University of Sheffield

Speaker Diarization System for the DIHARD Challenge

Mangesh Hemant Hambarde

Supervisor: Dr. Thomas Hain

A report submitted in fulfilment of the requirements
for the degree of MSc Computer Science with Speech and Language
Processing

in the

Department of Computer Science

September 11, 2019

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Mangesh Hemant Hambarde

Signature: Mangesh Hemant Hambarde

Date: 11/09/2019

Abstract

Speaker Diarization is commonly known as the task of finding out "who spoken when?" in an audio recording. It is an important field because it is a crucial preprocessing step for many other areas in speech technology. One of the key challenges faced in this field is how to deal with domain variation. There is a lot of speaker, channel and environment variability that exists in the speech signal. Most previous diarization research has focused on specific domains and performance on diverse datasets is expected to be poor. DIHARD is a challenge that has been created by the diarization research community to boost research in the area of diverse datasets. The main aim of the project is to build a complete speaker diarization system that works within the rules of the 2019 DIHARD challenge using the Kaldi toolkit. The system is not officially submitted to the challenge because the dissertation timeline did not allow it. Several different system configurations are explored in the project. The best system involved concatenating two different speaker embeddings into a single embedding. This resulted in a diarization error rate (DER) of 24.64% on the evaluation set, almost a 7.3% relative improvement compared to the baseline which was at 26.58%.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Speaker Diarization

Speaker diarization is commonly known as the task of finding out “who spoke when?” in an audio recording with an unknown amount of speakers. It aims to split the recording into segments according to their speaker identity. These segments can also be overlapping. It acts as an important upstream preprocessing step for several tasks in speech processing. For example, it can be used in automatic transcription services to find all spoken segments and the speaker identities for them. The segments can then be passed to an automatic speech recognition (ASR) system to recognize the words. Furthermore, this also allows speaker-adapted ASR models to be used in order to improve the accuracy of ASR.

With increase in computing power, speech processing technologies have achieved incredible advances in the past decade that were not possible earlier. This has increased interest in automatic transcription technologies that can be used to automatically index the enormous amount of audio and video information that is generated in the modern world. This allows the creation of search engines that search audio files for information, just like text documents. Examples of queries can be: “which speakers tend to dominate a conversation?”, “which speakers are most likely to interrupt others?” and “fetch all segments spoken by a particular speaker”. Since speaker diarization is an important part in any automatic transcription system, there is a great deal of research interest in the area.

Diarization is not an easy problem since the output is affected by several factors like the application domain (broadcast news, meetings, telephone audio, internet audio, restaurant speech, clinical recordings etc), types and quality of microphones used (boom, lapel, far-field), inter-channel synchronization problems, overlapping speech, etc. Recently most of the research has focused on the meeting speech domain, since most problems that exist in speech recognition are encountered in this domain. The meeting scenario is thus often termed as "speech recognition complete". But focusing on only one domain can lead to problems. It makes it hard to compare diarization systems that are trained for different domains. In the worst case it also causes overfitting to the domain that they are trained on.

The DIHARD challenge [2] was created to establish standard datasets and performance baselines for diarization that have a good amount of domain variability. Systems that are trained only for a single domain are expected to perform poorly on these datasets. This is where the word "hard" comes from, besides of course, *Die Hard*. The DIHARD datasets span several domains of speech like broadcast, meeting, telephone, restaurant, courtroom, YouTube speech etc.

1.2 Motivation and Objectives

Creating a diarization system for the DIHARD challenge can be a rewarding experience since it gives a chance to learn about state-of-the-art speaker diarization techniques. Thus the main aim of the project is to build such a system using the Kaldi toolkit [1] that works within the rules of the 2019 DIHARD challenge. Systems from last year's challenge (2018) can be used as a reference. The focus is to explore different possible configurations to get the best performing system.

1.3 Report Outline

This chapter gave a brief introduction to speaker diarization and the aims of the project. Chapter 2 covers more detail on the various steps of a speaker diarization system and the Kaldi toolkit. Chapter 3 describes the structure and rules of the DIHARD challenge and the composition of the datasets involved. Chapter 4 explains how the baseline system supplied with the 2019 DIHARD challenge works. Chapter 5 describes different experiments that were performed throughout the project along with results. Chapter 6 concludes the report.

Chapter 2

Stages of Speaker Diarization

This chapter discusses the general stages of a speaker diarization system and some of the new techniques that are commonly used these days. A good overview of the diarization field, although not up to date, can be found in [3] and [4]. In all diarization systems, the basic approach is to first extract the segments containing speech from the audio, further divide the segments into subsegments in which the speaker identities are constant, represent each subsegment using a speaker embedding, and cluster the subsegments together so that each cluster represents a unique speaker.

2.1 Speech Enhancement

Any speech system needs to address the problem of environmental noise in audio recordings. It is important to remove as much noise as possible, but also speaker information loss should be kept to a minimum. If done right, this stage results in increased performance in subsequent stages. If not, artefacts in denoised speech reduce diarization performance. In recent years, deep learning methods have partially solved the problem of artefacts, but generalization ability in varied domains is still a problem. This stage is not mandatory, but can be helpful in certain conditions.

2.2 Feature Extraction

The first important step in any speech system is to represent the speech recording using a sequence of feature vectors. This representation is much more compact as it only uses a few thousand parameters for every second of audio, compared to 44,100 samples in a raw waveform sampled at 44.1 KHz. The most commonly used features are Mel Frequency Cepstral Coefficients (MFCCs) [5]. A brief overview of MFCCs is as follows.

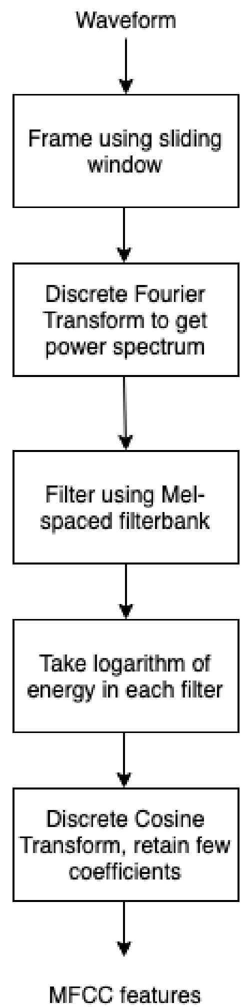

Figure 2.1: Stages of MFCC computation.

Speech production can be thought of as filtering of the sound produced from the vocal cords by the shape of the human vocal tract. The shape is influenced by the positions of the tongue, teeth, lips, velum etc and determines what sound comes out. Since the shape of the vocal tract is manifested in the envelope of the short time power spectrum of the speech signal, the job of the MFCCs can be thought of as to represent this envelope accurately.

As a first step, overlapping frames are extracted from the digitized signal using a sliding window. The properties of the signal are assumed to be stationary in this small window duration. Discrete Fourier Transform (DFT) is applied on each frame to get the periodogram of the frame. This tells us which frequencies are present in the frame. We need this because the cochlea in the human ear vibrates at different spots depending on the frequency composition of the sounds.

This still has too much information and we need to get rid of unnecessary information. The properties of the cochlea can be exploited for this. The cochlea cannot discern the difference between closely spaced frequencies, and this effect gets stronger as the frequencies increase. So to simulate this we can take energies from bins placed on the periodogram at increasing distances. The placement of the bins can be computed by using the Mel Scale [5], which is a perceptual scale of pitches judged by listeners to be equal in distance from one another. It is a logarithmic scale given by the below formula, which models the perceptual behaviour of the cochlea. It converts f Hz to m Mels. The bins are placed equidistant in Mel-domain, so that they are at an increasing distance in Hz-domain. The output of this operation can be interpreted as filtering the periodogram using a triangular filterbank called Mel-filterbank.

$$m = 2595 \log \left(1 + \frac{f}{700}\right)$$

After filtering and summing the values in say 26 bins, we get 26 energies. Logarithm is applied to each number because loudness is not perceived on a linear scale. Finally, Discrete Fourier Transform (DCT) is applied to these energies which decorrelates the energies, which allows modelling the features using diagonal covariance matrices. The first 12-13 coefficients are kept, rest are discarded. This is the MFCC feature vector for this frame. In cases decorrelated features are not necessary, for example neural networks, the DCT step is avoided and the features are called Mel-Filterbank features or FBANK features.

In most cases the MFCCs are also appended with delta and delta-delta coefficients, which are basically the first and second order derivatives calculated from the original coefficients using a sliding window context. These capture the acceleration trends in the coefficients on top of just their absolute values, and generally increase performance of speech systems.

2.3 Speech Activity Detection

Speech activity detection (SAD) extracts the segments in the audio that contain speech, and discards the rest. This is important because speaker diarization is only concerned with assigning speaker identities to speech segments, and does not need to do anything with non-speech segments. It is also possible to consider all the non-speech segments to be coming from a hypothetical new speaker which gets its own cluster after clustering, but that does not result in good performance as the amount of variability possible in non-speech sounds is too high. Therefore, doing a separate SAD step before diarization is the standard method.

There are two goals of a good SAD system - keep missed speech to a minimum, and keep false alarm speech to a minimum. The first might cause too few speakers to be detected, while the second pollutes clusters and degrades the diarization output. If the diarization system is used as a frontend for ASR, these errors cause word deletion and word insertion errors respectively.

Since this is a binary classification problem, it can be solved by using simple thresholding on the frame energy level. The MFCC features have the log energy of the frame as the first coefficient which can be used for this. Statistical model-based approaches are much more popular and are trained with large amounts of of diverse external data. Typically Long Short Term Memory networks (LSTMs) or Time Delay Neural Networks (TDNNs) are used for this task.

2.4 Segmentation

The goal of the segmentation task is to further divide the speech segments found after the SAD step into smaller subsegments such that there are no speaker turns within any of these subsegments. Speaker turns are defined to be the points in the audio where the set of talking speakers changes. For example within a speech segment, the point where spkr1 changes into spkr2 (spkr1 finished talking and spkr2 started immediately) would be a speaker turn because the set of talking speakers changes from [spkr1] to [spkr2]. Similarly, if spkr2 starts talking without waiting for spkr1 to end (causing an overlap) the set changes from [spkr1] to [spkr1,spkr2], which is also a speaker turn.

There are basically two ways to do segmentation. The first way is to try and automatically detect speaker turns and divide the segments at these points. The classical approach for doing this uses a sliding window, and compares consecutive windows. The comparison decides whether the two windows are better accounted by two separate models (different speaker sets) or single model (same speaker set) using an empirically determined threshold. Many distance metrics exist for this decision, for example the Delta Bayesian Information Criterion (ΔBIC) [6] metric. The second way is to divide the segments uniformly into very small subsegments (1-2 seconds) so that it is unlikely that the set of speakers changes within that segment, and assume that it is constant. It is not clear which of these ways yields better results. Uniform segmentation approaches are reported to work better for x-vectors, which are state-of-the-art [7].

2.5 Clustering

As the most important step of the diarization process, clustering works on the whole audio recording and groups together segments that belong to the same speaker. In the ideal case, all the segments belonging to a speaker exist in the same cluster, and the number of clusters is equal to the actual number of speakers. The clustering process needs a distance-like similarity measure between pairs of segments to work. Each segment can be represented by a point in vector space or a statistical model. This acts as the speaker representation for the segment, where the distance between any two representations is lower if the segments have speech from the same speaker.

Since the speaker verification and identification fields also use speaker models to capture speaker information, these models are adopted for clustering for diarization. The some of the best performing models are outlined below.

2.5.1 Speaker Representation

Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are generative models that can be used for modelling multivariate data. In GMMs, the probability of a data point is given by the weighted combination of the probabilities from several multivariate Gaussian distributions having their own mean and covariance matrices.

The goal is to train a GMM to represent each segment. For this, the feature vectors belonging to a segment can simply be pooled together and a GMM can be learned using the Expectation-Maximization (EM) [8] algorithm. But this is a problem because the number of feature vectors available from the segment would likely be insufficient to obtain a good estimate of the GMM parameters. This is because the number of GMMs is usually in the order of 512, 1024 or even 2048. To overcome this problem, a Universal Background Model (UBM) is trained using a large amount speech from the general population. This UBM is later adapted to each target segment using a Maximum Apriori (MAP) adaptation [9], resulting in an adapted GMM for each segment.

Now that we have a GMM to represent each segment, we can use different statistical similarity measures that can act as a distance metric that can be used for clustering. The Kullback-Leibler (KL) divergence [10] is a measure that estimates the distance between two random distributions. The Cross Likelihood Ratio (CLR) [11] is another measure and is given by the following.

$$CLR(S1, S2) = \log \frac{P(S1|M1)}{P(S1|M2)} + \log \frac{P(S2|M1)}{P(S2|M2)}$$

Where S_1 and S_2 are the segments that are being compared, and M_1 and M_2 are their corresponding GMMs. It can be seen that if the segments come from the same speaker, both denominators are high, and the distance decreases.

Later experiments found that only the means in the adapted GMMs carry most of the useful speaker information, the mixture weights and covariance matrices have too much variability to be of much use [12]. Hence the means of GMMs were concatenated into a single vector called a GMM supervector and simpler distance measures like cosine distance and Mahalanobis distance [13] were used as distance metrics for clustering.

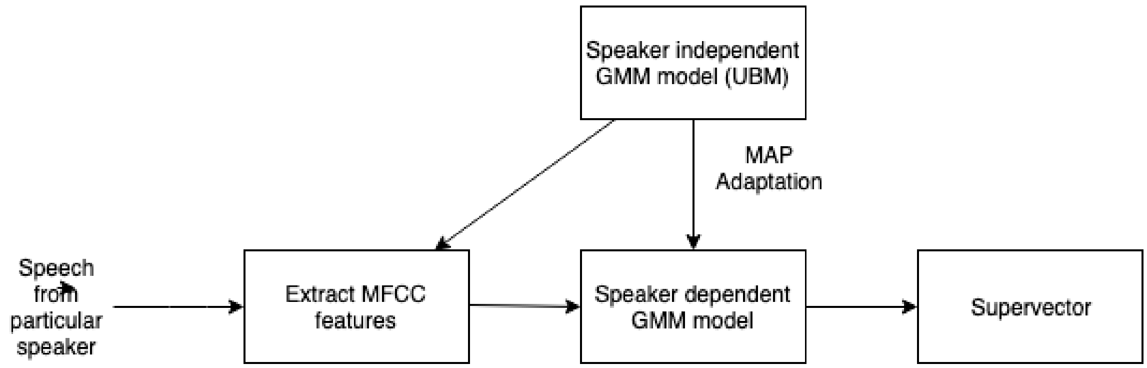

Figure 2.2: GMM adaptation.

I-vectors

I-vectors were introduced in [14] as a reduced dimension representation of the GMM supervector using joint factor analysis [15].

$$m_s = m_u + Tw_s$$

m_s and m_u are the adapted supervector for a segment S and the UBM supervector, respectively. w_s is the i-vector of the segment s. T is the "total variability matrix" which projects the supervector down to the i-vector representation. T is estimated from the training data using the EM algorithm. I-vectors are generally chosen to be about 400 dimensions.

I-vectors can be compared with simple cosine scoring to judge whether they represent the same speaker or not.

$$f_{cos}(w_1, w_2) = \frac{w_1.w_2}{|w_1||w_2|}$$

X-vectors

X-vectors are detailed in [16]. In this technique, a deep neural network (DNN) is trained to discriminate between the speakers in the training data. Each training example consists of a chunk of features (around 3 seconds) and the corresponding speaker label. An embedding called the x-vector is extracted from a designated layer. Thus, the variable-length utterances are mapped to fixed dimensional x-vectors. Unlike i-vector training which is unsupervised, speaker labels are needed for training [17].

An example architecture of the DNN is given in Table 2.1. T is the segment length, N is the number of speakers. There are layers that operate on speech frames, a statistical pooling layer that aggregates over frame-level representations to give segment-level representations, and in the end a softmax layer. First 5 layers work with a TDNN architecture [18]. The x-vectors are extracted from segment6.

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

Table 2.1: The x-vector DNN architecture.

2.5.2 Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) is the most commonly used clustering algorithm for diarization. It utilizes a bottom up approach in which the clustering is initialized with one cluster for each data point. It aims at reducing the number of clusters by 1 in each iteration by merging two of the most similar clusters. The basic algorithm is as follows.

```
while (num-clusters > min-clusters && merge-cost <= threshold) {
    if (size-of-new-cluster <= max-cluster-size) {
        merge two clusters with lowest cost
    }
}
```

2.5.3 Distance metrics

Probabilistic Linear Discriminant Analysis

Probabilistic Linear Discriminant Analysis (PLDA) [19] [20] is the state-of-the-art scoring technique for diarization. It is used to generate similarity scores between each pair of i-vectors or x-vectors in a given recording. PLDA is a probabilistic extension of Linear Discriminant Analysis (LDA). LDA is similar to Principal Component Analysis

(PCA) where data can be projected onto a lower dimension. Unlike PCA whose goal is data representation, the goal of LDA is class separability. Just like LDA, PLDA projects the data onto a lower dimension while maximizing discriminative ability by maximizing the ratio of between-class and within-class variance.

PLDA is used to calculate the log-likelihood ratio as follows. H_1 is the hypothesis that the speakers of the segments are the same. H_0 is the hypothesis that they are different.

$$f_{cos}(w_1, w_2) = \log p(w_1, w_2|H_1) - \log[p(w_1|H_0)p(w_2|H_0)]$$

2.6 Evaluation

Diarization Error Rate (DER)

This metric is the most popular for evaluating diarization. Basically, it is the total percentage of reference speaker time that is not correctly attributed to a speaker. "Correctly attributed" relates to the optimal mapping between reference and system speakers. The DER formula is given below.

$$DER = \frac{FA + MISS + ERROR}{TOTAL}$$

$TOTAL$ is the sum of durations of all reference speaker segments. FA or False Alarm, is the system speaker time that is not attributed to a reference speaker. This is when the system misinterprets a non-speech segment as speech. $MISS$ is the reference speaker time that is not accounted for by a system speaker. This is when the system misses a speech segment and classifies it as non-speech. $ERROR$ is the total system speaker time that is attributed to the wrong speaker. This is when the system incorrectly guesses the identity of a speaker.

It is easy to see that the best possible DER is 0%. Also if FA is always zero, 100% is the upper limit for DER. Because of its definition, speakers with more speaking time tend to contribute more to DER than speakers with less speaking time.

2.7 Kaldi toolkit

2.7.1 Introduction

The Kaldi speech recognition toolkit [1] was started in 2009 at John Hopkins University (JHU) with an aim to create a modern, well-engineered general purpose speech toolkit with a permissive license. Other aims of the project were to have a finite-state transducer (FST) based framework and have extensive linear algebra support. The toolkit depends on some external libraries that are freely available - OpenFST, BLAS and LAPACK. The toolkit includes programs written in C++ that wrap these libraries, which are in turn called from bash/python scripts that can be combined to create complete recipes that do a specific job like speech/speaker recognition, diarization etc.

Kaldi includes ready-to-use complete recipes for popular and widely available datasets such as those provided by the Linguistic Data Consortium (LDC). They are available as subdirectories of the `egs` directory in Kaldi's root directory.

```
(base) [acq18mh@snarl ~/workspace]$ ls kaldi/egs
README.txt           casia_hwdb               fisher_swbd
aidatatang_200zh     chime1                   formosa
aishell              chime2                   gale_arabic
aishell2             chime3                   gale_mandarin
ami                  chime4                   gp
an4                  chime5                   heroico
apiai_decode         cifar                    hkust
aspire               commonvoice              hub4_english
aurora4              csj                      hub4_spanish
babel                dihard2-voxceleb         iam
babel_multilang      dihard_2018              iban
bentham              fame                     ifnenit
bn_music_speech      farsdat                  librispeech
callhome_diarization fisher_callhome_spanish lre
callhome_egyptian    fisher_english           lre07
```

The C++ executables are have specific functionality so that they can be chained together in a typical Unix-like fashion to create complex pipelines. Given in Figure 2.3 is a simplified diagram of the Kaldi architecture taken from [1].

2.7.2 Briew Overview of a Kaldi Recipe

This section describes some important parts of a typical Kaldi recipe. JHU's recipe for the DIHARD 2018 challenge `egs/dihard_2018` is used as reference. It looks like the following.

```
(base) [acq18mh@snarl kaldi]$ ls -l egs/dihard_2018
-rw-r--r--. acq18mh mini Jun 24 21:43 README.txt
-rwxr-xr-x. acq18mh mini Aug 26 03:26 cmd.sh
drwxr-sr-x. acq18mh mini Aug 19 01:04 conf
```

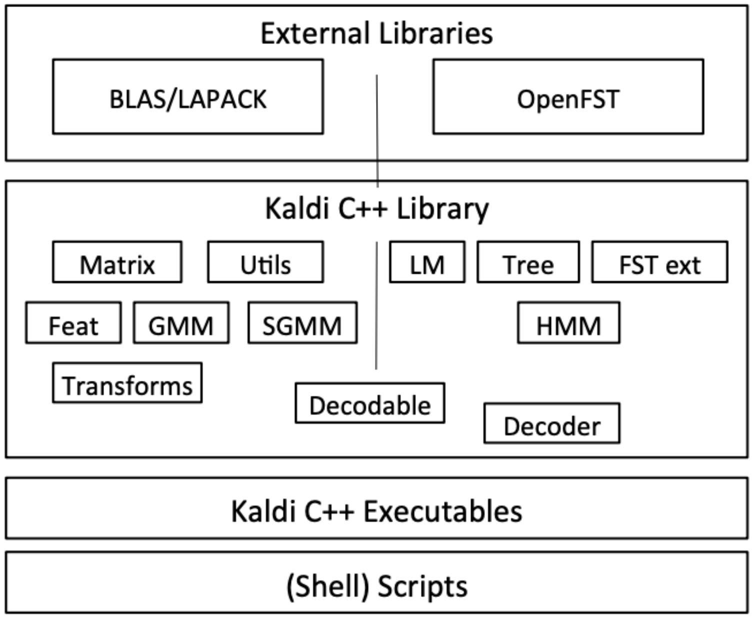

Figure 2.3: Kaldi architecture. Image source [1]

```
drwxr-sr-x. acq18mh mini Sep  7 19:21 data
lrwxrwxrwx. acq18mh mini Jun 24 21:43 diarization ->
 ../../callhome_diarization/v1/diarization
drwxr-sr-x. acq18mh mini Sep  7 19:20 exp
drwxr-sr-x. acq18mh mini Aug 30 01:16 local
drwxr-sr-x. acq18mh mini Sep  7 19:20 mfcc
-rwxr-xr-x. acq18mh mini Jun 24 21:43 path.sh
-rwxr-xr-x. acq18mh mini Aug 26 03:26 run.sh
lrwxrwxrwx. acq18mh mini Jun 24 21:43 sid -> ../../sre08/v1/sid
lrwxrwxrwx. acq18mh mini Jun 24 21:43 steps -> ../../wsj/s5/steps
lrwxrwxrwx. acq18mh mini Jun 24 21:43 utils -> ../../wsj/s5/utils
```

Most Kaldi recipes have a top level script called "run.sh" in the recipe directory. This is the starting point of the recipe. There are several scripts that are reused across many recipes. In many cases this is achieved by creating symlinks to other recipes, as seen by the arrows in the above listing. This recipe reuses scripts from the `callhome_diarization`, `wsj` and `sre08` recipes. Scripts that are specific to this recipe are placed in `local`, configuration files are placed in `conf` and `exp` contains files needed or created during the experiment. The `data` directory is important because it holds a lot of information about the data that the experiment works on.

```
(base) [acq18mh@snarl dihard_2018]$ ls -l data
```

```
-rw-r--r--.  1 acq18mh mini 3909183 Aug 28 03:32 feats.scp
-rw-r--r--.  1 acq18mh mini    1642 Aug 28 03:31 reco2num_spk
-rw-r--r--.  1 acq18mh mini 1768875 Aug 28 03:31 rttm
-rw-r--r--.  1 acq18mh mini  808834 Aug 28 03:32 segments
-rw-r--r--.  1 acq18mh mini  289353 Aug 28 03:32 spk2utt
drwxr-sr-x. 42 acq18mh mini    4096 Aug 28 03:32 split40
-rw-r--r--.  1 acq18mh mini  423522 Aug 28 03:32 utt2dur
-rw-r--r--.  1 acq18mh mini  392200 Aug 28 03:32 utt2num_frames
-rw-r--r--.  1 acq18mh mini  465297 Aug 28 03:32 utt2spk
-rw-r--r--.  1 acq18mh mini   27388 Aug 28 03:32 wav.scp
-rw-r--r--.  1 acq18mh mini   27388 Aug 28 03:32 vad.scp
```

There are 3 files that must be created manually in diarization recipes: wav.scp, utt2spk, segments. The wav.scp file holds information about the actual audio files and how to extract a waveform from them in a usable form. The utt2spk file contains identifiers for all utterances mapped to the identity of the speaker who speaks them. The spk2utt file is the opposite and is created automatically. The segments file holds information about the timestamps of the spoken utterances in the audio files. The feats.scp file is created after feature extraction and has the location of the features on disk. The vad.scp file is created after voice activity detection (or SAD) and contains the location of the vectors generated after SAD. These vectors are equal to the MFCCs in length and contain 0's and 1's, classifying each frame as speech or non-speech. The split40 directory contains a split of this “data directory” into 40 smaller data directories, for parallelization purposes.

Chapter 3

DIHARD challenge setup

This chapter discusses some specifics of the DIHARD challenge including the rules, evaluation mechanisms and datasets. Most of these details are published in [21] and [22].

3.1 Task Definition

The goal of the challenge is to automatically split audio recordings into segments based on speaker identity. These segments should be output in the form of Rich Transcription Time Marked (RTTM) files. Please see Appendix A for the RTTM format specification. Small pauses of less than 200 ms should be merged into a single contiguous segment. During these pauses the speaker is not considered to produce any vocalization. This obviously includes speech, but also speech errors, infant babbling, breaths, coughs, lip smacks, sneeze, laughs, humming. Basically any sound produced by the speaker by using the human vocal apparatus.

3.2 Evaluation Tracks

As a participant, there are four possible tracks to participate in. Each of these have their own leader boards. There are two input conditions - single channel and multichannel audio. There are also two SAD conditions - reference SAD, where manually transcribed segment boundaries are made available for use, and system SAD, where the participant is expected to run their own SAD algorithms on the raw audio. These two conditions give rise to four possible tracks.

- Track 1 - single channel audio using reference SAD
- Track 2 - single channel audio using system SAD

- Track 3 - multichannel audio using reference SAD
- Track 4 - multichannel audio using system SAD

3.3 Details on Data Provided

The single channel audio may be taken from a single distant microphone, or from one of the channels from a microphone array, head-mounted microphones, or a mix of these. The multichannel audio is drawn from the CHiME-5 dinner party corpus, which consists of conversational speech from dinner parties held in real homes. It is composed of several different recording sessions - each corresponding to a different party. The house had multiple arrays, so each session contains output from multiple distant microphone arrays, each containing multiple channels. Participants are required to generate one diarization output per array and are free to use any number of channels from the array. For example, if a recording session contains 4 microphone arrays, participants are expected to generate 4 RTTM files.

For the reference SAD condition, the reference segmentation has been generated by automatically merging overlapping or very close speaker turns from the human reference diarization. This is provided in the form of HTK [23] label files. Each audio recording has one label file. The label file has one line for each speech segment with the format `<start-timestamp><end-timestamp>speech`.

```
0.000 3.513 speech
4.698 7.133 speech
7.377 12.826 speech
13.284 16.797 speech
17.312 21.201 speech
...
```

3.4 Scoring

The outputs of the systems are judged by two metrics.

3.4.1 Diarization Error Rate (DER)

This is the primary scoring metric. The details have been discussed earlier in Chapter 2.

3.4.2 Jaccard Error Rate (JER)

The JER metric has been newly developed for DIHARD II. Its goal is to weigh every speaker's contribution to the score equally regardless of how much speech they produce, unlike DER. This is done as follows.

First, an optimal mapping is determined between the reference and system speakers using the Hungarian algorithm [24]. Then for each pair of speakers, the Jaccard Index [25] is computed. The Jaccard Index measures the similarity between two sets, and is defined as their intersection size divided by union size. This is always between 0 and 1. A value of 0 implies no similarity and 1 implies maximum similarity. We can invert the meaning of 0 and 1 by subtracting it from 1, making it an error measure instead of a similarity measure. We call this measure the speaker-specific JER, or JER_{ref}. The overall JER is then defined to be the average of JER_{ref} scores for each pair.

This can explained better using a Venn diagram. Suppose we have sets SYS and REF corresponding to system speaker duration and reference speaker duration respectively. The red region corresponds to the FA speech duration, blue region corresponds to the MISS speech duration, purple region corresponds to the speech duration for which speaker was correctly identified and TOTAL corresponds to the sum of all three regions.

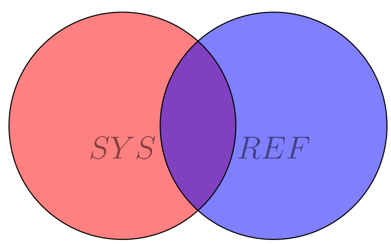

Figure 3.1: Jaccard Index Venn diagram

Jaccard Index is equal to the area of the purple region divided by the area of the whole coloured region. The JER_{ref} of this would be the sum of the red and blue region, divided by the whole coloured region.

$$JER_{ref} = \frac{FA + MISS}{TOTAL}$$

$$JER = \frac{1}{N} \sum_{ref} JER_{ref}$$

It can be understood from this definition that the JER never exceeds 100%.

3.5 Datasets

Since we work with the 2018 DIHARD datasets, we only describe their composition and not of the 2019 datasets. Please refer to Appendix B for a detailed description of the domains and sources. The development set consists of approximately 19 hours of 5-10 minute audio samples drawn from the domains given in Table 3.1. The evaluation set consists of samples drawn from the same domains and sources with a few exceptions. It has 21 hours of 5-10 minute long speech samples. All audio files are 16 KHz single channel FLAC files.

Domain	Source
Child language acquisition recordings	SEEDLingS
Supreme Court oral arguments	SCOTUS / OYEZ
Clinical interviews	ADOS interviews
Radio interviews	YouthPoint recordings
Map tasks	DCIEM Map Task Corpus
Sociolingustic interviews	SLX Corpus
Meeting speech	RT04 datasets
Audiobooks	LibriVox
YouTube videos	VAST project

Table 3.1: Composition of DIHARD 2018 dev set.

3.6 Evaluation Rules

The DIHARD challenge is an open evaluation, and the system outputs are supposed to be generated locally and submitted to LDC for scoring. Since portions of the test data are available in following corpora that are already widely available, usage of them for training is prohibited.

- DCIEM Map Task Corpus (LDC96S38)
- MIXER6 Speech (LDC2013S03)
- Digital Archive of Southern Speech (LDC2012S03 and LDC2016S05)
- NIST SRE10 evaluation data
- NIST SRE12 evaluation data
- DIHARD I evaluation sets

- the SEEDLingS subset of HomeBank

The National Institute of Standards and Technology (NIST) evaluations have a practice in which a "forgiveness collar" is applied to either sides of reference segments, to get rid of errors from inconsistent human annotations and uncertainty about when a speaker begins or ends. These collars are not scored. This does not apply for DIHARD and there are no forgiveness collars used. Overlapping speech is also evaluated - so the ideal system is expected to output overlapping segments if two speakers overlap in the source audio. A system that generates only flat segmentations (no overlap) will have higher DER because the overlapping segments could give rise to missed speech and speaker errors.

Lastly, human investigation of the evaluation data is not allowed. Although participants are allowed to automatically derive information using the development and evaluation sets.

3.7 Best systems from DIHARD 2018

Given below in Table 3.2 are the best systems from DIHARD 2018.

Rank	Team (System)	DER %
1	JHU (Sys4)	23.73 %
2	JHU (Sys5)	23.99 %
3	USTC-iFLYTEK (system1)	24.56 %
4	USTC-iFLYTEK (system4)	24.96 %
5	JHU (Sys2)	25.06 %
6	BUT (dev-s6)	25.07 %
7	BUT (dev-s2)	25.39 %
8	USTC-iFLYTEK (system2)	25.67 %
9	ViVoLab (VivoLab Sys 4)	26.02 %
10	ViVoLab (VivoLab Sys 2)	26.15 %

Table 3.2: Best systems from DIHARD 2018.

Chapter 4

Baseline setup

4.1 Overview

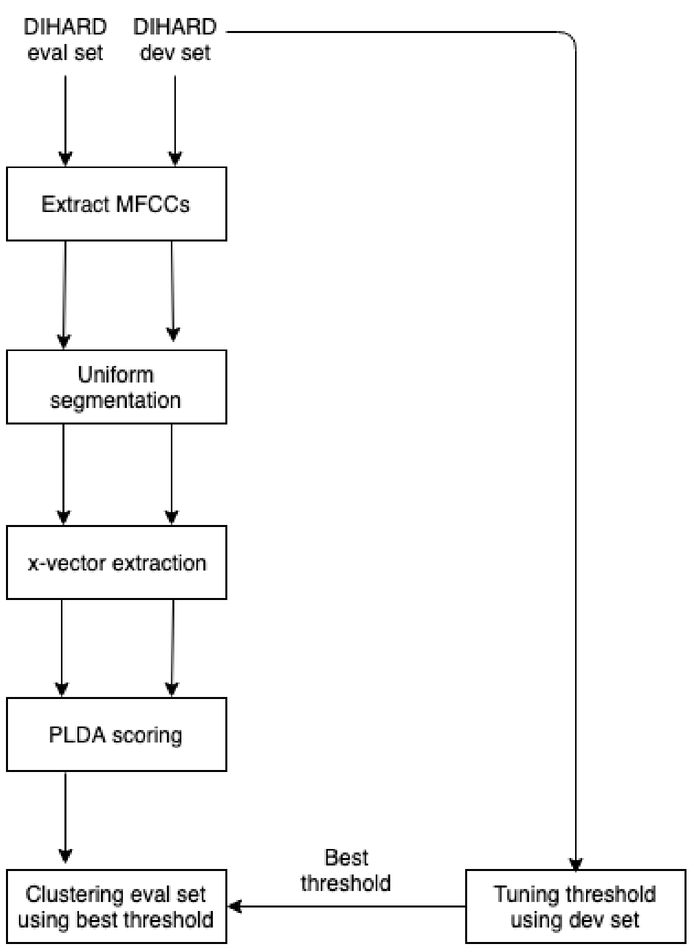

Figure 4.1: Block diagram of DIHARD diarization baseline.

There are three software baselines provided by the DIHARD II organizers, each for the parts of speech enhancement, speech activity detection and diarization. The speech enhancement baseline and the speech activity detection are meant to be used together in the case of system-generated SAD (tracks 2 and 4), but since we only work with reference SAD, we do not need them. Thus we will only describe the diarization baseline in the following sections.

The diarization baseline is based on the best performing submission [26] from JHU in the previous year's DIHARD challenge (DIHARD I). There are 4 Kaldi recipes, each for an evaluation track, but we will focus only on the recipe for Track 1 since we only work with single channel audio and reference speech segmentation.

4.2 Baseline directory structure

The baseline repository is localted at `https://github.com/iiscleap/DIHARD_2019_baseline_alltracks` and has the following directory structure. Some of the irrrelevant files have been removed.

```
DIHARD_2019_baseline_alltracks/
|-- data
|   |-- final.raw
|   |-- max_chunk_size
|   |-- min_chunk_size
|   |-- plda_track1
|   |-- plda_track2
|   |-- plda_track3
|   |-- plda_track4
|-- README.md
|-- recipes
|   |-- track1
|   |-- track2
|   |-- track2_den
|   |-- track3
|   |-- track4
|   `-- track4_den
|-- scripts
|   |-- alltracksrun.sh
|   |-- flac_to_wav.sh
|   |-- make_data_dir.py
|   |-- md_eval.pl
```

```
|   |-- prepare_feats.sh
|   |-- prep_eg_dir.sh
|   `-- split_rttm.py
`-- tools
    |-- env.sh
    |-- install_dscore.sh
    |-- install_kaldi.sh
}
```

The baseline code modifies and reuses the `egs/dihard_2018` recipe that was checked into Kaldi by the researchers at JHU. It does this by copying over new scripts and data that is needed to the `egs/dihard_2018` directory, `cd`'ing to that directory and running the recipe from there.

The `data` directory has pre-trained models (in Kaldi binary format) and some configuration parameters. `final.raw` is a DNN x-vector extractor, and the files starting with `plda` are the PLDA backends trained for the 4 tracks. The `recipes` directory has the `run.sh` files for all 4 recipes, we only care about `track1`. The `scripts` directory has extra scripts that are needed on top of the `egs/dihard_2018` Kaldi recipe - `alltracksrun.sh` is the main diarization script, `make_data_dir.py` makes the Kaldi data directory from the DIHARD datasets (creating the files wav.scp, segments, utt2spk), `prep_eg_dir.sh` copies the extra files from this repository to the `egs/dihard_2018` directory, `md_eval.pl` [27] is a diarization evaluation script that was developed by NIST, and others are self-explanatory. The `tools` directory holds scripts to install Kaldi and dscore [28], which are installed in the same directory.

4.3 Initial segmentation

The initial segmentation step is done by `make_data_dir.py`. It deals with separating speech and non-speech segments from the recording files using the reference SAD.

This results in a bunch of segments which are known to be containing only speech. These are treated as "utterances" in Kaldi terminology and act as keys in the `utt2spk`, `feats.scp` and `segments` files. These files reside in two Kaldi "data directories", one for each dev and eval.

4.4 Features

The baseline then extracts 30 dimensional MFCC features from 30 Mel bins using a 25 ms window that shifts by 10 ms. It uses the standard `steps/make_mfcc.sh` Kaldi script for this. The MFCC configuration used `mfcc.conf` is given below.

```
--sample-frequency=16000
--frame-length=25 # the default is 25
--low-freq=20 # the default.
--high-freq=7600 # Nyquist (8k in this case).
--num-mel-bins=30
--num-ceps=30
--snip-edges=false
```

Later, cepstral mean and variance normalization (CMVN) with a 3 second sliding window is applied using the `apply-cmvn-sliding` Kaldi tool.

4.5 Subsegmentation

After MFCC extraction is done for the segments, the segments are uniformly divided into smaller 1.5 second subsegments with a 0.75 second overlap. This creates new Kaldi data directories (one each for dev and eval sets) with newer keys corresponding to each subsegment.

4.6 Speaker representation

The baseline extracts an 512-dimensional x-vectors from each subsegment using `extract_xvectors.sh` from `egs/callhome_diarization`. The DNN extractor `final.raw` was pre-trained on the datasets VoxCeleb I and II [29], along with added augmentation. The augmentation was done by additive noise (noise, music, babble) using the MUSAN [30] dataset and reverberation using the Room Impulse Responses (RIRs) [31].

4.7 Scoring

For scoring two x-vectors, PLDA scores are used as a distance metric. The `plda_track1` file is used which has been trained using x-vectors extracted from a random subset (size 128k) of the VoxCeleb dataset. To adapt the extracted x-vectors to the DIHARD domain, they are whitened with a whitening transform learned from the DIHARD development set. Each pair of x-vectors within a recording is then scored using the PLDA backend by reusing `score_plda.sh` from `egs/callhome_diarization`. These scores are stored in matrix form for each recording.

4.8 Clustering

The x-vectors are then clustered using agglomerative clustering and a parameter sweep is done on the dev set to find the threshold that maximises the DER on the dev set. This threshold is then used for clustering the x-vectors of the eval set. The `cluster.sh` script from `egs/callhome_diarization` is used for clustering.

4.9 Diarization output

The clustering output is used to generate RTTMs using the script `make_rttm.py` from `egs/callhome_diarization`. The RTTMs give a flat segmentation of the recordings with no overlap. Since the x-vectors were extracted from segments that were overlapping, care needs to be taken when two adjacent segments are assigned to a different speaker. The script places the speaker boundary midway between the end of the first segment and the start of the second segment.

Chapter 5

Experiments and Results

5.1 Modifications to the baseline scripts

The diarization baseline scripts were modified to make it easier to run experiments. Arguments were added to specify the type of speaker embedding to be used, the specific trained model to be used, the specific PLDA backend to be used, and the PCA dimension to be used. It was made possible to run multiple experiments of the same recipe in parallel without overwriting each others' files. Some new scripts were written and a new C++ executable was added. The dscore library was modified to output a breakdown of the DER metric (MISS, FA, ERROR) instead of just the DER. Most of the modifications are present in the files `recipes/track1/run.sh` and `scripts/alltracksrun.sh` from the DIHARD baseline repository.

5.2 Baseline results

The baseline results shown in Table 5.1 have been computed by running the 2019 baseline on the datasets from previous year's DIHARD challenge (2018). This is because it was not possible to register for DIHARD 2019 before the deadline, and thus access to 2019 datasets was denied.

Luckily the 2018 datasets were released as a part of the June LDC newsletter [32]. Since the basic problem statement of the challenge remains the same as last year, the last year's datasets can still be used without any trouble. But this unfortunately means that it is no longer possible to verify the computed baseline scores with official scores, because the official scores only exist for the 2019 datasets. The only useful hint was found on the webpage at [33], which mentions a rough score of 20.71 on the 2018 development set.

The baseline (and all its modifications that use x-vectors) takes about 15 minutes to run on a 32-core machine with around 60GB of memory.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	19.96	11.62	53.92
eval	26.58	17.05	59.44

Table 5.1: Baseline scores.

The results already seem pretty good, considering that the system is not very complex. The best JHU system in DIHARD 2018 had an eval DER of 23.73%, and that included doing Variational Bayes refinement [34] as an extra step.

It is also important to mention that Missed Speech and False Alarm are not mentioned in any of the following experiments, because they are always constant (since we work with reference SAD only). Missed speech is always 8.34% for the dev set and 9.52% for the eval set. False alarm is always zero for both, as expected. Missed speech should also be zero from the same reason, but it is not because overlapping speaker detection is not done in any experiment. Only flat segmentations (no overlap in output segments) are generated, which causes missed speech errors wherever there is overlap. If the reference label files were created correctly by the organizers, this also means that 8.34% and 9.52% of the total speech duration (dev and eval respectively) is overlapping.

5.3 Using Existing Pre-trained Models

The first set of experiments was to see how certain pre-trained models freely available on the Internet perform. This was mainly done to get a quick idea of how an i-vector model performs, and whether training one is worthwhile. Another reason was to get comfortable with switching models in the baseline, which was not trivial because a lot of convoluted shell scripts needed to be understood for avoiding mistakes.

5.3.1 Kaldi VoxCeleb x-vector model

This model was downloaded from the Kaldi models webpage [35], and closely follows the recipe in [16]. The recipe used for training is available in Kaldi at `egs/voxceleb/v2`. Similar to the baseline recipe, this model is also trained on a combination of VoxCeleb I and II along with augmentation. Both produce 512-dimensional x-vectors. There are two differences though:

- The PLDA backend included in this model is trained on the whole training data, which consists of 1,276,888 utterances. The baseline PLDA backend is trained on a subset where each segment is at least 3 seconds long.

- The PLDA backend here is trained after doing LDA dimensionality reduction to 200 on the x-vectors. The baseline PLDA backend is trained directly on 512-dimensional x-vectors.

The results are given in Table 5.2.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	22.86	14.52	51.74
eval	26.42	16.89	55.55

Table 5.2: Scores with Kaldi VoxCeleb x-vector model.

This has already produced a small improvement over the baseline. The amount of training data used to train this PLDA backend is more than the baseline but the PLDA dimensionality is smaller, so it is hard to pinpoint the exact reason why this has better performance.

5.3.2 Kaldi VoxCeleb i-vector model

This model was also downloaded from the same Kaldi models webpage as the previous recipe, and closely follows the recipe in [16]. The recipe used for training is available in Kaldi at `egs/voxceleb/v1`. Similar to the x-vector recipe, the model is also trained on a combination of VoxCeleb I and II, but without augmentation. This is because i-vectors are unable to effectively utilize data augmentation, unlike x-vectors [16]. The UBM is trained on 2048 Gaussians using all the training utterances. The i-vector extractor is trained using the longest 100k utterances and produces 400-dimensional i-vectors. I-vectors are extracted for all the training utterances, reduced to 200 dimensions using LDA, and then used to train a PLDA backend.

The results of the diarization recipe modified to use this i-vector extractor are given in Table 5.3. Extracting i-vectors for the dev and eval set take much longer than extracting x-vectors, so this took around 6 hours. Every diarization run with i-vectors took a similar amount of time.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	26.18	17.83	59.81
eval	32.03	22.51	65.22

Table 5.3: Scores with Kaldi VoxCeleb i-vector model.

Clearly, this is much worse than the x-vector model which is expected according to [16] and [26].

5.4 Training Custom Models

Training our own models was considered because that would allow models to be trained on in-domain data (the DIHARD development set). This should possibly give better results, considering that the model would get a chance to train on data that has similar composition to the evaluation set. The amount of data available in the dev set is relatively small, only 19 hours, so we do not expect great results, but it is a good experiment to try. The recipes in `egs/voxceleb` were used as a starting point for both i-vector extractor and x-vector extractor training.

5.4.1 Training with DIHARD development set

The following results in Table 5.4 were obtained by training an x-vector model on the DIHARD development set. The reference RTTM files for the dev set were used to generate 28241 training utterances from 221 speakers. The recipe imposes a minimum feature length of 400 frames and a minimum 8 utterances per speaker, so after filtering only 2726 utterances from 90 speakers were used to train with. This meant that the trained DNN had 90 output nodes. Since the VoxCeleb dataset does not come with reference speech segmentation, the Kaldi program `compute-vad` is used with the configuration given below to classify each frame into speech or non-speech.

```
--vad-energy-threshold=5.5
--vad-energy-mean-scale=0.5
--vad-proportion-threshold=0.12
--vad-frames-context=2
```

The embedding layer has 512 nodes. X-vectors were extracted from the full dev set and reduced to 200 dimensions using LDA before training a PLDA backend.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	41.35	33.00	74.60
eval	43.16	33.64	75.96

Table 5.4: Scores with x-vector model trained on DIHARD dev.

These seem to be pretty bad, but they align with the known fact that x-vectors perform poorly on small amounts of data, as mentioned in the conclusion of [36].

As an additional experiment, augmentation was applied to the dev set. The augmentation was done similar to what the baseline does: 4 variants of the training set were created (reverb, noise, babble, music) and added to the original set, multiplying the number of utterances by 5. This resulted in 141,205 utterances from 221 speakers, reduced to 13,630 utterances from 90 speakers after filtering. This increased amount of training data resulted in a small increase in performance, as given in Table 5.5. But it still did not increase the performance beyond the baseline.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	35.54	27.20	76.56
eval	39.43	29.91	78.65

Table 5.5: Scores with x-vector model trained on DIHARD dev + augmentation.

Next, an i-vector model was trained on the dev set. Knowing that i-vectors perform better on relatively small amounts of data compared to x-vectors [36], it was expected to have better performance than the previous experiment. And it did perform much better, as given in the Table 5.6.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	24.90	16.56	57.68
eval	33.97	24.44	64.44

Table 5.6: Scores with i-vector model trained on DIHARD dev.

This confirmed the findings from [36] that the i-vector model without augmentation performs better than the x-vector model with augmentation, if the amount of available data is small.

Training the in-domain i-vector model with augmentation was also considered during the last stages of the dissertation. The i-vector training on the the non-augmented dev set took 17 hours on a 32-core machine, with near 100% CPU usage all the time. Thus due to lack of time, i-vector training was not attempted with an augmented dev set, which would be 5 times bigger.

5.4.2 Training with combination of VoxCeleb and DIHARD development set

For the next set of experiments the amount of training data was increased by adding data from VoxCeleb I. VoxCeleb II was not used because it is 7 times bigger than VoxCeleb I, making the training set too big, especially for i-vector training. There are 153,516 utterances from 1,251 speakers in VoxCeleb I, so this increases the total amount of training data significantly. All the parameters of the training remained unchanged.

The results of the x-vector model trained on the combination of VoxCeleb I and DIHARD dev set are given in Table 5.7.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	23.45	15.11	56.89
eval	29.44	19.92	61.37

Table 5.7: Scores with x-vector model trained on combination of VoxCeleb I and DIHARD dev.

Now that the amount of training data available for x-vector training is not small, it showed a good improvement in performance compared to the model that was just trained on the DIHARD dev set. But it still does not cross the baseline, which was trained with a much larger amount of data.

It was also attempted to train an x-vector model after adding augmentations to the combination of VoxCeleb I and DIHARD dev. But attempts for that were unsuccessful as the DNN training kept getting interrupted because of memory allocation failures on the GPU. This was most likely because the GPUs on the grid were shared with other users, causing available GPU memory to be low. Even though there were multiple GPUs, Kaldi seems to choose GPUs randomly, regardless of that fact that they are free or busy. This problem likely did not happen when training with just the dev set because the training was much faster, reducing the chances of someone submitting a parallel GPU job. This could possibly be fixed by setting the GPU to exclusive mode using the `nvidia-smi` tool, but superuser access was needed for that.

An i-vector model was also trained on the combination of VoxCeleb I and DIHARD dev set. The results are given in Table 5.8. There was an improvement compared to the i-vector model only trained on the dev set, which is expected. Although this time with increased data it did not outperform the x-vector model.

Dataset	DER (%)	Speaker error (%)	JER (%)
dev	25.15	16.81	56.78
eval	31.61	22.08	60.74

Table 5.8: Scores with i-vector model trained on combination of VoxCeleb I and DIHARD dev.

5.5 Feature concatenation

For the final set of experiments, a new approach was tried for speaker embeddings. No new i-vector or x-vector extractor was trained in these experiments. Instead, the Kaldi pre-trained models were used to extract both i-vectors and x-vectors from the combination of VoxCeleb I and DIHARD dev set. These two sets of vectors were concatenated to form a set of new vectors (called c-vectors henceforth for simplicity).

These c-vectors were reduced in dimension using LDA and later used to train a PLDA backend using the speaker labels available. The diarization recipe was also updated to extract both i-vectors and x-vectors from the dev and eval sets and concatenate them if c-vectors were to be used.

This new PLDA model was trained with varying dimensions from 200 up to 800. The results are given below in Table 5.9.

Dim	DER(dev)	DER(eval)	S.Err(dev)	S.Err(eval)	JER(dev)	JER(eval)
200	26.40	28.18	18.05	18.65	51.77	54.66
300	25.11	26.64	16.76	17.12	49.02	52.48
400	25.25	26.63	16.91	17.10	47.65	52.54
500	26.23	25.52	17.89	15.99	47.58	50.30
600	25.53	25.22	17.19	15.70	47.17	50.27
700	24.88	24.95	16.54	15.42	47.11	50.85
800	23.62	24.64	15.27	15.12	46.63	51.05

Table 5.9: Scores with PLDA backend trained on c-vectors extracted from combination of VoxCeleb I and DIHARD dev.

This best model is when c-vectors are 800 dimensional, at 24.64% DER on the eval set. This is a significant improvement over the baseline, which is almost 2% higher in absolute terms and 7.3% higher in relative terms.

A new Kaldi C++ executable called `append-vectors` was written to concatenate two sets of arbitrary vectors because there was no program to do this that worked with vectors. There are executables called `append-vector-to-feats` and `concat-feats`, but they only work with matrices because feats are stored as matrices in Kaldi.

5.6 Hyperparameters

This sections describes the hyperparameters that were used to run the above experiments. Due to lack of time, it was not possible to experiment with tuning any of these and the defaults existing in the baseline or the reference training recipes were used.

MFCC feature vector length

The x-vector systems used 30 dimensional MFCCs because Kaldi's pre-trained x-vector extractor DNN had 30 inputs. For consistency, the MFCC dimensionality for the the self trained x-vector extractor DNN was also set to 30. Similarly, the i-vector systems used 72 dimensional features (24 dimensional MFCCs with delta and delta-delta coefficients appended) because Kaldi's pre-trained UBM and i-vector extractor was trained with 72 dimensional Gaussians. The self trained i-vector extractor also used the same.

Speaker Embedding Length

This is an obvious one because it has a significant effect on the amount of speaker information that is present in each embedding. The Kaldi pre-trained models extracted 400 dimensional i-vectors, and the self-trained i-vector extractor was configured to do the same. Similarly in the x-vector case the length was always 512.

Segment Length and Overlap

The baseline diarization system used uniform segmentation with segment length 1.5 sec and 0.75 sec overlap (50%). This was not changed in any of the diarization runs.

Clustering Threshold

The threshold during the clustering step in diarization is in terms of the log likelihood ratio provided by PLDA scores. In a perfectly calibrated system, the score is 0. The best threshold was selected from the list -0.5, -0.4, -0.3, -0.2, -0.1, -0.05, 0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5 using the computed DER on the dev set.

Number of UBM Gaussians

The number of Gaussians in the pre-trained and self-trained trained UBMs was always 2048.

PLDA dimension

The dimensionality of the PLDA backend was always 200, except in the c-vectors experiment where it varied from 200 to 800, and the baseline where it was 512.

5.7 Discussion of results

Three main experiments were carried out. The first included running the baseline with pre-trained i-vector and x-vector models available on the Kaldi website that have been trained on a large amount of data from the VoxCeleb dataset. The VoxCeleb dataset consists of audio taken from YouTube videos recorded in a variety of conditions, so there is significant channel, environment, and speaker variance. Other baseline parameters like segment length, overlap, clustering threshold, PLDA dimension were left unchanged. The x-vector model resulted in a similar performance as the baseline

(26.42%), and the i-vector model performed much worse with 32.03%. This is expected as x-vectors have been found to out-perform i-vectors [16], especially with data augmentation.

The second experiment involved training own models for i-vector and x-vector extraction. The motivation for this was to take advantage of the in-domain data (the DIHARD dev set) to have better suited models for the DIHARD eval set, the composition of which is similar to the dev set. There were two parts to this - in the first part, these models were trained on only the DIHARD dev set. This was predicted to not have good performance as the dev set has only 19 hours of audio from about 200 speakers, which is really small especially for the x-vector model. As predicted, the x-vector model's performance was terrible at 43.16% (without augmentation) and 39.43% (with augmentation). The i-vector model's performance was much better at 34.61% suggesting that i-vectors perform better with small amounts of data. In the second part, the models were trained on a combination of VoxCeleb I and the DIHARD dev set. The goal was to increase the amount of training data using an external source, yet retaining some in-domain data. Both the x-vector and i-vector models were trained using this combination (without augmentation) and the results showed a good improvement with 29.44% and 31.61% respectively, x-vectors performing better this time.

The third experiment involved reusing Kaldi's pre-trained i-vector and x-vector extractors (since they performed the best) but training own PLDA backend by using concatenated i-vectors and x-vectors. These new vectors were called c-vectors for simplicity. The i-vectors and x-vectors were extracted from the combination of VoxCeleb I and the DIHARD dev set. The baseline was also modified to extract both i-vectors and x-vectors from the dev and eval sets, concatenate them to get c-vectors, and score them using the new PLDA backend. Various dimensions of c-vectors were tried and the best performing was 800 at 24.64%. This is 2% lower than the baseline score which is at 26.58%. Scoring with c-vectors was also found to generalize better due to closer DER scores between dev and eval compared to the earlier experiments.

Chapter 6

Conclusions

The main aim of this dissertation was to study the state-of-the-art techniques in speaker diarization and create a Kaldi-based system for the DIHARD II challenge. A detailed overview of the diarization process was presented that talked about some of the past and recent techniques used in the field. The motivation behind the DIHARD challenge and the evaluation rules were discussed. The working of the diarization baseline provided by the organizers of the challenge was discussed in detail. A brief overview of the Kaldi toolkit was also provided.

Several difficulties were faced during the dissertation. The first difficulty was getting the DIHARD 2019 datasets. When the topic for this dissertation was finalized, the DIHARD registration deadline had already passed. Thus, access to the 2019 datasets was denied. Luckily, the organizers made the 2018 development set available from LDC in June. This made it possible to run the systems by slightly modifying the file structure of the dataset. But the actual performance were not seen until the evaluation set was released, which was in the next month in July. This caused a significant delay in the dissertation. Another big problem was making the i-vector extractor training work on the university grid. A lot of time was spent to realize that memory was a bottleneck on the systems with 60 GB memory, which led to thrashing and slowing down the training which was already extremely slow. It was fixed by forcing the recipe to run on the only two systems which had 250 GB of memory. Another problem was compiling Kaldi with GPU, without which the x-vector DNN training was extremely slow. Other general problems included navigating through unfamiliar shell script code, getting the DIHARD baseline to work on the grid, fixing Kaldi errors, and writing some Kaldi C++ code to create a new program.

The system was created using the 2018 DIHARD datasets because the 2019 datasets were not available for use. The main aim was achieved and the best diarization system consisted of training a PLDA backend using concatenated i-vectors and x-vectors extracted from a combination of the VoxCeleb I dataset and the DIHARD development set. This achieved a performance of 24.64% DER on the DIHARD evaluation set which surpassed the baseline performance by a 7.3% relative and 2% absolute improvement and a 2%. The c-vector model also seems to generalize better, since the differences in DER between the dev and eval set are smaller than the previous experiments. Looking at the systems from last year's challenge, this system is placed 4th. Some other experiments included training i-vector and x-vector extractors using scripts already present in Kaldi. This dissertation was a rewarding experience because it gave the opportunity to get familiar with recent advancements in the diarization field and implementing them, gaining a good amount of experience with Kaldi and getting to know people in the speech research community.

Chapter 7

Appendix A: RTTM File Format Specification

RTTM files are text files containing one turn per line, each line containing ten space-delimited fields:

- Type – segment type; should always by “SPEAKER”
- File ID – file name; basename of the recording minus extension (e.g., “rec1_a”)
- Channel ID – channel (1-indexed) that turn is on; should always be “1”
- Turn Onset – onset of turn in seconds from beginning of recording
- Turn Duration – duration of turn in seconds
- Orthography Field – should always by “<NA>”
- Speaker Type – should always be “<NA>”
- Speaker Name – name of speaker of turn; should be unique within scope of each file
- Confidence Score – system confidence (probability) that information is correct; should always be “<NA>”
- Signal Lookahead Time – should always be “<NA>”

For instance:
SPEAKER CMU 20020319-1400 d01 NONE 1 130.430000 2.350 <NA><NA>juliet <NA><NA>
SPEAKER CMU 20020319-1400 d01 NONE 1 157.610000 3.060 <NA><NA>tbc <NA><NA>
SPEAKER CMU 20020319-1400 d01 NONE 1 130.490000 0.450 <NA><NA>chek <NA><NA>

Chapter 8

Appendix B: Domains and Sources

8.1 Domains

- *Audiobooks*

  Excerpts from recordings of speakers reading aloud passages from public domain English language texts. The recordings were selected from LibriVox and each recording consists of a single, amateur reader. Care was taken to make sure that the chapters and speakers drawn from were not present in LibriSpeech, which also draws from LibriVox.

- *Broadcast interview*

  Student-lead radio interviews conducted during the 1970s with popular figures of the era (e.g., Ann Landers, Mark Hamill, Buckminster Fuller, and Isaac Asimov). The recordings are selected from the unpublished LDC YouthPoint corpus.

- *Child language*

  Excerpts from day long recordings of infant (6 to 18 months) speech. All audio was recorded in the home using a LENA recording device, which consists of a vest worn by the child into which a microphone has been sewn. Because of their age, the child "speech" consists of a mixture of simplistic speech consisting of short utterances (possible very disfluent), babbling, laughing, crying, and diverse uncategorizeable non-speech vocalizations. Other speakers may be present in the recording, typically one or more parents, but also siblings, friends of siblings, aunts and uncles, and adult friends of the parents. Some of the recordings have quiet backgrounds, while others have radios or televisions playing. All recordings were taken from the SEEDLingS corpus.

- *Clinical*

  Recordings of Autism Diagnostic Observation Schedule (ADOS) interviews conducted to identify whether a child fit the clinical diagnosis for autism. ADOS is a roughly hour long semi-structured interview in which clinicians attempt to elicit language that differentiates children with Autism Spec- trum Disorder from those without (e.g., "What does being a friend mean to you?"). The children included in this collection ranged from 12-16 years in age and exhibit a range of diagnoses from autism to non-autism language disorder to ADHD to typically developing. Interviews are typically recorded for quality assurance purposes; in this case, the recording was conducted using a ceiling mounted microphone. The recordings are selected from the unpublished LDC ADOS corpus.

- *Courtroom*

  Recordings of oral arguments from the 2001 term of the U.S. Supreme Court. The original recordings were made using individual table-mounted microphones, one for each participant, which could be switched on and off by the speakers as appropriate. The outputs of these microphones were summed and recorded on a single-channel reel-to-reel analogue tape recorder. All recordings taken from SCOTUS, an unpublished LDC corpus.

- *Map task*

  Recordings of speakers engaged in a map task. Each map task session contains two speakers sitting opposite one another at a table. Each speaker has a map visible only to him and a designated role as either "Leader" or "Follower". The Leader has a route marked on his map and is tasked with communicating this route to the Follower so that he may precisely reproduce it on his own map. Though each speaker was recorded on a separate channel via a close-talking microphone, these have been mixed together for the DIHARD releases. The recordings are drawn from the DCIEM Map Task Corpus (LDC96S38).

- *Meeting*

  Recordings of meetings containing between 3 and 7 speakers. The speech in these meetings is highly in- teractive in nature consisting of large amounts of spontaneous speech containing frequent interruptions and overlapping speech. For each meeting a single, centrally located distant microphone is provided, which may exhibit excessively low gain. For the development set, these meetings are drawn from RT04, while for the evaluation set they are drawn from ROAR.

- *Restaurant*

  Informal conversations recorded in restaurants using binaural microphones. Each session contains between 4 and 7 speakers seated at the same table at a restaurant at lunchtime and was recorded from a binaural microphone worn by a designated facilitator; the mix of the two channels recorded by this microphone are provided. This data exhibits the following properties, which are expected to make it particularly challenging for automated segmentation and recognition: – due to the microphone setup, the majority of the speakers are farfield – background speech from neighboring tables is often present, sometimes at levels close to that of the primary speakers in the conversation – background noise is abundant with clinking silverware, moving chairs/tables, and loud music all common – the conversations are informal and highly interactive with interruptions and frequent overlapped speech All data is taken from LDC's unpublished CIR corpus.

- *Sociolinguistic field recordings*

  Sociolinguistic interviews recorded under field conditions. Recordings consists of a single interviewer attempting to elicit vernacular speech from an informant during informal conversation. Typically, interviews were recorded in the home, though occasionally they were recorded in a public location such as a park or cafe. The development set recordings were drawn from SLX and the evaluation set from DASS.

- *Sociolinguistic lab recordings*

  Sociolinguistic interviews recorded under quiet conditions in a controlled environment. All data is taken from the PZM microphones of LDC's Mixer 6 collection (LDC23013S03).

- *Web video*

  English and Mandarin amateur videos collected from online video sharing sites (e.g., YouTube and Vimeo). This domain is expected to be particularly challenging as the videos present a diverse set of topics and recording conditions; in particular, many videos contain multiple speakers talking in a noisy environment, where it can be difficult to distinguish speech from other kinds of sounds. All data is selected from LDC's VAST collection.

8.2 Sources

- *ADOS*

  ADOS is an unpublished LDC corpus consisting of transcribed excerpts from ADOS interviews con- ducted at the Center for Autism Research (CAR) at the Children's Hospital of Philadelphia (CHOP). All interviews were conducted at CAR by trained clinicians using ADOS module 3. The interviews were recorded using a mixture of cameras and audio recorded from a ceiling mounted microphone. Portions of these interviews determined by a clinician to be particularly diagnostic were then segmented and transcribed. Note that in order to publish this data, it had to be de-identified by applying a low-pass filter to regions identified as containing personal identifying information (PII). Pitch information in these regions is still recoverable, but the amplitude levels have been reduced relative to the original signal. Filtering was done with a 10th order Butterworth filter with a passband of 0 to 400 Hz. To avoid abrupt transitions in the resulting waveform, the effect of the filter was gradually faded in and out at the beginning and end of the regions using a ramp of 40 ms.

- *CIR*

  Conversations in Restaurants (CIR) is a collection of informal speech recorded in restaurants that LDC originally produced for the NSF Hearables Challenge, an NSF-sponsored challenge designed to promote the development of algorithms or methods that could improve hearing in a noisy setting. It consists of conversations between 3 and 6 speakers, all LDC or Penn employees, seated at the same table at a restaurant near the University of Pennsylvania campus. Recording sessions were held at lunch time using a rotating list of restaurants exhibiting diverse acoustic environments and typically lasted 60-70 minutes. All recordings were conducted using binaural microphones mounted on either side of one speaker's head. A limited number of regions from one recording were found to contain PII. These regions were de- identified using the same low-pass filtering approach as in ADOS

- *DASS*

  The Digital Archive of Southern Speech, or DASS, is a corpus of interviews (each lasting anywhere from 3 to 13 hours) recorded during the late 60s and 70s in the Gulf Coast region of the United States. It is part of the larger Linguistic Atlas of the Gulf States (LAGS), a long-running project that attempted to preserve the speech of a region encompassing Louisiana, Alabama, Mississippi, and Florida as well as parts of Texas, Tennessee, Arkansas, and Georgia. Each interview was conducted in the field by a trained interviewer, who attempted to elicit conversation about common topics like family, the weather, household articles, agriculture, and social connections. It is distributed by LDC as LDC2012S03 and LDC2016S05. Due to the nature of the interviews, they sometimes contain PII or sensitive materials. All such regions have been replaced by tones of matched duration. Unfortunately, this process does not appear to have been systematic, with the result that the type of tone (pure or complex), power, and frequency differs across the corpus.

- *DCIEM*

  The DCIEM Map Task Corpus (LDC96S38) is a collection of recordings of two-person map tasks recorded for the DCIEM Sleep Deprivation Study. This study was conducted by the Defense and Civil Institute of Environmental Medicine (Department of National Defense, Canada) to evaluate the effect of drugs on performance degradation in sleep deprived individuals. Three drug conditions (Modafinil vs. Amphetamine vs. placebo) were crossed with three sleep conditions (18 hours vs. 48 hours vs. 58 hours awake). During each session, subjects performed a battery of neuropsychological tests (e.g., tracking tasks, time estimation tasks, attention-splitting tasks), questionnaires, and a map task. All audio was recorded via close-talking microphones under quiet conditions.

- *LibriVox*

  LibriVox is a collection of public domain audiobooks read by volunteers from around the world. It consists of more than 10,000 recordings in 96 languages. Portions have previously appeared in the popular LibriSpeech corpus, though care was taken to ensure that DIHARD did not select from this subset.

- *MIXER6*

  Mixer 6 (LDC2013S03) is a large-scale collection of English speech across multiple environments, modalities, degrees of formality, and channels that was conducted at LDC from 2009 through 2010. The collection consists of interviews with 594 native speakers of English spanning 1,425 sessions, each roughly 40-45 minutes in duration. Each session contained multiple components (e.g., informal conversation styled after a sociolinguistic interview or transcript reading) and was captured by a variety of microphones, including lavalier, head-mounted, podium, shotgun, PZM, and array microphones. While the corpus was released without speaker segmentation or transcripts, a portion of the corpus was subsequently transcribed at LDC. DIHARD II draws its selections from this subset.

- *ROAR*

  ROAR is a collection of multiparty (3 to 6 participant) conversations recorded by LDC as part of the DARPA ROAR (Robust Omnipresent Automatic Recognition) project in Fall 2001. While portions of this collection have previously been exposed during the NIST RT evaluations, all DIHARD data comes from previously unexposed meetings. The meetings were recorded at LDC in a purpose built room using a combination of lavalier, head mounted, omnidirectional, PZM, shotgun, podium, and array microphones. For each meeting, a single centrally located distant microphone is provided.

- *RT04*

  RT04 consists of meeting speech released as part of the NIST Spring 2004 Rich Transcription (RT-04S) Meeting Recognition Evaluation development and evaluation sets. This data was later re-released by LDC as LDC2007S11 and LDC2007S12. It consists of recordings of multiparty (3 to 7 participant) meetings held at multiple sites (ICSI, NIST, CMU, and LDC), each with a different microphone setup. For DIHARD, a single channel is distributed for each meeting, corresponding to the RT-04S single distant microphone (SDM) condition. Audio files have been trimmed from the original recordings to the 11 minute scoring regions specified in the RT-04S un-partitioned evaluation map (UEM) files8.

- *SCOTUS*

  SCOTUS is an unpublished LDC corpus consisting of oral arguments from the 2001 term of the U.S. Supreme Court. The recordings were transcribed and manually word-aligned as part of the OYEZ project, then forced aligned and QCed at LDC.

- *SEEDLingS*

SEEDLingS is a corpus of child speech collected at the University of Rochester. Excerpts from day-long recordings conducted in the home were selected, then segmented and transcribed by LDC.

- *SLX*

  SLX (LDC2003T15) is a corpus of sociolinguistic interviews conducted in the 1960s and 1970s by Bill Labov and his students. The interview subjects range in age from 15 to 81 and represent a diverse sampling of ethnicities, backgrounds, and dialects (e.g., southern Amercian English, African American English, northern England, and Scotland). While the recordings have good sound quality for field recordings (especially from that era), they were collected in a range of environments ranging from noisy homes (e.g., small children running around in the background) to public parks to gas stations.

- *VAST*

  The Video Annotation for Speech Technologies (VAST) corpus is a (mostly) unexposed collection of approximately 2,900 hours of web videos (e.g., YouTube and Vimeo) intended for development and evaluation of speech technologies; in particular, speech activity detection (SAD), diarization, language identification (LID), speaker identification (SID), and speech recognition (STT). Collection emphasized videos where people are talking with a particular emphasis on videos where the speakers spoke primarily English, Mandarin, and Arabic, which comprise the bulk of the corpus9. Portions of this corpus have been exposed previously as part of the NIST 2017 Speech Analytic Technologies Evaluation, the NIST 2017 Language Recognition Evaluation, NIST 2018 Speaker Recognition Evaluation, and DIHARD I.

- *YouthPoint*

  YouthPoint is an unpublished LDC corpus consisting of episodes of YouthPoint, a late 1970s radio program run by students at the University of Pennsylvania. The show had an interview format similar to shows such as NPR's Fresh Air and consisted of interviews between University of Pennsylvania students and various popular figures. The recordings were conducted in a studio on open reel tapes and later digitized and transcribed at LDC.

Bibliography

[1] 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*, no. CONF. IEEE Signal Processing Society, 2011.

[2] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "First dihard challenge evaluation plan," 2018.

[3] 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.

[4] S. E. Tranter and D. A. Reynolds, "An overview of automatic speaker diarization systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557–1565, Sep. 2006.

[5] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling." in *ISMIR*, vol. 270, 2000, pp. 1–11.

[6] S. Chen, P. S. Gopalakrishnan, and I. M. Watson, "Speaker , environment and channel change detection and clustering via the bayesian information criterion," 1998.

[7] J. Patino, H. Delgado, R. Yin, H. Bredin, C. Barras, and N. W. Evans, "Odessa at albayzin speaker diarization challenge 2018." in *IberSPEECH*, 2018, pp. 211–215.

[8] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[9] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digit. Signal Process.*, vol. 10, no. 1, pp. 19–41, Jan. 2000. [Online]. Available: http://dx.doi.org/10.1006/dspr.1999.0361

[10] S. Kullback, *Information theory and statistics.* Courier Corporation, 1997.

[11] C. Barras, X. Zhu, S. Meignier, and J.-L. Gauvain, "Improving speaker diarization," 2004.

[12] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech communication*, vol. 52, no. 1, pp. 12–40, 2010.

[13] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[14] 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, May 2011.

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

[16] 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.

[17] T. Stafylakis, J. Rohdin, O. Plchot, P. Mizera, and L. Burget, "Self-supervised speaker embeddings," *arXiv preprint arXiv:1904.03486*, 2019.

[18] 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.

[19] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 531–542.

[20] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[21] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "Second dihard challenge evaluation plan," 2019.

[22] ——, "The second dihard diarization challenge: Dataset, task, and baselines," *arXiv preprint arXiv:1906.07839*, 2019.

[23] S. J. Young and S. Young, *The HTK hidden Markov model toolkit: Design and philosophy*. University of Cambridge, Department of Engineering Cambridge, England, 1993.

[24] "Hungarian algorithm." [Online]. Available: http://www.hungarianalgorithm.com

[25] L. Hamers *et al.*, "Similarity measures in scientometric research: The jaccard index versus salton's cosine formula." *Information Processing and Management*, vol. 25, no. 3, pp. 315–18, 1989.

[26] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe *et al.*, "Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge," in *Proc. Interspeech*, 2018, pp. 2808–2812.

[27] "md-eval.pl." [Online]. Available: https://web.archive.org/web/20061001115045/http://www.nist.gov/speech/tests/rt/rt2006/spring/code/md-eval-v21.pl

[28] N. Ryant, "dscore." [Online]. Available: https://github.com/nryant/dscore

[29] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[30] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[31] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

[32] "June ldc newsletter." [Online]. Available: https://www.ldc.upenn.edu/communications/newsletter/june-2019-newsletter

[33] "Dihard ii unofficial repository." [Online]. Available: http://archive.is/Ha8x3

[34] G. Sell and D. Garcia-Romero, "Diarization resegmentation in the factor analysis subspace," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4794–4798.

[35] "Kaldi models webpage." [Online]. Available: http://kaldi-asr.org/models.html

[36] Z. Huang, L. P. García-Perera, J. Villalba, D. Povey, and N. Dehak, "Jhu diarization system description." in *IberSPEECH*, 2018, pp. 236–239.