay layers operating on FBank frames. The statistic pooling layer aggregates the mean and standard deviation across frames 7 and 9. Speaker embeddings are obtained from layer 12, and the softmax layer is discarded after training.

[5]The python implementation of spherical K-Means is available at `https://github.com/jasonlaska/spherecluster`

This network, and the accompanying PLDA model, was trained for speaker recognition on 16kHz audio from the Voxceleb1 and Voxceleb2 datasets with data augmentation, which provided an additional 5,000,000 segments. The network was trained for 3 epochs using categorical cross-entropy loss and SGD.

Table 4.2: The network architecture of the BUT DIHARD system [48]. K represents the input feature dimensionality, T is the number of input frames and N is the number of speakers. The time-delay layers are centered around frame t.

Layer	Layer context	(Input) x output
frame1	{t-2,t-1,t,t+1,t+2}	(5xK) x 1024
frame2	{t}	1024x1024
frame3	{t-4,t-2,t,t+2,t+4}	(5 X 1024) x 1024
frame4	{t}	1024 X 1024
frame5	{t-3,t,t+3}	(3 X 1024) X 1024
frame6	{t}	(1024 X 1024)
frame7	{t-4,t,t+4}	(3 X 1024) X 1024
frame8	{t}	1024 X 1024
frame9	{t}	1024 X 2000
stats pooling (frame7,frame9)	[0,T]	2*1024+2*2000 X 512
segment1	[0]	512 X 512
softmax	[0]	512 X N

This x-vector system is used in a larger diarisation pipeline[6] which uses VBHMM clustering to perform diarisation. The BUT DIHARD system performs diarisation as follows:

1. Perform SAD on the input audio using either a SAD model or oracle SAD marks.
2. Divide long speech segments into shorter, overlapping speech segments, with each segment having a maximum length of 1.5 seconds and a 250ms step between each segment. Segments shorter than 1.5 seconds are allowed when necessary.

[6]Available at: `https://github.com/BUTSpeechFIT/VBx/tree/v1.0_DIHARDII`

3. Using the TDNN described in Table 4.2, extract an x-vector for each speech segment.

4. Train the RG, LDA and PLDA components on the extracted x-vectors. Before training the PLDA model, use the LDA transform to reduce the dimensionality of the x-vectors from 512 to 128.

5. Perform VBHMM clustering on the x-vectors, as described in Section 2.6.3. During clustering, if two adjacent overlapping segments belong to different speakers, the speaker change point is taken as the midpoint of the two adjacent segments.

6. The timestamps of each speaker embedding and their associated clusters, or speaker labels, are then written to a .rttm file which constitutes the diarised output [18].

BUT ResNet101 baseline

The BUT ResNet101 system is a TDNN which uses a ResNet101 architecture [70]: a 2-dimensional CNN with residual connections between layers originally proposed by Kaiming et al. [73] for image recognition. The network uses 64-dimensional FBanks with a frame length of 25ms extracted every 10ms as input which is passed to a convolution layer followed by 4 standard ResNet blocks. A statistics pooling layer is applied after the ResNet blocks, followed by a single fully connected layer. The network architecture is shown in Table 4.3.

This network, and the accompanying PLDA model, was trained for speaker recognition on 16kHz audio from Voxceleb1, Voxceleb2, and CN-CELEB [74] with data augmentation, which amounted to roughly 89,000,000 additional segments. The network was trained for 3 epochs using additive angular margin loss [75] and SGD.

Table 4.3: The ResNet 101 architecture [70]. The first dimension of the input shows the size of the filterbank and the second dimension (T) indicates the number of input frames.

Layer	Structure		Stride	Output
Input	-		-	64xTx1
Conv2D-1	3 x 3, 32		1	64xTx32
ResNetBlock-1	$\begin{bmatrix} 1 \times 1, 32 \\ 3 \times 3, 32 \\ 1 \times 1, 128 \end{bmatrix}$	$\times 3$	1	64xTx128
ResNetBlock-2	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$	$\times 4$	2	32xT/2x256
ResNetBlock-3	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix}$	$\times 23$	2	16xT/4x512
ResNetBlock-4	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix}$	$\times 3$	2	8xT/8x1024
Stats pooling	-		-	16x1024
Flatten	-		-	16384
Linear	-		-	256

This x-vector system is used in a larger diarisation pipeline[7] and uses VBHMM clustering to perform diarisation. The BUT ResNet-101 system performs diarisation as follows:

1. Perform SAD on the input audio using either a SAD model or oracle SAD marks.
2. Divide long speech segments into shorter, overlapping speech segments, with each segment having a maximum length of 1.5 seconds and a 250ms step between each segment. Segments shorter than 1.5 seconds are allowed when necessary.

[7]Available at: `https://github.com/BUTSpeechFIT/VBx`

3. Using the TDNN described in Table 4.3, extract an x-vector for each speech segment.

4. Train the RG, LDA and PLDA components on the extracted x-vectors. Before training the PLDA model, use the LDA transform to reduce the dimensionality of the x-vectors from 258 to 128.

5. Perform VBHMM clustering on the x-vectors, as described in Section 2.6.3. During clustering, if two adjacent overlapping segments belong to different speakers, the speaker change point is taken as the midpoint of the two adjacent segments.

6. The timestamps of each speaker embedding and their associated clusters, or speaker labels, are then written to a .rttm file which constitutes the diarised output [18].

The main differences between the three x-vector systems is that the E-TDNN uses 2 second segments without overlap while the BUT DIHARD and BUT ResNet101 systems use 1.5 second segments with a 250ms step between segments (1.75 seconds overlap). Furthermore, the E-TDNN system uses K-Means clustering while the other two systems use VBHMM clustering.

4.3 Evaluating pre-trained systems

This section evaluates the various pre-trained systems (Sections 4.3.2 and 4.3.3), as described in Section 4.2.2. We also perform a preliminary evaluation of the domain adaptation techniques described in Section 2.5, namely retraining the RG and LDA transforms (Section 4.3.4) and PLDA interpolation (Section 4.3.5). For all experiments in this section, we use the 'dev-other' subset of the EMRAI corpus with data augmentation, as described in our experimental setup (Section 4.3.1), unless otherwise indicated.

4.3.1 Experimental setup

As mentioned in Section 3.2, we use randomly sampled noise from the 'free sound' subset of the MUSAN noise corpus for data augmentation. We create 6 versions of the 'dev-other' subset of the EMRAI corpus, each with a different level of background noise added with uniform intensity. The 6 noise levels (in SNR) are 5dB, 3dB, 0dB, -3dB and -5dB respectively with an additional subset containing no noise. The same noisy segments are identically added in each noise level. Therefore, the only difference between noisy subsets is the average noise level, e.g., the 5dB subset will have an average Signal-to-Noise-Ratio (SNR) of 5dB when measured over the entire set. The actual background noise is the same for all sets.

As shown in Section 4.2.2, the pre-trained x-vector systems are accompanied by their own diarisation pipeline, with each system following its own method of segmentation and clustering. The pre-trained x-vector systems and accompanying diarisation pipelines are initially used 'as-is' ,i.e., we do not change the segmentation parameters or clustering methods used by the systems as described in Section 4.2.2. The default system settings are revisited in Section 5.3 where we define our baseline system. It is also important to note that we use the diarisation pipeline to identify and label different speakers on a per-dialogue basis and not identify the same speakers in different calls (as that would be speaker recognition).

Whilst evaluating the BUT DIHARD and BUT ResNet101 systems, we use a grid search to approximate the optimal hyperparameters for the VBHMM clustering backend. The specific ranges over which the grid searches are executed are shown in Table A.1. When using VBHMM clustering, we provide the number of speakers or HMM states beforehand since it is also required by the K-Means steps in the E-TDNN pipeline.

All error metrics, with the exception of the F1-Score, are calculated using a standard 250ms collar. During scoring, overlapped speech is ignored and diarisation is performed using oracle SAD marks.

4.3.2 SAD systems: no adaptation

Method

To identify the best performing SAD system, we compare the performance of each pre-trained model on all 6 noise levels of the 'dev-other' subset of the EMRAI corpus. We compare performance in terms of the F1-score, as described in Section 2.7.1. All SAD models are used 'as-is'.

To compute the F1-score, we first convert the oracle and system SAD marks (given as timestamps) to binary arrays where a 0 represents non-speech and a 1 represents speech. Each digit in the binary array corresponds to 10ms of audio, which is the resolution of all three baseline models. The F1-score is then computed using the binary array of the oracle SAD marks as ground truth, and the binary array of the system SAD marks as predictions.

Results

Table 4.4 shows the mean F1-score of each SAD baseline for each of the 6 noise levels of the EMRAI 'dev-other' subset. Also shown is the average F1-score over all noise levels. The TDNN-SAD showed the best performance, followed by WebrtcVAD which only marginally outperformed the TDNN-SAD on high noise levels (greater than 0dB). The LSTM-SAD performed the worst, but its performance is still comparable to the other two systems at relatively low noise levels (less than 0dB).

Table 4.4: Mean F1-score for each noise level of the EMRAI 'dev-other' subsets using the baseline SAD models. Also shown is the average F1-score over all noise levels. The error is the standard deviation across dialogues within each subset.

Noise level	Webrtcvad	LSTM-SAD	TDNN
no noise	0.934 ± 0.009	0.931 ± 0.018	**0.952 ± 0.012**
SNR 5	0.932 ± 0.013	0.931 ± 0.021	**0.935 ± 0.010**
SNR 3	0.931 ± 0.013	0.930 ± 0.019	**0.932 ± 0.010**
SNR 0	0.929 ± 0.014	0.922 ± 0.024	**0.930 ± 0.013**
SNR -3	**0.927 ± 0.013**	0.898 ± 0.040	0.920 ± 0.015
SNR -5	**0.926 ± 0.012**	0.877 ± 0.078	0.913 ± 0.023
All	0.929	0.915	**0.930**

4.3.3 X-vector systems: no adaptation

Method

To compare the 'out-of-the-box' performance of each pre-trained x-vector system, as described in Section 4.2.2, we measure the diarisation performance of each pre-trained system over all 6 noise levels of the EMRAI 'dev-other' subset without applying any domain adaptation. The diarisation performance is compared in terms of DER and JER and we used oracle SAD marks.

Results

Table 4.5 shown the mean DER and JER of each pre-trained x-vector system calculated over all 6 noise levels of the EMRAI 'dev-other' subset. Notice that the BUT DIHARD shows the lowest error rate, in terms of mean performance over all noise levels. Similarly, Table 4.6 shows the mean DER and JER for all 6 noise levels respectively. Notice that none of the pre-trained systems clearly outperforms the other two on all noise levels and

that the E-TDNN and BUT DIHARD systems have a lower mean error rate compared to the ResNet101 on most noise levels with the ResNet101 only having a lower mean error rate than the E-TDNN at -5dB.

Table 4.5: Mean DER and JER of each pre-trained x-vector system taken over all 6 noise levels of the EMRAI 'dev-other' subset. The x-vector systems are implemented without domain adaptation. The error margin is the standard deviation of the mean diarisation error between noise levels.

Mean error (%)	E-TDNN	BUT DIHARD	BUT ResNet101
DER	10.38 ± 8.13	**8.91 ± 3.59**	11.95 ± 6.12
JER	15.04 ± 8.96	**12.75 ± 5.54**	17.64 ± 9.33

Table 4.6: Mean diarisation error taken over all noise levels of the EMRAI 'dev-other' subset without any domain adaptation. Results are reported using Oracle SAD and the error margin is the standard deviation within noise levels.

Noise Level	E-TDNN		BUT DIHARD		BUT ResNet101	
	DER (%)	JER (%)	DER (%)	JER (%)	DER (%)	JER (%)
no noise	4.46 ± 2.08	6.85 ± 3.93	5.56 ± 8.59	7.43 ± 13.74	5.09 ± 8.59	6.92 ± 10.11
5dB	6.18 ± 3.41	10.06 ± 5.89	5.97 ± 8.46	8.28 ± 13.62	7.03 ± 8.68	10.15 ± 7.01
3dB	6.54 ± 3.39	10.71 ± 5.96	7.15 ± 10.34	10.14 ± 16.19	9.62 ± 12.06	14.10 ± 17.45
0dB	8.56 ± 6.94	13.89 ± 10.35	9.36 ± 13.44	13.27 ± 19.92	13.33 ± 14.27	19.65 ± 19.88
-3dB	10.09 ± 6.89	16.72 ± 10.33	10.17 ± 12.70	15.01 ± 19.08	14.60 ± 12.73	22.36 ± 18.67
-5dB	26.50 ± 13.29	31.98 ± 17.61	15.22 ± 16.32	22.40 ± 23.39	22.02 ± 15.39	32.65 ± 21.17

4.3.4 X-vector systems: retrained RG and LDA transforms

Method

For domain adaptation we first investigate the utility of using a retrained RG and LDA transform, as described in Section 2.5.1, and we compare the performance of the pre-trained systems using these retrained transforms. The process of retraining a RG and LDA

transform is depicted in Figure 4.1: the transforms are retrained on x-vectors extracted from ID data (EMRAI in this case) using a pre-trained x-vector system. After the ID transforms are trained, we train a PLDA on OOD x-vectors, which were transformed using the ID RG and LDA transforms. OOD x-vectors are extracted from the same data on which the E-TDNN and BUT DIHARD baselines are trained (Voxceleb1 and Voxceleb2 with data augmentation). The ID and OOD x-vectors are extracted on ID and OOD data respectively, using the same pre-trained x-vector system.

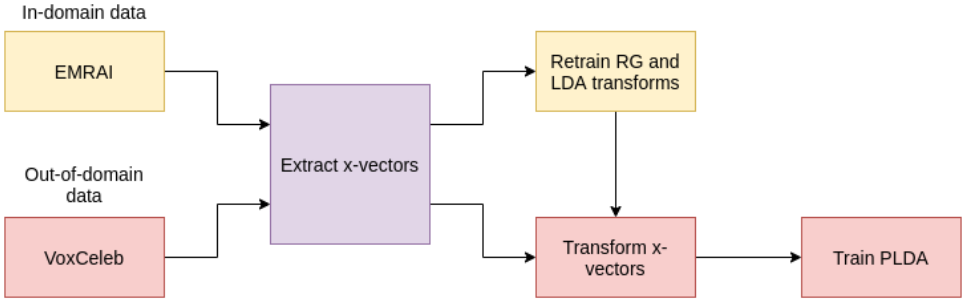

Figure 4.1: Process for retraining an ID RG and LDA transform.

We approached the retraining of the transforms in two ways: our first approach, referred to as 'all-noise-levels', uses all 6 noise levels of the EMRAI 'dev-other' subset as training data. In our second approach, referred to as 'single-noise-level', we train and implement a separate RG and LDA transform for each noise level. This is done to determine how sensitive the transforms are to noise and how much data the transforms need for training as using 6 noise levels simultaneously increases the size of the dataset six fold. However, since the added noise is the same for each noise level this would not necessarily have the same effect as actually training on a larger dataset. Apart from the retrained transforms the x-vector systems remain unaltered. We compare the diarisation performance of the adapted systems in terms of DER and JER using oracle SAD marks. As this chapter is only a preliminary investigation a complete evaluation and comparison of retrained RG and LDA transforms can be found in Section 5.4.1 using the SAIGEN dataset.

Results

Table 4.7 shows the mean DER and JER taken over all 6 noise levels of the EMRAI 'dev-other' subset using retrained RG and LDA transforms following the 'all-noise-level' and 'single-noise-level' approaches. The mean DER and JER of the unadapted systems are also included for comparison. The E-TDNN and BUT DIHARD baselines show a significant increase in diarisation performance when using retrained transforms with both systems showing comparable performance. The E-TDNN performed the best across all metrics. The 'single-noise-level' training approach performed marginally better than the 'all-noise-levels' alternative, but shows no real advantage since the 'single-noise-level' approach requires the transforms to be retrained 6 times and complicates the adaptation process without yielding significant gains.

Additionally, the mean error rate for each noise level of the 'dev-other' subset for both the 'all-noise-levels' and 'single-noise-level' approaches can be seen in Tables A.2 and A.3 respectively. Notice that, in the case of the BUT ResNet101 baseline, the retrained transforms show an increased performance on lower noise levels compared to the unadapted system, but the diarisation performance decreases with higher noise levels (noise levels of 0dB and higher). Hence, for the BUT ResNet101 system, the use of retrained transforms are still beneficial to diarisation performance but the efficacy thereof degrades as the noise level increases.

Table 4.7: Mean DER and JER of each pre-trained x-vector system taken over all 6 noise levels of the EMRAI 'dev-other' subset. The x-vector systems are each implemented using a RG and LDA transform that had been retrained on the EMRAI corpus following the all-noise-levels and single-noise-level approaches. The performance of the pre-trained systems without any domain adaptation are included for comparison and the error margin is the standard deviation between the mean diarisation error of noise levels.

Adaptation	Mean error (%)	E-TDNN	BUT DIHARD	BUT ResNet101
No adaptation	DER	10.38 ± 8.13	**8.91 ± 3.59**	11.95 ± 6.12
	JER	15.04 ± 8.96	**12.75 ± 5.54**	17.64 ± 9.33
Retrained transforms, all-noise-levels	DER	**4.91 ± 1.97**	7.10 ± 4.51	12.34 ± 8.07
	JER	**7.64 ± 3.66**	10.14 ± 7.25	18.10 ± 12.18
Retrained transforms, single-noise-level	DER	**4.58 ± 1.71**	7.51 ± 5.30	12.35 ± 7.60
	JER	**6.98 ± 3.20**	10.81 ± 8.46	18.10 ± 11.66

4.3.5 X-vector systems: PLDA interpolation

Method

For our third and final evaluation, we investigate the use of PLDA interpolation, as described in Section 2.5.2, and we compare the performance of the pre-trained x-vector systems using interpolated PLDA models. The process of training and interpolating ID and OOD PLDA models is depicted in Figure 4.2. In addition to retraining the RG and LDA transforms, we train an ID PLDA model using the retrained transforms, which is then interpolated with the OOD PLDA model as shown in Equation 2.12. The optimal interpolation parameter α is estimated by comparing the mean diarisation error over the ID dataset for $\alpha \in [0.5, 1.0]$ and selecting the α with the lowest corresponding error rate.

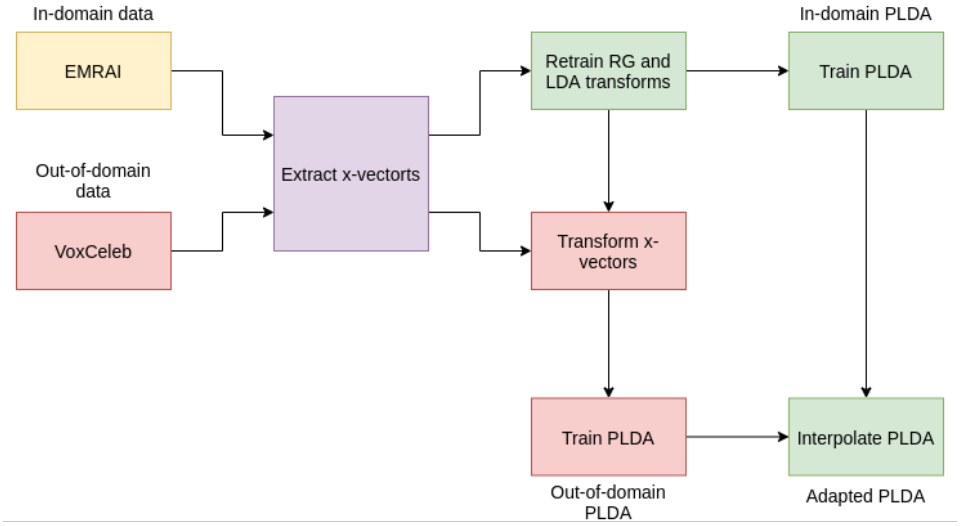

Figure 4.2: Process for training and interpolating ID and OOD PLDA models.

As with the retrained transforms, we approach the PLDA training and interpolation in two ways: for our first approach, referred to as 'all-noise-levels', we retrained the ID components on all 6 noise levels of the EMRAI 'dev-other' subset. For our second approach, referred to as 'single-noise-level', we retrained separate ID components on each noise level. Apart from the retrained transforms and interpolated PLDA models, the x-vector systems remain unaltered. We compare the diarisation performance of the adapted systems in terms of DER and JER using oracle SAD marks. As this chapter is only a preliminary investigation a complete evaluation and comparison of retrained PLDA interpolation can be found in Section 5.4.1 using the SAIGEN dataset.

Results

Table 4.8 shows the mean DER and JER taken over all 6 noise levels of the EMRAI 'dev-other' subset using PLDA interpolation following the 'all-noise-levels' and 'single-noise-level' training approaches. Notice that the PLDA interpolation provides an increase in diarisation performance across all pre-trained systems. As with the retrained transforms in Section 4.3.4, the 'single-noise-level' showed no significant improvement over the 'all-noise-levels' approach.

Additionally, the mean error rate for each noise level of the 'dev-other' subset with PLDA interpolation for both the 'all-noise-levels' and 'single-noise-level' training approaches are shown Tables A.4 and A.5 respectively. Notice that, for all pre-trained systems, the diarisation performance is less varied between noise levels compared to only using retrained transforms which indicates better robustness across different levels of background noise.

Table 4.8: Mean DER and JER of each pre-trained x-vector system taken over all 6 noise levels of the EMRAI 'dev-other' subset. The x-vector systems are each implemented using PLDA interpolation with the ID PLDA being trained on the EMRAI corpus. The results for both the all-noise-levels and single-noise-level approaches are shown and the performance of the pre-trained systems without any domain adaptation are included for comparison. The error margin is the standard deviation between the mean diarisation error of noise levels.

Adaptation	Mean error (%)	E-TDNN	BUT DIHARD	BUT ResNet101
No adaptation	DER	10.38 ± 8.13	**8.91 ± 3.59**	11.95 ± 6.12
	JER	15.04 ± 8.96	**12.75 ± 5.54**	17.64 ± 9.33
Retrained Transforms, all-noise-levels	DER	**4.91 ± 1.97**	7.10 ± 4.51	12.34 ± 8.07
	JER	**7.64 ± 3.66**	10.14 ± 7.25	18.10 ± 12.18
Retrained Transforms, single-noise-level	DER	**4.58 ± 1.71**	7.51 ± 5.30	12.35 ± 7.60
	JER	**6.98 ± 3.20**	10.81 ± 8.46	18.10 ± 11.66
PLDA Interpolation, all-noise-levels	DER	**4.44 ± 1.54**	4.52 ± 1.00	6.80 ± 3.34
	JER	6.78 ± 2.95	**6.02 ± 1.94**	9.91 ± 5.70
PLDA Interpolation, single-noise-level	DER	**4.10 ± 1.52**	4.75 ± 1.35	6.96 ± 3.05
	JER	**6.03 ± 2.74**	6.42 ± 2.51	10.35 ± 5.26

4.4 Conclusion

This chapter presented a description and evaluation of the pre-trained SAD and x-vector models and domain adaptation techniques. The results of this evaluation are then used

to select a credible baseline for experimentation on the SAIGEN corpus (Chapter 5).

The main findings in this chapter are as follows:

- Prior to performing domain adaptation, the BUT DIHARD system showed the best average performance (Table 4.5) which can be attributed to its performance on high noise levels. However, the E-TDNN system achieved a similar, and in some cases better performance than the BUT DIHARD system on low noise levels (Table 4.6). The ResNet101 system achieved the lowest overall performance.
- All x-vector systems showed a noticeable increase in performance when using a retrained RG and LDA transform (Table 4.7). As this chapter only serves as a preliminary investigation, the diarisation performance was reported using the same data on which the transforms had been trained, the performance of retrained transforms on unseen ID data is investigated in the next chapter. Regardless, the E-TDNN system outperformed the BUT DIHARD and BUT ResNet101 systems when using retrained transforms, both in terms of DER and JER.
- The use of PLDA interpolation, together with the retrained transforms, showed an additional increase in the diarisation performance of all x-vector systems, most notably in the BUT DIHARD and BUT ResNet101 systems. The E-TDNN and BUT DIHARD systems showed similar performance with both systems performing better than the BUT Resnet101 system. As with the retrained transforms, results are reported using the same data on which the ID components had been retrained. The performance of PLDA interpolation on unseen ID data is investigated in the next chapter.
- In terms of SAD, the TDNN-SAD and WebrtcVAD showed similar performance across all noise levels, with the LSTM-SAD system performing the worst. The performance of the SAD systems in a real-world scenario are evaluated in the next chapter.

Given the performance increases afforded by retraining the RG and LDA transforms on in-domain data and interpolating ID and OOD PLDA models, these methods are used

as the starting point for domain adaptation in a real-world environment, as shown in the next chapter. Lastly, as neither of the pre-trained x-vector systems clearly outperformed the other systems over all metrics and noise levels (Table 4.6) we choose the E-TDNN system, due to its architectural simplicity, for all following experiments.

Chapter 5

System refinement

In this chapter a baseline system is established and adapted to the SAIGEN corpus, which reflects a real-world environment. The baseline system is adapted by retraining and adapting the statistical components on the SAIGEN corpus and a novel algorithm for fine-tuning PLDA interpolation using cluster analysis is introduced. Lastly, the use of the adapted systems for speaker-specific adaptations is investigated using a pre-trained ASR system.

5.1 Introduction

The previous chapter compared several pre-trained SAD and x-vector systems and contains a preliminary evaluation of domain adaptation on the EMRAI corpus using retrained RG and LDA transforms and PLDA interpolation. In this chapter, we adapt the pre-trained E-TDNN model, which we selected from the previous chapter, to the SAIGEN corpus.

All experiments in this chapter are performed on the SAIGEN corpus using the same manner of cross validation, as described in our experimental setup (Section 5.2). The baseline

results on the SAIGEN corpus are established using the unadapted, pre-trained E-TDNN system and the best performing pre-trained SAD model. After the baseline system had been established, the effects of the default system settings on the diarisation error are investigated (Section 5.3). Secondly, the baseline system is adapted to the SAIGEN domain using retrained RG and LDA transforms and PLDA interpolation and we introduce the use of silhouette coefficients to automatically fine-tune PLDA interpolation on a per-call basis (Section 5.4). Lastly, the performance of a pre-trained ASR system is compared with and without the use of the adapted system(s) for speaker-specific adaptations (Section 5.5).

5.2 Experimental setup

All experiments in this chapter are conducted on the SAIGEN corpus with no additional data augmentation unless otherwise indicated. Furthermore, all results are reported using cross validation: we create 3 development and test splits of the SAIGEN corpus. In each split, the SAIGEN corpus is divided into roughly an 80-20 split with the condition that the test set has to contain at least 1 call from each call centre. As such, when reporting the mean error rate, we first calculate the average error over each development and test split separately. Thereafter the mean error is reported as the mean of the development or test splits. When reporting the mean error, all error margins represent the standard error across splits. We did not use each call centre as its own split as the centres contain different amounts of dialogues and speakers.

Lastly, hypothesis testing is performed using the corrected resampled paired Student's t-test [76], unless stated otherwise[1]. All hypothesis tests are conducted as one-sided tests with the null hypothesis being rejected at $p_{value} < 0.05$. We report it as such, rather than only stating the significance level, in order not to confuse the significance level (usually α) with the interpolation parameter α, as used in this study.

The corrected resampled Student's t-test as proposed by Nadeau and Bengio is used

[1] A Wilcoxon test is used in Section 5.3.1 for reasons explained in that section.

because the development and test splits are generated via random subsampling. As shown by Dietterich [77], the standard Student's t-test has a high type I error when used with random subsampling, since this introduces an overlap between training sets which violates the t-test's assumption of independent training sets. The corrected resampled Student's t-test, however, takes this dependency into account, and the fact that not only the test set is variable, but also the training set [76]. Furthermore, paired tests are used to directly compare the performance of two models across the same datasets rather than comparing the performance of multiple models across the same datasets, as done in the Friedman test [78], for example.

It should be reiterated that, due to the computational cost of training and testing multiple folds, only 3 development and test splits are used, as opposed to the more ideal 10-fold cross-validation. The smaller sample size given by the 3 splits will, as a consequence, limit the statistical power of hypothesis testing. While we therefore utilise standard error to provide an indication of the variability of results (as is more typical in current machine learning studies) we also report on the significance of results, for the sake of completeness.

5.3 Baseline diarisation system

In this section we establish a baseline system to be used in all experimentation on the SAIGEN corpus. The baseline system, as depicted in Figure 5.1, uses the E-TDNN x-vector system, which had been chosen based on its performance on the EMRAI corpus in Sections 4.2.2 through 4.3.5.

To establish a baseline system, different SAD models are evaluated on the SAIGEN corpus and the best performing model is used when reporting results on system SAD (Section 5.3.1). Thereafter, the effects of the default system settings on diarisation performance is investigated to ensure that our baseline system is optimally configured (Sections 5.3.2 and 5.3.3). Finally, the performance of the unadapted baseline system is established on the SAIGEN corpus (section 5.3.4).

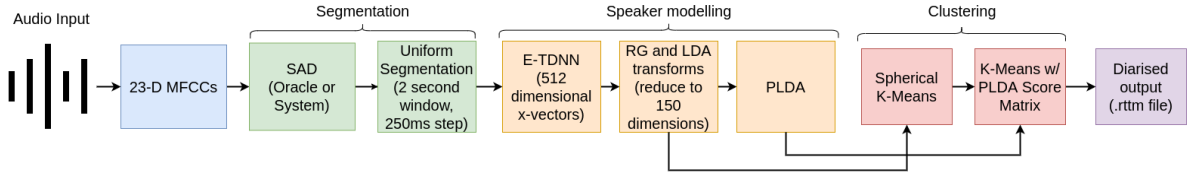

Figure 5.1: Our baseline system used for experimentation on the SAIGEN corpus.

5.3.1 Evaluation of SAD systems

Before establishing baseline results on the SAIGEN corpus, the pre-trained SAD models, as described in Section 4.2.1, are evaluated on the SAIGEN corpus and the best performing model is used when reporting results on system SAD.

Method

The SAD performance is compared in terms of F1-score, as described in Section 2.7.1. All SAD models are used 'as-is'. The F1-score is computed in the same manner as in Section 4.2.1: by converting the oracle and system SAD marks to binary arrays where a 0 represents non-speech and a 1 represents speech. Each digit in the binary array corresponds to 10ms of audio, which is the resolution of all three pre-trained models. The oracle SAD marks are used as ground truth, and the system SAD marks as predictions; the F1-score is calculated accordingly.

Results

Table 5.1 shows the mean F1-score for each SAD baseline over the entire SAIGEN corpus and for each call centre respectively. The TDNN-SAD performed significantly better than both the Webrtcvad and LSTM SAD systems (Wilcoxon, $p_{value} < 0.05$)[2]. Recall

[2]We use a Wilcoxon signed-rank test instead of the corrected resampled Student's t-test as the datasets (call-centres) are independent. Additionally, due to the small size of the dataset it cannot be reliably determined if the differences in performance of the 3 models are normally distributed; we therefore follow

from Section 4.2.1 that the WebrtcVAD showed comparable performance to the TDNN-SAD on the EMRAI corpus. This is, however, not true for the SAIGEN corpus where the TDNN-SAD consistently outperforms the other systems by a considerable margin. *The TDNN-SAD will therefore be used for all experiments in this chapter when reporting on system SAD.*

Table 5.1: Mean F1-score taken over each call-centre of the SAIGEN corpus using the baseline SAD models. The error margin is the standard deviation within each subset.

Subset	Webrtcvad	LSTM	TDNN
Call Centre 1	0.78 ± 0.56	0.76 ± 0.53	**0.86 ± 0.31**
Call Centre 2	0.81 ± 0.57	0.76 ± 0.77	**0.86 ± 0.38**
Call Centre 3	0.85 ± 0.43	0.80 ± 0.56	**0.89 ± 0.33**
Call Centre 4	0.82 ± 0.52	0.76 ± 0.74	**0.89 ± 0.26**
Call Centre 5	0.76 ± 0.89	0.69 ± 0.94	**0.82 ± 0.71**
Call Centre 6	0.75 ± 0.39	0.73 ± 0.47	**0.83 ± 0.39**
Call Centre 7	0.73 ± 0.11	0.69 ± 0.59	**0.80 ± 0.11**
Call Centre 8	0.67 ± 0.12	0.60 ± 0.88	**0.81 ± 0.71**
Call Centre 9	0.86 ± 0.40	0.85 ± 0.47	**0.90 ± 0.29**
Entire Corpus	0.76 ± 0.10	0.73 ± 0.89	**0.83 ± 0.79**

5.3.2 X-vector segmentation

In Chapter 4.3.1 the E-TDNN x-vector system is implemented 'as-is', without changing any of the default settings. In this subsection we investigate the effects of the segmentation settings on diarisation performance – all other default settings, such as the dimensionality of the LDA transform are not investigated in this study.

Demvsar's advice [78] and use the Wilcoxon test which is a non-parametric alternative to the Student's t-test.

Method

The E-TDNN system segments speech using a default 2-second sliding window without overlap (2000ms step between segments). Hence, baseline results are established by comparing diarisation performance on the SAIGEN corpus while varying the default step size between segments[3]. The segment size itself remains unchanged.

Results

Table 5.2 shows the mean DER taken over all three development and test splits of the SAIGEN corpus using the E-TDNN and TDNN-SAD with various segmentation step sizes. We used a constant sliding window length of 2 seconds and, for each segmentation setting, we used oracle SAD and TDNN-SAD (system SAD). Notice that the DER remains relatively constant across step sizes. Although no significant difference is observed between step sizes (resampled Student's t-test, $p_{value} > 0.05$), *we chose to perform all further experiments using a 250ms step size*, as it resulted in the lowest mean DER across the developments splits. Additionally, we see a higher mean DER on the test splits compared to development splits when using TDNN-SAD but not when using oracle SAD, this indicates that the TDNN-SAD performs worse on the test splits than the development splits and not the E-TDNN x-vector system itself. However, as SAD systems are not the focus of this research we do not attempt to adapt the TDNN-SAD to have similar performance on the development and test splits.

[3]As mentioned in Section 4.2.2, when overlapping segments are labelled as belonging to different speakers, the speaker change point is chosen as the midpoint between the two overlapping segments.

Table 5.2: Mean DER measured on the SAIGEN corpus using the TDNN-SAD and E-TDNN baseline systems over various segmentation settings. The DER is reported over all three development and test splits with the standard error.

Step Size (ms)	Oracle		TDNN-SAD	
	Development	Test	Development	Test
250	17.00 ± 0.28	17.20 ± 1.16	23.99 ± 0.35	28.23 ± 0.15
500	17.12 ± 0.34	17.55 ± 1.33	24.10 ± 0.19	28.58 ± 0.20
750	17.14 ± 0.56	17.15 ± 1.10	24.10 ± 0.20	27.75 ± 0.40
1000	17.07 ± 0.31	17.12 ± 1.39	24.02 ± 0.13	27.60 ± 0.34
2000	17.90 ± 0.44	17.31 ± 0.76	24.07 ± 0.18	28.42 ± 0.23

5.3.3 SAD segment padding

The TDNN-SAD is chosen for the baseline system due to its performance on the SAIGEN corpus. However, the TDNN-SAD applies segment padding as a post-processing step which is an algorithm that slightly pads segments with silence, to compensate for over aggressive boundary placement. While details of the segment padding algorithm are confidential, its effect on diarisation performance is unknown and should be verified.

Method

To investigate how segment padding affects diarisation performance we measure the mean DER over the SAIGEN corpus with and without segment padding. Additionally, we decompose the DER into its core components: false alarm, missed detection and speaker confusion. This decomposition allows us to determine how segment padding effects diarisation.

Results

Table 5.3 shows the decomposed DER taken over the SAIGEN corpus using TDNN-SAD with and without segment padding. The missed detection component of the DER is lower when using segment padding as opposed to no segment padding (corrected resampled Student's t-test, $p_{value} < 0.05$), which indicates the padded segments help to include speech which the TDNN-SAD missed. The opposite is true for the false alarm rate as it is higher when using segment padding (corrected resampled Student's t-test, $p_{value} < 0.05$), indicating that, although segment padding can include missed speech, it also inevitably includes non-speech. Lastly, although the mean speaker confusion rate is lower when using segment padding, no significant difference is observed (corrected resampled Student's t-test, $p_{value} > 0.05$). Therefore, seeing that segment padding improves the missed detection rate *we will use TDNN-SAD with segment padding for all future experiments and will simply refer to it as 'system-SAD'*, as it is better to include more silence (as given by the false-alarm rate) than to exclude speech altogether.

Table 5.3: Mean DER measured on the SAIGEN corpus using TDNN-SAD with and without segment padding and the E-TDNN baseline (using a 2 second sliding window with a 250ms step). The DER and all its components are reported over all three development and test splits. The error margin is the standard error.

Error (%)	Development		Test	
	TDNN-SAD with padding	TDNN-SAD without padding	TDNN-SAD with padding	TDNN-SAD without padding
False alarm	4.67 ± 0.18	**2.4 ± 0.21**	7.59 ± 0.34	**3.82 ± 0.26**
Missed detection	**2.51 ± 0.42**	5.14 ± 0.31	**3.77 ± 0.51**	8.41 ± 0.29
Speaker confusion	**16.82 ± 0.31**	17.81 ± 0.40	**16.81 ± 0.24**	17.64 ± 0.33
DER	**23.99 ± 0.35**	25.35 ± 0.30	**28.23 ± 0.15**	29.87 ± 0.26

5.3.4 Baseline performance

This section showed that diarisation performance is relatively unaffected by the step size between segments and that the TDNN-SAD benefits from using segment padding. The baseline system will therefore use the E-TDNN model with a 2 second window and 250ms step between windows and the TDNN-SAD will be used with segment padding. The performance of the baseline system on the SAIGEN corpus is shown in Table 5.4.

Table 5.4: Mean DER and JER error achieved by the unadapted baseline over all 3 development and test splits. Results are reported using oracle and system SAD and the error margin is the standard error.

Subset	SAD	DER (%)	JER(%)
Development	Oracle	17.00 ± 0.28	32.79 ± 0.46
	System	23.99 ± 0.35	45.84 ± 0.57
Test	Oracle	17.20 ± 1.16	33.66 ± 1.77
	System	28.23 ± 0.15	49.80 ± 1.23

5.4 Domain adaptation

Having established the baseline system and performance, the system can be adapted. In this section, the baseline system is adapted to the SAIGEN corpus by retraining and adapting the statistical components of the diarisation pipeline, as shown in Sections 4.3.4 and 4.3.5, but using the SAIGEN corpus instead of the EMRAI corpus. Additionally, we investigate the use of cluster analysis techniques, namely silhouette coefficients, to fine-tune PLDA interpolation. All hypothesis tests in this Section are summarised in Tables A.12 through A.15.

5.4.1 Retrained transforms and PLDA interpolation

Method

To retrain the RG and LDA transformations, we follow the same process as in Figure 4.1 but we exchange the EMRAI corpus for the SAIGEN corpus: we train ID transforms on the development set of the SAIGEN corpus and an OOD PLDA model on the x-vectors extracted from the development sets of Voxceleb1 and Voxceleb2 using the ID transforms. For PLDA interpolation we follow the same process as in Figure 4.2: we train an ID PLDA model on x-vectors extracted from the development sets of the SAIGEN corpus using the ID transforms and interpolate the ID and OOD PLDA models. The utterances used to train the ID models are obtained using the oracle SAD labels, and we perform domain adaption on each of the 3 dataset splits separately. When adapting the system to the SAIGEN corpus, we use the entire development set for training and not individual call centres as each call centre adds entirely new data, instead of the same data with different levels of background noise as in chapter 4.

During PLDA interpolation, the optimal interpolation parameter α (shown in Equation 2.12) is estimated on a held-out [47] set or on the entire development set [48]. Given the size of the SAIGEN corpus, using a held-out set might not allow for a development set that is large enough to sufficiently train an ID PLDA model. The optimal interpolation is therefore estimated by selecting the α in the range $\alpha \in [0.5, 1.0]$ which corresponds to the lowest mean diarisation error in each development split[4].

Results

Using the adapted system, we report the mean DER and JER over all splits using oracle SAD and system SAD in Figures 5.2 and 5.3 respectively. The use of retrained transforms provides significant performance increases in terms of both DER and JER regardless of the SAD used (corrected resampled Student's t-test, $P_{value} < 0.05$). Lastly, PLDA

[4]When choosing the α for interpolation we assume that the ID data is more important to diarisation performance than the OOD data, hence the range $\alpha \in [0.5, 1.0]$

interpolation showed similar performance compared to retrained transforms when using oracle and system SAD.

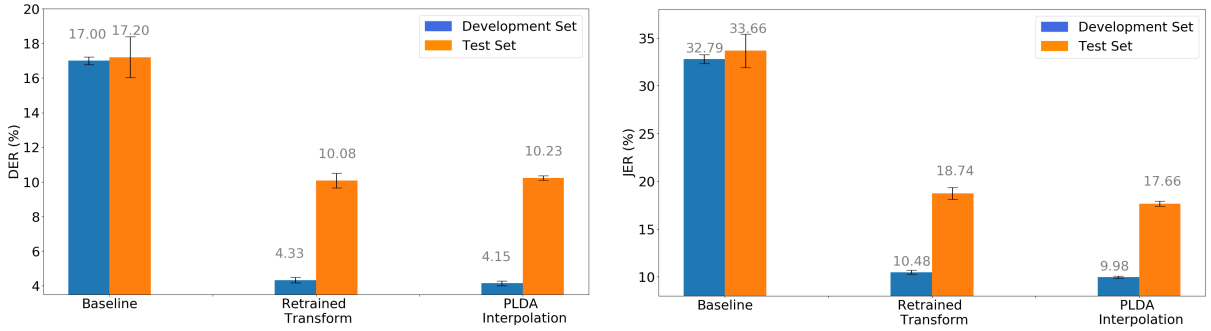

Figure 5.2: Mean DER and JER with and without domain adaptation taken over all development and test splits of the SAIGEN corpus. Results are reported using oracle SAD and the error margin is the standard error.

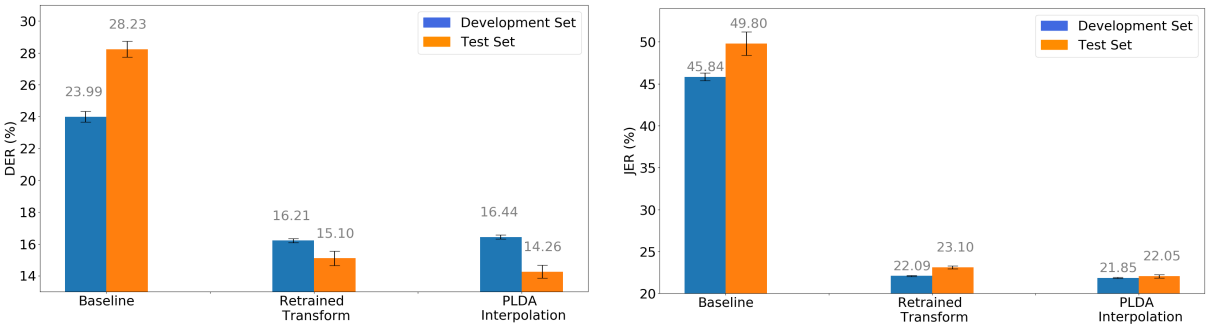

Figure 5.3: Mean DER and JER with and without domain adaptation taken over all development and test splits of the SAIGEN corpus. Results are reported using system SAD and the error margin is the standard error.

5.4.2 Fine-tuning PLDA interpolation using silhouette coefficients

As mentioned in Section 5.4.1, we fine-tune PLDA interpolation by estimating the optimal α parameter on the development set (a single α used for all calls). However, as the SAIGEN corpus contains calls from different call centres with different recording conditions and languages, each call does not necessarily have the same optimal α. Consequently,

we aim to obtain an optimal α for each call individually, in an unsupervised manner, by using silhouette coefficients to choose an optimal α based on cluster quality (per-file fine-tuning). In this way the silhouette coefficients serves as a proxy for diarisation error since we assume that there exists a correlation between the silhouette coefficients and the optimal α values and, by extension, diarisation error.

Silhouette Coefficients

Silhouette coefficients measure the quality of data clusters (after the data had been clustered) by comparing the distance between datapoints in the same cluster (intra-cluster distance) to the distance between datapoints in different clusters (inter-cluster distance) [79]. To calculate the silhouette coefficients of data clusters, we first define the average intra-cluster and inter-cluster distances of a datapoint i as $a(i)$ and $b(i)$ respectively, where:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} d(i, j) \tag{5.1}$$

and

$$b(i) = \min_{k \neq i} [\frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)] \tag{5.2}$$

Here C_i is the cluster that datapoint i belongs to and $|C_i|$ is the size of cluster C_i; $d(i, j)$ is the distance between datapoints i and j. C_k represents some other cluster than C_i with i and k in $\{1...N\}$. N is the total number of datapoints in the dataset. Thus, $a(i)$ is the average distance between datapoint i and all other datapoints in C_i and $b(i)$ is the average distance between datapoint i and all other datapoints in the closest cluster. The silhouette coefficient of datapoint i is now defined as:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}, \text{ if } |C_i| > 1 \tag{5.3}$$

with $-1 \leq s(i) \leq 1$. Since more than one datapoint is needed per cluster to calculate a silhouette coefficient, $s(i)$ is undefined in cases where $|C_i| = 1$. The silhouette coefficient over an entire dataset is defined as:

$$SC = \bar{s}(n) \tag{5.4}$$

Here $\bar{s}(n)$ represents the mean silhouette coefficient over the entire dataset which is then used to evaluate cluster quality: an SC (silhouette coefficient) value of 1 represents perfect clusters: the clusters have an arbitrary inter-cluster distance and an intra-cluster distance of 0. Conversely, an SC value of -1 represents no clustering i.e. clusters overlap completely and no distinction is made between classes.

Method

To determine the optimal α on a per-call basis we perform PLDA interpolation over a range of α values and choose the α resulting in the highest average Silhouette Coefficient (SC) for each call: a higher SC implies better (or more separated) clusters and therefore better diarisation performance. The procedure for per-file fine-tuning of PLDA interpolation is as follows:

1. Perform diarisation using PLDA interpolation on each call for $\alpha \in [0.5, 1.0]$ (an α of 0.5 marks the point where the ID and OOD PLDA models have an equal effect on interpolation, refer to Equation 2.12). The resulting output will be the K-Means clusters of speaker embeddings for each call and each α.
2. For each call, calculate the SC for each α.
3. For each call, identify the α with the highest corresponding SC.

In our experiments, we compare two methods of computing SC: The first method follows the standard modus operandi and uses the cosine distance between speaker embeddings to compute the average inter- and intra-cluster distances ('standard SC'). The second method uses the cosine distance between the columns of the PLDA score matrix to compute the average inter- and intra-cluster distances ('score matrix SC'). The intuition behind our second method is that, if the cosine distance between the n^{th} and m^{th} column of the PLDA score matrix is small, the n^{th} and m^{th} speaker embeddings have similar PLDA scores with respect to other embeddings and are therefore similar to each other and likely belong to the same speaker.

Results

To compare the per-file fine-tuning of PLDA interpolation to standard domain adaption we measure the mean DER and JER over all splits using oracle SAD and system SAD, as shown in Figures 5.4 and 5.5 respectively. The dotted lines represent the mean DER and JER using the true optimal α for each call, i.e, the mean error rate when manually choosing the α that results in the lowest error rate for each call. The 'score matrix SC' and 'standard SC' methods show similar performance with neither method clearly outperforming the other in terms of DER or JER (corrected resampled Student's t-test, $p_{value} > 0.05$).

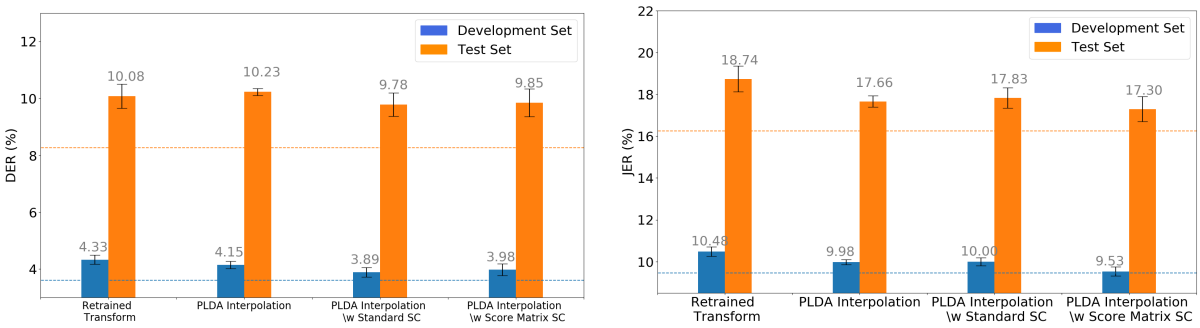

Figure 5.4: Mean DER and JER with and without the use of silhouette coefficients for fine-tuning the α parameter in PLDA interpolation. Results are computed over all development and test splits of the SAIGEN corpus using oracle SAD. The dotted lines represent the mean DER and JER when manually choosing the optimal α for each call in the development and test sets. The error margin is the standard error between splits.

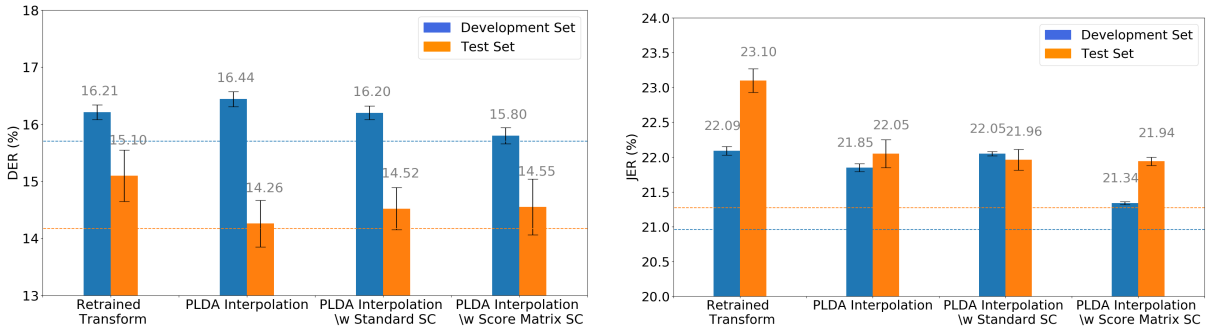

Figure 5.5: Mean DER and JER with and without the use of silhouette coefficients for fine-tuning the α parameter in PLDA interpolation. Results are computed over all development and test splits of the SAIGEN corpus using system SAD. The dotted lines represent the mean DER and JER when manually choosing the optimal α for each call in the development and test sets. The error margin is the standard error between splits.

The trade-off between a lower JER and DER depends on the downstream application as a low DER implies a low speaker confusion for the majority of speakers, and conversely, since JER is calculated as the average of per-speaker errors, a lower JER implies a low per-speaker error rate. In applications where dialogue from each speaker is equally important, one might choose to optimise for JER and use 'score matrix SC'. In applications where only the dialogue from majority speakers are important, one might choose to optimise using 'standard SC'. This explains why the mean DER is higher on the development set than the test set when using system SAD (Figure 5.5) as the majority speakers are correctly labelled on the test set but not on the development set where speaker representations may be more balanced. The inverse is true for JER which indicates that certain speakers have been wrongly labelled.

As shown in Section 3.3, the SAIGEN corpus consists of two types of speakers, the agent and caller, which are both equally important for ASR but imbalanced in terms of representation. It is therefore better, when using the SAIGEN corpus, to adapt system performance in terms of JER. Table 5.5 shows the average relative improvement in JER (relative to the unadapted baseline) of the adapted systems. The SC interpolation methods shows a higher average relative improvement than the standard interpolation method, with the 'Score Matrix SC' showing the highest average performance. However, given the

large error margins of the 'Score Matrix SC' method, especially on the test set, it appears that this method is more effective for some calls than others. In fact, the 'Score Matrix SC' interpolation method showed no significant increase in diarisation performance over standard PLDA interpolation despite the lower mean error (corrected resampled Student's t-test, $p_{value} > 0.05$).

Table 5.5: Relative improvement over the unadapted baseline in terms of JER (higher is better) when using retrained RG and LDA transforms, standard PLDA interpolation (a single α used for all calls) and PLDA interpolation with SC for fine-tuning (a separate α estimated for each call). Results are reported using oracle and system SAD and the error margin is the standard error.

SAD	Adaptation	Relative JER improvement (%)	
		Development	Test
Oracle	Retrained transforms (no interpolation)	68.01 ± 1.13	43.99 ± 3.58
	Standard PLDA interpolation	69.40 ± 0.82	47.64 ± 3.45
	PLDA interpolation with standard SC	69.40 ± 1.12	46.74 ± 3.12
	PLDA interpolation with score matrix SC	**70.16 ± 1.00**	**48.15 ± 4.19**
System	Retrained transforms (no interpolation)	51.80 ± 0.44	54.55 ± 0.67
	Standard PLDA interpolation	52.61 ± 0.36	56.72 ± 0.94
	PLDA interpolation with standard SC	51.88 ± 0.43	56.87 ± 0.75
	PLDA interpolation with score matrix SC	**53.44 ± 0.48**	**56.89 ± 1.14**

Finally, we investigate the correlation between SC and the diarisation error rate of each call to determine how good of a proxy SC is for diarisation error by calculating the Pearson correlation between SC and diarisation error as we sweep across α. Table 5.6 shows the mean Pearson correlation coefficient between the SC calculated using 'score matrix SC' and the corresponding error rate for each call over all development and test splits[5]. The SC shows a negative correlation to both DER and JER which indicates that the SC increases as the diarisation error decreases. Notice that the SC shows a higher negative correlation to DER than JER indicating that, at least for 'score matrix SC' the SC is a better proxy for DER than JER.

Table 5.6: Mean Pearson correlation coefficient between the SC and diarisation of each call within each development and test split. The SC is calculated using the 'score matrix SC' method and the error margins are the standard error between splits.

SAD	Set	Pearson Correlation Coefficient (SC vs. Score Metric)	
		DER	JER
Oracle	Development	-0.88 ± 0.02	-0.71 ± 0.01
	Test	-0.86 ± 0.02	-0.62 ± 0.02
System	Development	-0.87 ± 0.01	-0.69 ± 0.01
	Test	-0.77 ± 0.05	-0.61 ± 0.04

Similarly, Figure 5.6 shows the probability distribution of the Pearson correlation coefficient between the SC (calculated using the 'score matrix SC' method) and diarisation error for each all calls within each split. As in Table 5.6, we see a negative correlation between SC and diarisation error.

[5]The Pearson correlation coefficient is calculated using the scipy python package available at: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html`

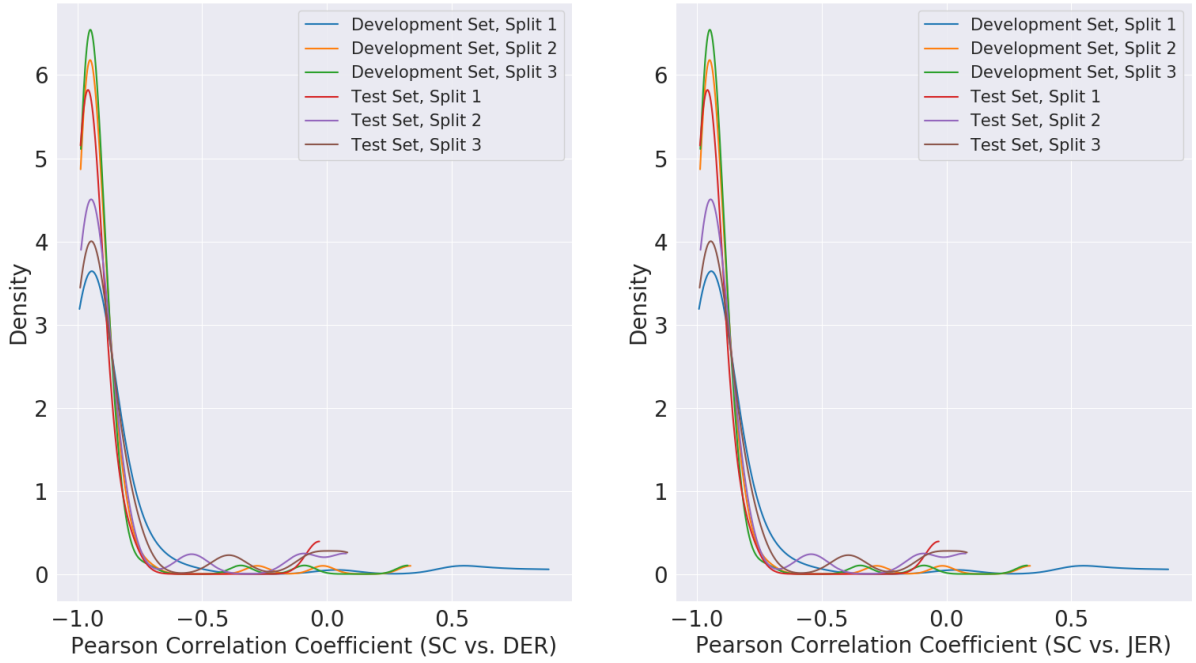

Figure 5.6: Probability density function showing the Pearson correlation coefficient between the SC and diarisation error in each call within all development and tests splits.

5.5 Effects of diarisation on a pre-trained ASR system

Speaker diarisation is often used as a pre-processing step in ASR as it provides speaker indexes which can then be used by ASR systems for speaker-specific adaptations. By adapting a speaker diarisation system to the target domain, the accuracy of the speaker indexes are improved which could improve the performance of speaker-specific adaptations. This section investigates the performance of a pre-trained ASR system on the SAIGEN corpus when using speaker indexes, provided by our adapted system(s), for speaker-specific adaptations.

Method

The pre-trained ASR system trained used in this section was trained on South African speech[6][7]. We provide the output of our speaker diarisation system as metadata to the ASR system with which to perform speaker-specific adaptations. The performance of the ASR system is measured in terms of word error rate (WER) and is calculated as follows:

$$WER = \frac{insertions + substitutions + deletions}{total} \times 100 \tag{5.5}$$

where insertions are words that have been incorrectly inserted or added, substitutions are words that have been changed to something else, and deletions are words that have been removed.

We evaluate the performance of the diarisation-aided speaker-specific adaptations when performing ASR in three ways: we first provide the system with oracle speaker information (oracle speaker info) which includes the oracle SAD marks and speaker tags, i.e., 'perfect' diarisation. We then provide speaker information obtained from performing diarisation with standard PLDA interpolation (per-corpus fine-tuning) and 'Score Matrix SC' (per-file fine-tuning) as defined in Section 5.4.2. Due to time constraints, we could not test both 'Standard SC' and 'Score Matrix SC', therefore, we chose to evaluate 'Score Matrix SC' as it showed similar performance to 'Standard SC' in terms of both DER and JER (Figures 5.4 and 5.5). When performing system SAD we manually remove audio segments containing music and ringing from the SAIGEN corpus as these noisy segments will result in poor performance on the part of the ASR system which would make it impossible to compare results.

[6]Kaldi recipe available at: `https://github.com/kaldi-asr/kaldi/tree/master/egs/wsj/s5`

[7]The ASR system is a proprietary system developed by SAIGEN. The exact details of the training data and model architecture are proprietary.

Results

Table 5.7 shows the mean WER taken over all development and test sets of the SAIGEN corpus. Additionally, The mean WER for each individual development and test set split can be found in Table A.11. Note that the use of the oracle speaker info for speaker-specific adaptations provides some benefit in terms of WER, as opposed to no speaker info, wherein the ASR system does not perform any speaker-specific adaptations. The 'Score Matrix SC' method of PLDA interpolation has similar performance to standard PLDA interpolation, despite having a lower DER and JER than PLDA interpolation across all splits.

Table 5.7: Average WER taken over all 3 splits of the SAIGEN corpus. The error margin is the standard error between splits.

SAD	Subset	No speaker info	Oracle speaker info	PLDA interpolation	PLDA interpolation with score matrix SC
Oracle	Development	51.26 ± 0.28	49.26 ± 0.25	49.70 ± 0.33	49.73 ± 0.30
	Test	53.73 ± 0.93	51.99 ± 0.86	52.83 ± 0.75	52.80 ± 1.27
System	Development	52.79 ± 0.30	-	50.99 ± 0.29	51.06 ± 0.29
	Test	55.49 ± 0.95	-	54.05 ± 0.94	54.13 ± 0.93

5.6 Conclusion

This chapter investigated the adaptation of a pre-trained speaker diarisation system in a real-world environment. First, a baseline system was established and its performance was compared over variations of the default system settings, namely segmentation and segment padding. Thereafter, various adaptation techniques are analysed and compared, specifically the retraining of statistical components and PLDA model interpolation. The use of silhouette coefficients for automatically fine-tuning the PLDA interpolation process on a per-file basis is also explored. Finally, the use of speaker-adaptations, based on the

speaker indexes provided by the adapted system, is investigated with a pre-trained ASR system.

Our main findings are as follows:

- The step size used in segmentation did not have a remarkable effect on baseline performance. Furthermore, when using system SAD, it is better to include segment padding as a post-processing step.
- In terms of domain adaptation, the largest immediate improvement is seen when using RG and LDA transforms that had been retrained on ID data. Additionally, although PLDA interpolation (Figures 5.2 and 5.3) has a lower mean diarisation error, when compared to retrained transforms, the result was not found to be statistically significant.
- By using the SC as an estimate of cluster quality, it is possible to approximate the optimal interpolation parameter α on a per-call basis and achieve comparable performance to the alternative which estimates a single α for all calls. It is also shown that the use of the PLDA score matrix for calculating the SC performs slightly better, in terms of mean JER, than using the cosine distance between embeddings (Figure 5.3). The inverse seems to be true for DER.
- As shown in Table 5.5, the use of the SC to fine-tune the PLDA interpolation process, especially the 'Score Matrix SC' method, provides the highest relative increase in diarisation performance over the unadapted baseline (in terms of mean JER). Furthermore, this method of domain adaptation is computationally inexpensive as the only added step is to calculate the SC, which could make it a practical solution for the adaption of speaker diarisation systems to low-resource domains, especially when the in-domain data differs in terms of recording conditions.
- By using the metadata provided by our adapted system(s) for speaker-specific adaptations in a pre-trained ASR system we see a slight, but consistent improvement in WER. We observe no remarkable difference in WER when using an adapted system

with or without the use of silhouette coefficients for PLDA interpolation. (Table 5.7)

Chapter 6

Conclusion

This chapter summarizes the key findings and contributions made in this study. Possibilities and recommendations for further work are also discussed.

6.1 Introduction

This study investigated ways in which a pre-trained speaker diarisation system can be adapted to an unseen, low-resource domain, specifically South African call centres. The main goal of the study was to determine whether domain adaptation techniques are effective in a challenging, real-world environment of practical importance. In this chapter, the key findings of this study are discussed and the original research objectives are revisited in light of these findings. Secondly, the practical implications of this work and future research are explored.

6.2 Key findings

This research investigated the adaptation of pre-trained speaker diarisation systems to low-resource domains, and was primarily focused on the adaptation of domain-specific components. It was shown that the performance of a pre-trained system on a new domain can be considerably improved by adapting the statistical components namely, the RG, LDA and PLDA transforms, without otherwise altering other components in the system, such as the x-vector or SAD models which require large amounts of training data. The key findings of this research are as follows:

On the use of pre-trained systems:

- When choosing a baseline system, we found it beneficial to follow the 'Occam's razor' approach: the E-TDNN x-vector system, which is the least complex system in terms of architecture and clustering, showed similar or better performance compared to its more complicated counterparts in preliminary testing on the EMRAI corpus. This is not to say that more complex architectures and clustering are not useful, but the E-TDNN system was adequate for the purposes of this research, as the focus was predominantly on the statistical components, which were similar in all the pre-trained models.

- Whilst configuring the E-TDNN baseline, which was identified as a suitable baseline due to its performance on the EMRAI corpus, we investigate the effect of the default system settings, namely step size and segment padding, on diarisation performance. The diarisation performance did not significantly change for different step sizes but segment padding proved to be beneficial for the TDNN-SAD system as it improved the missed detection and speaker confusion rates. Therefore, it is advantageous to investigate the effects of a pre-trained system's default settings prior to domain adaptation to ensure that the system is properly configured.

Domain adaptation through statistical modelling:

- The adaptation of statistical components require considerably less data than is typically required to train an x-vector system, which makes it a viable solution for domain adaptation in low-resource domains.
- The addition of a SAD component does not negate the improvements afforded by the adapted statistical components, although the SAD component decreases overall diarisation performance (when compared with oracle SAD segmentation, which is not available in practice).
- Although the SAIGEN corpus was considered to represent a single domain – South African call centres – the calls within the dataset contained slight domain mismatches, such as differences in the types of non-speech, languages and recording conditions. To address these slight mismatched conditions, the use of cluster analysis was introduced to fine-tune the domain adaptation process, specifically PLDA interpolation, on a per-call basis as it cannot be assumed that each call would necessarily have the same 'optimal' adaptation.
- It was shown that, by using the SC as a measure of cluster quality, the optimal interpolation parameter or adaptation can be estimated on each call without requiring a held-out set or human supervision. However, the difference between SC for fine-tuning PLDA interpolation and standard PLDA interpolation was not found to be statistically significant, even though it had a better average performance. A further investigation is needed to determine if the use of the SC to fine-tune the interpolation process is a practical solution in cases were the ID data is slightly mismatched.

6.3 Addressing research questions

We discuss and answer our research questions in light of the key findings in Section 6.2:

- How can sub-components of existing speaker diarisation systems be adapted to new domains without retraining the entire system?

  - We find that a pre-trained system can be adapted by focusing on domain-specific components, namely the RG and LDA transforms and the PLDA model.
  - The greatest immediate improvement, relative to the unadapted baseline, is seen when retraining the RG and LDA on ID data; the use of PLDA interpolation as an added step provides a slight additional improvement (Tables 5.4 and 5.5).
  - The PLDA interpolation could potentially be improved, in terms of mean diarisation error rate by using the SC to estimate the optimal interpolation parameter α for each individual file (same tables as above), although the performance is highly variable across calls and should be investigated further.
  - The performance of an adapted system can be inferred by examining the SC of speaker clusters: a SC coefficient alludes to a better system (Table 5.6 and Figure 5.6.

- Does the SAIGEN corpus contain sufficient data for the domain adaptation of a pre-trained system?
  - Performance increases can be obtained from a relatively small datasets (Table 3.2), compared to the amount of data used to train the x-vector system (VoxCeleb1 and VoxCeleb2, as described in Section 4.2).
  - The fact that PLDA interpolation makes use of an ID and OOD model makes it a suitable technique for low-resource environments, such as the SAIGEN corpus, as the OOD is trained on a very large corpus and the contribution of the ID model during the interpolation process can be controlled.
  - The SAIGEN corpus does, therefore, contain sufficient data to adapt the statistical components of a pre-trained system. The adaptation of the x-vector system itself on the SAIGEN corpus is not investigated in this study.

- Can an adapted speaker diarisation system be used to improve the performance of an ID downstream ASR system? If yes, then to what extent?

  - By adapting a pre-trained speaker diarisation system using only a small in-domain dataset, we were able to extract speaker-specific metadata that produced a small improvement in the performance of an ASR system on the same target data (Table 5.7).
  - The observed improvements were small relative to the improvements of the speaker diarisation system before and after adaptation. However, the use of speaker diarisation in ASR depends on the baseline performance of the ASR system.

6.4 Practical Implications

In this section, the practical use of this research is highlighted in the broader context of the speech technology field:

- Only a small ID dataset is needed to adapt the statistical components of a pre-trained speaker diarisation system to a point where it is beneficial for downstream applications and is therefore a practical approach for real-world systems where large ID datasets are not always available.
- Given limited time or computational resources, as is often the case for practical applications, it is recommended that systems are first adapted by only retraining the RG and LDA components as these are the least computationally expensive components to train and delivered the best immediate improvement. PLDA interpolation can then be added as an additional step as the computational resources required by this step are reliant on the size of the OOD dataset, which may require a larger training capacity (as was the case in this research).
- The use of SC for fine-tuning the PLDA interpolation process is useful when adapting a diarisation system to a new domain, especially when the target domain is sourced from similar, but slightly mismatched sub-domains – South African call centres in our case.

- The methods of domain adaptation for speaker diarisation systems investigated in this research can be used for improved speaker indexing in unlabelled, ID data. In practice, speaker indexing is used to correctly label audio transcriptions with speaker information, a frequent requirement of call centres that utilise speech recognition services.

- The adapted speaker diarisation systems could also be beneficial for downstream ASR systems that require speaker indexes for speaker-specific adaptations, depending on the specific ASR system and other environmental factors, such as the target data.

6.5 Future research

While we have conducted a study on the adaptation of statistical components for speaker diarisation systems, there are still several questions that have arisen through the course of this research and require further exploration:

- For all experiments, the statistical components were retrained on the entire development set. Although these techniques proved beneficial for both the EMRAI corpus (5 hours of training data) and the SAIGEN corpus (18 hours of training data), the point at which these techniques are no longer effective, from a data availability standpoint, should be investigated.

- Labelled ID data was used to retrain components. However, labelled ID data is not always available, and methods for unsupervised training have been proposed by Wang et al. [80], and Le Lan et al. [81].

- In this work, the SC is used as an indirect indication of diarisation performance. It would be worth investigating if the same method can be used to approximate the performance of a system on unlabelled data, thereby enabling diarisation systems to be adapted on unlabeled corpora.

- This research only investigated two default system settings: step size and segment padding. However, other settings such as segment size and LDA dimensionality should also be investigated.
- This research successfully applies the use of the SC for fine-tuning PLDA interpolation. It should be investigated if the SC can be used to automatically fine-tune other hyperparameters, such as the default system settings.

6.6 Conclusion

Domain mismatch is a complex and common problem in speaker diarisation and the broader field of speech recognition as a whole, especially in low-resource environments. Through this research, we have studied this problem and potential solutions and demonstrated that it is indeed possible to adapt a pre-trained speaker diarisation system to a new domain without intensive retraining.

Our results show that the statistical components of a speaker diarisation pipeline can be adapted to a target domain using a small, labelled ID dataset. We show that the diarisation performance of a pre-trained x-vector system can be improved by retrained the RG, LDA and PLDA models on ID data, and by interpolating the ID and OOD PLDA models. Furthermore, we show that by using the SC to analyse the quality of speaker clusters, it is possible to fine-tune the PLDA interpolation process in a per-file manner.

We hope that this research and the findings presented herein can aid others in designing and implementing speaker diarisation systems in challenging environments, and that the restrictions imposed by data availability may be a catalyst for innovation, as it certainly was in our case.

Bibliography

[1] X. Anguera Miró, *Robust Speaker Diarization for Meetings*. Universitat Politècnica de Catalunya, 2006.

[2] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker Diarization: A Review of Recent Research," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 356–370, 2012.

[3] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[4] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker Diarization Using Deep Neural Network Embeddings," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 4930–4934.

[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN Embeddings for Speaker Recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 5329–5333.

[6] Y. Konig, L. Heck, M. Weintraub, K. Sonmez, *et al.*, "Nonlinear Discriminant Feature Extraction for Robust Text-independent Speaker Recognition," Citeseer.

[7] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep Neural Networks for Small Footprint Text-Dependent Speaker Verification," in

2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 4052–4056.

[8] L. Li, Y. Lin, Z. Zhang, and D. Wang, "Improved Deep Speaker Feature Learning for Text-Dependent Speaker Recognition," in *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, IEEE, 2015, pp. 426–429.

[9] S. Ioffe, "Probabilistic Linear Discriminant Analysis," in *European Conference on Computer Vision*, Springer, 2006, pp. 531–542.

[10] D. Garcia-Romero, A. McCree, S. Shum, N. Brummer, and C. Vaquero, "Unsupervised Domain Adaptation for I-vector Speaker Recognition," in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, vol. 8, 2014.

[11] L. van Wyk, M. H. Davel, and C. J. Van Heerden, "Unsupervised Fine-Tuning of Speaker Diarisation Pipeline using Silhouette Coefficients," in *Southern African Conference for Artificial Intelligence Research*, 2021.

[12] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-End Neural Speaker Diarization with Self-Attention," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, IEEE, 2019, pp. 296–303.

[13] K. Kinoshita, M. Delcroix, and N. Tawara, "Integrating End-to-End Neural and Clustering-Based Diarization: Getting the Best of Both Worlds," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 7198–7202.

[14] K. S. Rao and K. Manjunath, *Speech Recognition using Articulatory and Excitation Source Features*. Springer, 2017, pp. 85–92.

[15] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech Enhancement Based on Deep Denoising Autoencoder.," in *Interspeech*, vol. 2013, 2013, pp. 436–440.

[16] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A Regression Approach to Speech Enhancement Based on Deep Neural Networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.

[17] T. Gao, J. Du, L.-R. Dai, and C.-H. Lee, "Densely Connected Progressive Learning for LSTM-Based Speech Enhancement," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 5054–5058.

[18] J. S. Garofolo, C. D. Laprun, and J. G. Fiscus, *The Rich Transcription 2004 Spring Meeting Recognition Evaluation*. Citeseer, 2006.

[19] M. Ravanelli and Y. Bengio, "Speaker Recognition From Raw Waveform with Sincnet," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, IEEE, 2018, pp. 1021–1028.

[20] T. Ng, B. Zhang, L. Nguyen, S. Matsoukas, X. Zhou, N. Mesgarani, K. Vesely, and P. Matějka, "Developing a Speech Activity Detection System for the DARPA RATS Program," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[21] T. Pfau, D. P. Ellis, and A. Stolcke, "Multispeaker Speech Activity Detection for the ICSI Meeting Recorder," in *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU'01.*, IEEE, 2001, pp. 107–110.

[22] N. Ryant, M. Liberman, and J. Yuan, "Speech Activity Detection on Youtube Using Deep Neural Networks.," in *INTERSPEECH*, Lyon, France, 2013, pp. 728–731.

[23] S. Chen, P. Gopalakrishnan, *et al.*, "Speaker, Environment and Channel Change Detection and Clustering Via the Bayesian Information Criterion," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Virginia, USA, vol. 8, 1998, pp. 127–132.

[24] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel, "Strategies for Automatic Segmentation of Audio Data," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (cat. no. 00ch37100)*, IEEE, vol. 3, 2000, pp. 1423–1426.

[25] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, "A Review of Speaker Diarization: Recent Advances with Deep Learning," *arXiv preprint arXiv:2101.09624*, 2021.

[26] A. F. Agarap, "Deep Learning Using Rectified Linear Units (RELU)," *arXiv preprint arXiv:1803.08375*, 2018.

[27] R. Hecht-Nielsen, "Theory of the Backpropagation Neural Network," in *Neural Networks for Perception*, Elsevier, 1992, pp. 65–93.

[28] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, 1988, pp. 21–28.

[29] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[30] B. Liu, W. Zhang, X. Xu, and D. Chen, "Time Delay Recurrent Neural Network for Speech Recognition," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1229, 2019, p. 012 078.

[31] V. Peddinti, D. Povey, and S. Khudanpur, "A Time Delay Neural Network Architecture for Efficient Modeling of Long Temporal Contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[32] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker Diarization with LSTM," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 5239–5243.

[33] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully Supervised Speaker Diarization," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 6301–6305.

[34] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj, *et al.*, "Chime-6 Challenge: Tackling Multispeaker Speech Recognition for Unsegmented Recordings," *arXiv preprint arXiv:2004.09249*, 2020.

[35] A. Nagrani, J. S. Chung, J. Huh, A. Brown, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, "Voxsrc 2020: The Second VoxCeleb Speaker Recognition Challenge," *arXiv preprint arXiv:2012.06867*, 2020.

[36] Google, *Multi-Class Neural Networks: Softmax*, `https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax`, Accessed: 23-09-2021.

[37] P. Kenny, "Bayesian Speaker Verification with Heavy-Tailed Priors.," in *Odyssey*, vol. 14, 2010.

[38] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support Vector Machines Versus Fast Scoring in the Low-Dimensional Total Variability Space for Speaker Verification," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[39] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of I-vector Length Normalization in Speaker Recognition Systems," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[40] S. Lyu and E. P. Simoncelli, "Nonlinear Extraction of Independent Components of Natural Images Using Radial Gaussianization," *Neural Computation*, vol. 21, no. 6, pp. 1485–1519, 2009.

[41] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep Neural Network-based Speaker Embeddings for End-to-End Speaker Verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, IEEE, 2016, pp. 165–170.

[42] C. M. Bishop, "Pattern Recognition and Machine Learning," *Machine Learning*, vol. 128, no. 9, 2006.

[43] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher Discriminant Analysis with Kernels," in *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (cat. no. 98th8468)*, IEEE, 1999, pp. 41–48.

[44] T. Hastie and R. Tibshirani, "Discriminant Analysis by Gaussian Mixtures," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 155–176, 1996.

[45] P. Singh, *Probabilistic linear discriminant analysis (PLDA) explained*, Nov. 2020. [Online]. Available: `https://towardsdatascience.com/probabilistic-linear-discriminant-analysis-plda-explained-253b5effb96`.

[46] M. Diez, L. Burget, F. Landini, S. Wang, and H. Černocky, "Optimizing Bayesian HMM Based X-vector Clustering for the Second DIHARD Speech Diarization Challenge," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 6519–6523.

[47] D. Garcia-Romero and A. McCree, "Supervised Domain Adaptation for I-vector Based Speaker Recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 4047–4051.

[48] F. Landini, S. Wang, M. Diez, L. Burget, P. Matejka, K. molkova, L. Mosner, A. Silnova, O. Plchot, O. Novotny, *et al.*, "BUT System for the Second DIHARD Speech Diarization Challenge," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 6529–6533.

[49] W. H. Day and H. Edelsbrunner, "Efficient Algorithms for Agglomerative Hierarchical Clustering Methods," *Journal of Classification*, vol. 1, no. 1, pp. 7–24, 1984.

[50] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[51] M. Diez, L. Burget, and P. Matejka, "Speaker Diarization Based on Bayesian HMM with Eigenvoice Priors.," in *Odyssey*, 2018, pp. 147–154.

[52] N. Ryant, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "Third DIHARD Challenge Evaluation Plan," *arXiv preprint arXiv:2006.05815*, 2020.

[53] M. Gales and S. Young, "The Application of Hidden Markov Models in Speech Recognition," 2008.

[54] M. Diez, L. Burget, S. Wang, J. Rohdin, and J. Cernocky, "Bayesian HMM Based X-vector Clustering for Speaker Diarization.," in *INTERSPEECH*, 2019, pp. 346–350.

[55] J. S. Garofolo, C. D. Laprun, and J. G. Fiscus, *The Rich Transcription 2004 Spring Meeting Recognition Evaluation*. Citeseer, 2006.

[56] J. G. Fiscus, N. Radde, J. S. Garofolo, A. Le, J. Ajot, and C. Laprun, "The Rich Transcription 2005 Spring Meeting Recognition Evaluation," in *International Workshop on Machine Learning for Multimodal Interaction*, Springer, 2005, pp. 369–389.

[57] J. G. Fiscus, J. Ajot, M. Michel, and J. S. Garofolo, "The Rich Transcription 2006 Spring Meeting Recognition Evaluation," in *International Workshop on Machine Learning for Multimodal Interaction*, Springer, 2006, pp. 309–322.

[58] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "First DIHARD Challenge Evaluation Plan," *2018, tech. Rep.*, 2018.

[59] ——, "The Second DIHARD Diarization Challenge: Dataset, Task and Baselines," *arXiv preprint arXiv:1906.07839*, 2019.

[60] H. Bredin, "PyAnnote Metrics: A Toolkit for Reproducible Evaluation, Diagnostic, and Error Analysis of Speaker Diarization Systems.," in *INTERSPEECH*, 2017, pp. 3587–3591.

[61] T. P. Trappenberg, "Machine Learning with sklearn," in *Fundamentals of Machine Learning*, Oxford University Press, 2019, pp. 38–65.

[62] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The Kaldi Speech Recognition Toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, 2011.

[63] E. Edwards, M. Brenndoerfer, A. Robinson, N. Sadoughi, G. P. Finley, M. Korenevsky, N. Axtmann, M. Miller, and D. Suendermann-Oeft, "A Free Synthetic Corpus for Speaker Diarization Research," in *International Conference on Speech and Computer*, Springer, 2018, pp. 113–122.

[64] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR Corpus Based on Public Domain Audio Books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 5206–5210.

[65] D. Snyder, G. Chen, and D. Povey, "Musan: A Music, Speech, and Noise Corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[66] J. Van Meggelen, R. Bryant, and L. Madsen, *Asterisk: The Definitive Guide: Open Source Telephony for the Enterprise*. O'Reilly Media, 2019.

[67] Q. Lin, W. Cai, L. Yang, J. Wang, J. Zhang, and M. Li, "DIHARD II is Still Hard: Experimental Results and Discussions from the DKU-LENOVO Team," *arXiv preprint arXiv:2002.12761*, 2020.

[68] M. Lavechin, M.-P. Gill, R. Bousbib, H. Bredin, and L. P. Garcia-Perera, "End-to-End Domain-Adversarial Voice Activity Detection," 2020. [Online]. Available: `https://arxiv.org/abs/1910.10655`.

[69] J. Profant, *Robust Speaker Verification with Deep Neural Networks*, 2019. [Online]. Available: `https://www.vutbr.cz/en/students/final-thesis/detail/122072`.

[70] F. Landini, J. Profant, M. Diez, and L. Burget, "Bayesian HMM Clustering of X-vector Sequences (VBx) in Speaker Diarization: Theory, Implementation and Analysis on Standard Tasks," *Computer Speech & Language*, vol. 71, p. 101 254, 2022.

[71] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A Large-Scale Speaker Identification Dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[72] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep Speaker Recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[74] Y. Fan, J. Kang, L. Li, K. Li, H. Chen, S. Cheng, P. Zhang, Z. Zhou, Y. Cai, and D. Wang, "CN-Celeb: A Challenging Chinese Speaker Recognition Dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 7604–7608.

[75] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive Angular Margin Loss for Deep Face Recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[76] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, pp. 239–281, 2003.

[77] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[78] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[79] P. J. Rousseeuw, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[80] Q. Wang, K. Okabe, K. A. Lee, and T. Koshinaka, "A Generalized Framework for Domain Adaptation of PLDA in Speaker Recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 6619–6623.

[81] G. Le Lan, D. Charlet, A. Larcher, and S. Meignier, "An Adaptive Method for Cross-Recording Speaker Diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1821–1832, 2018.

Appendix A

Supplemental content

This appendix contains supplemental content, such as tables and figures which support the results and findings present in the main text.

A.1 Appendix: Chapter 4

Table A.1: Testing range for each hyperparameter used during the evaluation of VBHMM clustering

Hyperparameter	Testing Range
P_{loop}	0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
F_A	10,20,30,40,50,60,70,80,90
F_B	0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

Table A.2: Mean diarisation error taken over all noise levels of the EMRAI 'dev-other' subset using a retrained RG and LDA transform trained on all 6 noise levels. Results are reported using Oracle SAD.

Noise Level	E-TDNN		BUT DIHARD		BUT ResNet101	
	DER (%)	JER (%)	DER (%)	JER (%)	DER (%)	JER (%)
no noise	2.85 ± 1.03	3.69 ± 1.87	3.61 ± 0.88	4.25 ± 1.46	3.69 ± 2.17	4.42 ± 3.75
5dB	3.72 ± 1.51	5.44 ± 2.90	4.36 ± 1.76	5.70 ± 3.14	6.08 ± 6.36	8.64 ± 9.49
3dB	3.94 ± 1.67	5.86 ± 3.09	4.50 ± 1.51	6.02 ± 2.82	8.01 ± 9.10	11.72 ± 13.19
0dB	4.60 ± 1.85	7.12 ± 3.74	6.10 ± 6.43	8.75 ± 10.24	13.56 ± 13.78	20.43 ± 20.33
-3dB	6.13 ± 3.15	9.95 ± 5.62	8.38 ± 9.04	12.42 ± 13.39	17.65 ± 15.08	26.79 ± 21.84
-5dB	8.26 ± 4.40	13.79 ± 7.68	15.62 ± 14.86	23.72 ± 21.88	25.10 ± 16.76	36.62 ± 22.94

Table A.3: Mean diarisation error taken over all noise levels of the EMRAI 'dev-other' subset using retrained RG and LDA transforms which had been separately trained on each noise level. Results are reported using Oracle SAD.

Noise Level	E-TDNN		BUT DIHARD		BUT ResNet101	
	DER (%)	JER (%)	DER (%)	JER (%)	DER (%)	JER (%)
no noise	2.63 ± 0.82	3.24 ± 1.44	3.62 ± 0.90	4.25 ± 1.51	3.75 ± 1.99	4.56 ± 3.56
5dB	3.48 ± 1.51	4.95 ± 2.89	4.39 ± 1.87	5.75 ± 3.27	6.16 ± 6.58	8.75 ± 9.74
3dB	3.74 ± 1.60	5.47 ± 2.97	4.73 ± 1.81	6.46 ± 3.29	8.72 ± 9.31	13.04 ± 14.26
0dB	4.31 ± 1.86	6.55 ± 3.56	6.03 ± 6.82	8.50 ± 10.59	14.01 ± 13.73	21.20 ± 20.10
-3dB	6.28 ± 5.52	10.07 ± 9.26	8.54 ± 9.14	12.93 ± 14.66	17.39 ± 14.46	26.60 ± 21.25
-5dB	7.02 ± 3.87	11.57 ± 6.85	17.74 ± 16.31	26.95 ± 24.39	23.99 ± 16.99	35.47 ± 23.72

Table A.4: Mean diarisation error taken over all noise levels of the EMRAI 'dev-other' subset with PLDA interpolation trained on all 6 noise levels. Results are reported using oracle SAD.

Noise Level	E-TDNN		BUT DIHARD		BUT ResNet101	
	DER (%)	JER (%)	DER (%)	JER (%)	DER (%)	JER (%)
no noise	2.76 ± 0.10	3.52 ± 1.81	3.45 ± 0.75	3.92 ± 1.13	3.30 ± 0.67	3.71 ± 0.97
5dB	3.52 ± 1.41	5.06 ± 2.72	3.88 ± 1.13	4.79 ± 2.03	4.00 ± 3.27	5.10 ± 2.44
3dB	3.67 ± 1.56	5.33 ± 2.87	4.09 ± 1.24	5.21 ± 2.22	5.31 ± 3.74	7.45 ± 6.40
0dB	4.23 ± 1.67	6.44 ± 3.28	4.35 ± 1.38	5.72 ± 2.50	6.87 ± 4.49	10.36 ± 7.91
-3dB	5.40 ± 2.29	8.64 ± 4.31	5.11 ± 2.32	7.13 ± 4.23	9.24 ± 6.48	14.21 ± 10.49
-5dB	7.03 ± 3.36	11.69 ± 6.12	6.24 ± 3.34	9.32 ± 6.10	12.05 ± 10.03	18.62 ± 14.09

Table A.5: Mean diarisation error taken over all noise levels of the EMRAI 'dev-other' subset with PLDA interpolation using separately trained RG and LDA transforms and PLDA models for each noise level. Results are reported using oracle SAD.

Noise Level	E-TDNN		BUT DIHARD		BUT ResNet101	
	DER (%)	JER (%)	DER (%)	JER (%)	DER (%)	JER (%)
no noise	2.51 ± 7.51	3.01 ± 1.27	3.49 ± 0.76	4.00 ± 1.23	3.80 ± 1.29	4.73 ± 2.30
5dB	3.22 ± 1.49	4.44 ± 2.84	3.90 ± 1.08	4.82 ± 1.93	4.67 ± 2.03	6.43 ± 3.82
3dB	3.32 ± 1.41	4.63 ± 2.63	4.06 ± 1.25	5.15 ± 2.21	5.54 ± 3.58	7.93 ± 6.24
0dB	3.91 ± 1.51	5.80 ± 2.97	4.44 ± 1.30	5.91 ± 2.42	6.97 ± 3.73	10.61 ± 6.54
-3dB	4.91 ± 2.43	7.68 ± 4.57	5.47 ± 2.90	7.79 ± 5.14	8.67 ± 5.75	13.31 ± 9.40
-5dB	6.73 ± 6.77	10.63 ± 9.45	7.15 ± 6.18	10.82 ± 10.02	12.10 ± 7.37	19.10 ± 11.60

A.2 Appendix: Chapter 5

Table A.6: The mean segment coverage, purity and boundary precision and recall rates of the TDNN-SAD on the SAIGEN corpus, with and without segment padding. A 250ms collar is used during scoring.

Post-processing	Coverage	Purity	Boundary Precision	Boundary Recall
With Padding	**90.43**	70.24	**84.53**	56.19
Without Padding	89.37	**84.29**	78.79	**71.56**

Table A.7: Mean DER with and without domain adaptation taken over all development and test splits of the SAIGEN corpus. Results are reported using oracle SAD and the error margin is the standard error between splits.

Adaptation	DER (%) (System SAD)	
	Development	Test
Baseline (no adaptation)	17.00 ± 0.28	17.20 ± 1.16
Retrained Transforms (no interpolation)	4.33 ± 0.16	10.09 ± 0.43
Standard PLDA Interpolation (per-corpus fine-tuning)	4.18 ± 0.13	9.83 ± 0.12
PLDA Interpolation \w Standard SC (per-file fine-tuning)	3.89 ± 0.17	10.00 ± 0.41
PLDA Interpolation \w Score Matrix SC (per-file fine-tuning)	3.98 ± 0.21	9.54 ± 0.48

Table A.8: Mean JER with and without domain adaptation taken over all development and test splits of the SAIGEN corpus. Results are reported using oracle SAD and the error margin is the standard error between splits.

Adaptation	JER (%) (Oracle SAD)	
	Development	Test
Baseline (no adaptation)	32.80 ± 0.47	33.66 ± 1.78
Retrained Transforms (no interpolation)	10.48 ± 0.22	18.74 ± 0.62
Standard PLDA Interpolation (per-corpus fine-tuning)	10.03 ± 12.40	17.50 ± 0.27
PLDA Interpolation \w Standard SC (per-file fine-tuning)	9.78 ± 0.19	17.83 ± 0.49
PLDA Interpolation \w Score Matrix SC (per-file fine-tuning)	9.86 ± 0.22	17.30 ± 0.60

Table A.9: Mean DER with and without domain adaptation taken over all development and test splits of the SAIGEN corpus. Results are reported using system SAD and the error margin is the standard error between splits.

Adaptation	DER (%) (Oracle SAD)	
	Development	Test
Baseline (no adaptation)	23.99 ± 0.35	28.38 ± 0.31
Retrained Transforms (no interpolation)	16.21 ± 0.13	15.10 ± 0.45
Standard PLDA Interpolation (per-corpus fine-tuning)	16.44 ± 0.13	14.90 ± 0.41
PLDA Interpolation \w Standard SC (per-file fine-tuning)	16.20 ± 0.12	14.52 ± 0.37
PLDA Interpolation \w Score Matrix SC (per-file fine-tuning)	15.80 ± 14.05	14.55 ± 0.49

Table A.10: Mean JER with and without domain adaptation taken over all development and test splits of the SAIGEN corpus. Results are reported using system SAD and the error margin is the standard error between splits.

Adaptation	JER (%) (System SAD)	
	Development	Test
Baseline (no adaptation)	45.84 ± 0.46	49.80 ± 1.40
Retrained Transforms (no interpolation)	22.09 ± 0.06	23.10 ± 0.17
Standard PLDA Interpolation (per-corpus fine-tuning)	21.72 ± 0.06	22.05 ± 0.20
PLDA Interpolation \w Standard SC (per-file fine-tuning)	22.05 ± 0.03	21.96 ± 0.15
PLDA Interpolation \w Score Matrix SC (per-file fine-tuning)	21.34 ± 0.02	21.94 ± 0.06

Table A.11: Average WER over for all development and test splits of the SAIGEN corpus. Results are reported with and without speaker specific adaptations. The error rate is the standard deviation between calls within the indicated split.

SAD	Subset		No speaker info	Oracle speaker info	PLDA Interpolation	PLDA Interpolation with Score Matrix SC
Oracle	Development	Split 1	51.44 ± 22.35	49.32 ± 22.60	49.89 ± 22.54	49.94 ± 22.62
		Split 2	50.70 ± 21.60	48.80 ± 21.95	49.07 ± 22.05	49.13 ± 21.87
		Split 3	51.63 ± 21.93	49.67 ± 22.03	50.14 ± 21.95	50.11 ± 22.03
	Test	Split 1	53.43 ± 20.96	52.32 ± 20.88	52.48 ± 21.01	52.39 ± 20.92
		Split 2	55.47 ± 23.52	53.34 ± 23.34	54.27 ± 22.85	55.18 ± 22.90
		Split 3	52.29 ± 22.82	50.41 ± 23.41	51.75 ± 23.16	50.83 ± 23.30
System	Development	Split 1	53.08 ± 22.86	-	51.23 ± 23.28	51.27 ± 23.16
		Split 2	52.18 ± 21.88	-	50.40 ± 22.29	50.48 ± 22.21
		Split 3	53.09 ± 22.29	-	51.32 ± 22.61	51.42 ± 22.53
	Test	Split 1	55.30 ± 21.06	-	53.54 ± 21.23	53.79 ± 21.19
		Split 2	57.81 ± 24.37	-	55.87 ± 24.72	55.88 ± 24.53
		Split 3	54.72 ± 23.55	-	52.73 ± 24.16	52.70 ± 23.97

Table A.12: Values for all one-sided corrected resampled Student's t-tests in Sections 5.4.1 and 5.4.2. The tests are conducted using DER and oracle SAD.

Null Hypothesis (Rows)/ Alternate Hypothesis (Columns)	Retrained Transform	PLDA Interpolation	Standard SC	Score Matrix SC
Baseline	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
Retrained Transform	-	0.43	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
PLDA Interpolation	-	-	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
Standard SC	-	-	-	0.14

Table A.13: P-values for all one-sided corrected resampled Student's t-tests in Sections 5.4.1 and 5.4.2. The tests are conducted using DER and system SAD.

Null Hypothesis (Rows)/ Alternate Hypothesis (Columns)	Retrained Transform	PLDA Interpolation	Standard SC	Score Matrix SC
Baseline	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
Retrained Transform	-	0.49	0.40	0.10
PLDA Interpolation	-	-	0.45	0.24
Standard SC	-	-	-	$< 5.00 \times 10^{-2}$

Table A.14: P-values for all one-sided corrected resampled Student's t-tests in Sections 5.4.1 and 5.4.2. The tests are conducted using JER and system SAD.

Null Hypothesis (Rows)/ Alternate Hypothesis (Columns)	Retrained Transform	PLDA Interpolation	Standard SC	Score Matrix SC
Baseline	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
Retrained Transform	-	0.22	0.43	$< 5.00 \times 10^{-2}$
PLDA Interpolation	-	-	0.21	$< 5.00 \times 10^{-2}$
Standard SC	-	-	-	0.27

Table A.15: P-values for all one-sided corrected resampled Student's t-tests in Sections 5.4.1 and 5.4.2. The tests are conducted using JER and system SAD.

Null Hypothesis (Rows)/ Alternate Hypothesis (Columns)	Retrained Transform	PLDA Interpolation	Standard SC	Score Matrix SC
Baseline	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
Retrained Transform	-	0.12	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
PLDA Interpolation	-	-	$< 5.00 \times 10^{-2}$	$< 5.00 \times 10^{-2}$
Standard SC	-	-	-	0.15