NANYANG
TECHNOLOGICAL
UNIVERSITY

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

APPLICATION OF DEEP LEARNING IN THE FIELD OF SPEAKER DIARIZATION

PRANAV V GRANDHI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN BITS PILANI HYDERABAD CAMPUS

2022

Contents

Abstract

In recent years, the application of deep learning in the field of speaker diarization has shown a lot of promise. Recent advances in computer hardware and software have enabled the application of machine learning in various fields. Specifically in the field of speaker diarization, a lot of progress has been made in recent times. Speaker Diarization is the task of identifying the question of 'which person spoke when?'. Given an audio input the task is to label the audio with speaker identity for every timestamp. This also includes multiple people speaking at once. Speaker Diarization algorithms have often been merged with speech recognition algorithms and are often complementary to each other. Applications of Speaker Diarization are umpteen in number in today's world from automatic transcript generation to diarizing meetings and lectures. Diarization helps in enhancing the readability of a speech transcription. With the increase in the number of meeting recordings, phone calls and broadcasts, speaker diarization received a lot of attention by the speech community. Speaker Diarization consists of multiple smaller sub segments like speech activity detection, speech segmentation, embedding of the speakers, and finally clustering. In this paper, we review the recent advancements in speaker diarization as well as discuss ways of improving the current state of the art methods. The main goal of this thesis is to first set up and run the baseline model which is a combination of x-vectors to extract the speech embeddings, followed by using a Bi-LSTM model to generate the similarity matrices and then Spectral clustering to obtain the final results. After successfully running the baseline model, the goal is to apply knowledge distillation to obtain the x-vectors from Pytorch and not kaldi. The ultimate goal of the project is to come up with a solution that will reduce the Diarization Error Rate and build an efficient Speaker Diarization model.

Keywords: Diarization, deep learning.

Acknowledgement

I would like to express my deep gratitude to Dr. Chng Eng Siong, my thesis supervisor and Prof. Aruna Malapati, my thesis co-supervisor, for their patient guidance, enthusiastic support, useful critique of this research work and for their assistance in keeping my progress on schedule. I would also like to thank my advisors Melvin Wong, Liu Chnyu and Lim Zhi Hao for their useful guidance without which this project would not be possible. I would also like to thank BITS Pilani for providing me with the wonderful opportunity to carry out this thesis. The project has given me a keen sense and interest towards academic research and inspired me to work on interesting research that benefits society.

Acronyms

Acronyms goes here.

Symbols

Symbols goes here.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The word ‘Diarize’ in its most literal sense is storing an event in a diary. It often means taking notes. Speaker Diarization addresses the question of ‘who spoke when’. [1] The inputs are typically audio files of multiple people speaking at once or at different time intervals. The expected output is labeling of a speaker for each time interval. Thus, the whole process of diarization consists of speech segments with a label. Speaker diarization also accounts for multiple speakers and detecting segments in the audio where no speaker is speaking. The most important modules in speaker diarization are as follows: Front-end Processing Speech Activity Detection Segmentation Speaker Embedding Clustering Post Processing Each of the above shown segments will be discussed in detail in the later segments.

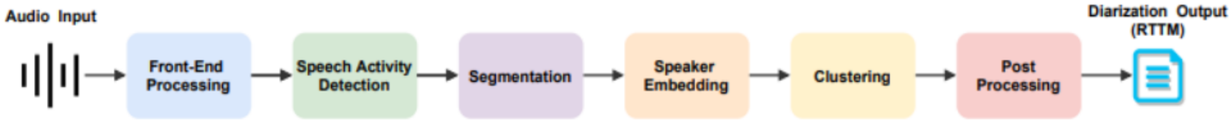

Figure 1.1: Speaker diarization.

[2]

1.2 Motivation

Speaker Diarization is a canonical sub-task in many of the other Speech Recognition Problems such as Meeting Transcription, Audio Indexing, and Conversation AI. [3] With the rapid growth of Deep Learning, solving the Speaker Diarization problem has become more feasible. It can enhance the readability of an automatic speech transcription by structuring the audio stream into speaker turns and, when used together with speaker recognition systems, by providing the speaker’s true identity. Speaker diarization is a combination of speaker segmentation and speaker clustering. The first aims at finding speaker change points in an audio stream. The second aims at grouping together speech segments on the basis of speaker characteristics. This knowledge is very useful in automating tasks.

Chapter 2

Literature Review

2.1 Front-end Processing

Front end processing is the first step in most of the speaker diarization processes. It is done to enhance the quality of the dataset and clean it so that unnecessary noise, reverberation is not present. In some cases, speech enhancement is also applied. These methods will be described in detail in this section. In general, the input audio signal can be represented as a mixture of the source audio of the individual person, the room impulse response, and the additional noise. This is represented as follows: Here, x denotes the whole audio source. It can be represented as the sum of all 'K' speakers, where each speaker is represented as 'i'. 'H' represents the room's impulse response, s is the speaker representation and 'n' is the noise. In each of the above three segments 'f' and 't' are the frequency and the time period.

2.1.1 Speech Enhancement and De-noising

This method is used to reduce the speech component of the speech segment. It has shown great improvement to reduce the error rate. LSTM (Long-short Term memory) based approaches have been successful to reduce noise to a large extent. [4] This method is popular for single channel audio input files. This means the audio is captured from a single source. In other types of audio inputs such as multichannel input, different methods are used. One popular approach for multiple channels is variance distortionless response (MVDR) beamforming.

2.1.2 Dereverberation

Dereverberation is the technique of reducing reverberation in the audio input that could cause bias in the model. The techniques used for dereverberation are mainly statistical signal processing methods. WPE (Weighted Prediction Error) is one of the popular techniques in this area. [5] This technique has consistently shown good results. In most cases, the reduce in the error wasn't large like the de noising methods, but it has consistently shown to reduce error rate in almost all datasets.

2.1.3 Speech Seperation

Another important feature for most speaker diarization techniques is that it should be consistent even when multiple speakers speak at once. Hence, for multiple concurrent speakers, multichannel inputs with techniques such as beamforming has shown great results. However, single channel data has often given worse results when speaker overlap is significant.

2.2 Speech Activity Detection

Speech Activity Detection (SAD) is the process that distinguishes the non speech segments to the speech segments in the input dataset. This process is also known as voice activity detection (VAD). Non-speech segments are segments of the audio input that do not correspond to any human speech. It can be background noise or any non-human sounds. This is a pivotal step in most speaker diarization pipelines. If this module is working incorrectly, it has shown to affect the final outcome very drastically. The mistakes often propagate till the end. The process of speech activity detection comprises two main parts. The first is feature extraction. In this step, features are extracted such as

- Pitch
- Signal Energy
- MFCC (Mel-frequency Cepstrum)

The second part is the classifier. The main job of the classifier is to use the input features that were extracted in the previous step and use them to distinguish speech segments from the non speech segments. The final output of the classifier will consist of an input audio frame with a label corresponding to speech segment or non speech segment. Several classifiers have shown success in the past such as

- Gaussian Mixture Models (GMMs) [6]
- Hidden Markov Model (HMMs) [7]
- Convolutional neural networks (CNNs) [8]
- Long short Term Memory (LSTM) [9]
- Multilayer Perceptron (MLP) [10]

Speech Activity Detection is an important part of the whole pipeline. This is because the most popular error metric called Diarization Error Rate (DER) has components that affect negatively if speech segments are predicted for speakers when the segment is actually a non speech segment.

2.3 Speech Separation

Speech Segmentation is the next process in the speaker diarization pipeline after the Speech Activity Detection. The main goal of the speech segmentation module is to break the input dataset into various segments. There are mainly two different kinds of segmentation. They are :

- Segmentation by speaker change point detection
- Uniform Segmentation

$$\begin{aligned} H_0 &: \mathbf{x}_1 \cdots \mathbf{x}_N \sim \mathcal{N}(\mu, \Sigma), \\ H_1 &: \mathbf{x}_1 \cdots \mathbf{x}_i \sim \mathcal{N}(\mu_1, \Sigma_1), \\ &\quad \mathbf{x}_{i+1} \cdots \mathbf{x}_N \sim \mathcal{N}(\mu_2, \Sigma_2), \end{aligned}$$

Figure 2.1: Speaker representation.

The above two categories have their own advantages and disadvantages. Segmentation by speaker change detection is the process of segmenting the input audio into segments that define speaker change. Hence, each segment is of varying length depending on how long each speaker spoke for. This method is not very successful when multiple speakers speak at once and there is a lot of overlap. However, this method has been very successful and was the standard for speech segmentation. The process of choosing the segments is by using various metrics between two hypotheses. Here, let H0 denote that the left side segment and the right segment belong to the same speaker. Let H1 assume that the left and right speaker segments belong to different speakers. Several metrics are used to test these hypotheses. In each of the cases, both H0 and H1 are considered to follow a Gaussian distribution with a particular mean and covariance. Several metrics were used to quantify the likelihood of both the hypotheses. Some of them are as follows:

- Generalized Likelihood Ratio (GLR) [11]
- Kullback Leibler Distance [12]
- Bayesian Information Criterion (BIC) [13]

BIC has shown to produce great results. The second type of speech segmentation is called uniform segmentation. The main difference between this method and the previous is that the speech segments are of uniform lengths. This method faces some issues due to the variability of the segment length. However, with the recent advancements in speaker embeddings such as x-vectors, uniform segmentation gained popularity.

2.4 Speaker Embedding

After the speech segmentation is done, the next step in the speaker diarization pipeline is the speaker embeddings. The main goal of the speaker embedding is the process of converting each segment into a readable vector format and calculating the similarity between each of the segments. Therefore, this step consists of two main processes. The first is to produce the speaker vector representations for each segment. The second is to produce a similarity matrix between each of the segments. After this the similarity matrices are taken as input in the clustering phase where the final output of the diarization pipeline is obtained.

2.4.1 Speaker Representations

Speaker representation is one of the most important steps in speaker diarization. It is the process that creates vectors that will be used to create the similarity matrices. Over the years there have been many different kinds of speaker representations. Some of the important ones are as follows:

- Joint Factor Analysis [14]
- X-vector [15]
- I-vector [16]
- D-vector [17]

In this paper, we will discuss x-vectors, i-vectors and the d-vectors as these are some of the most widely used speaker representations. X-vectors and d-vectors have been introduced recently and are the most popular neural network based implementations.

2.4.2 I-Vector

These vectors were introduced as an improvement to the Joint Factor Analysis. This is because JFA does not account for the speaker variability efficiently. Hence, a proposition was made to model the GMM supervector as a concatenation of mean GMM vectors: Where m is a speaker and channel-independent supervector, usually modeled by a UMB, T is a low-rank matrix, and w is a standard normal distributed vector, also called the “identity vector”.

2.4.3 X-vectors

X-vectors have shown great results in speaker diarization in recent years. With the advent of deep learning, speaker representations such as x-vectors are one of the most popular forms of speaker representations. The dense neural network architecture shown below is the architecture to produce the x-vectors. The first five layers are the TDNN (Time Delay Neural Network) layers. These ;ayers aggregate the temporal data. Finally, this is forwarded to the statistical pooling layer which aggregates the data along the time domain while computing the means and the standard deviation. These are concatenated and sent to the segment 6 layer. This final layer is a fully connected layer. This final layer is called the x-vectors. These are essentially DNNs that are trained to discriminate between speakers. It maps variable length utterances to fixed dimensional embeddings called as x-vectors. These TDNN layers operate at frame levels. The final layer goes through statistical pooling. As the layers progress, the context width also increases. This in turn reduces the complexity of the TDNN model and hence the final complexity is only twice that of a normal DNN. It also simultaneously helps in capturing long term correlations.

2.4.4 D-vectors

D-vectors [18], are one of the most prominent forms of speaker diarization. The input for generating these vectors are the stacked filter bank features which have the context frames. These are then passed along multiple fully connected layers. These layers are trained using cross entropy loss. The vectors that are obtained from the last layer are called the d-vectors.

2.5 Clustering

Clustering is the final step in the process of speaker diarization. There are two main types of clustering. They are as follows:

- Online Clustering: Once the segment is available, the label is immediately emitted. Future segments are not visited to give the predictions.
- Offline Clustering: In offline clustering, the predictions or labels are generated only after all the speaker embeddings are generated. This is usually better than online clustering because additional contextual information is available.

For speaker diarization, the most commonly used clustering methods are Agglomerative Hierarchical Clustering and Spectral Clustering.

2.5.1 Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering (AHC) [19] is a standard clustering approach for tasks when the clusters are expected to form a hierarchy which is represented as a Dendrogram in the literature. The idea is to iteratively merge clusters starting at the lowest level with singleton clusters, thereby forming the hierarchy. At each step, the most similar clusters are merged. For Speaker Diarization, AHC is stopped when the similarity threshold is reached or when a target number of clusters exist. The latter is especially useful in scenarios when the number of clusters is known.

2.5.2 Spectral Clustering

Spectral Clustering [20] is a method that is widely used in many of the clustering modules for speaker diarization. It is based on graph theory. The main goal is to use the eigenvectors of the adjacency matrix of the parent graph and construct graph based partitions. This technique reduces multidimensional data into clusters in the smaller dimension. It can be easily implemented using the normal linear algebra modules. The goal is to generate the clusters given the input of a similarity matrix. Essentially the data has to be connected for spectral clustering. The goal is to make cuts in the connected graph to get the connected clusters of same speakers across various segments.

2.6 Evaluation Metrics

There are several evaluation metrics for speaker diarization. Some of them are as follows:

- Diarization Error Rate (DER)
- Jaccard Error Rate (JER)
- Equal Error Rate (EER)

2.6.1 Diarization Error Rate

DER (Diarization Error Rate) [21] is one of the most widely used metrics for speaker diarization. DER has three main components :

- False Alarm Time : accounts for the time the system overestimates the number of speakers. It is the percentage of scored time that a hypothesized speaker is labeled as a non-speech in the reference.
- Missed Speech : It is the percentage of scored time that a hypothesized non-speech segment corresponds to a reference speaker segment. The Miss Time for the amount of the system underestimates the number of speakers, and Speaker Error Time or Confusion accounts for the time the system's output speaker does not match the reference.
- Speaker Error: It is the percentage of scored time that a speaker ID is assigned to the wrong speaker.

$$DER = \frac{T_{missed} + T_{FA} + T_{spkr\_err}}{T_{spkr}}$$

Figure 2.2: Diarization Error Rate

2.6.2 Jaccard Error Rate

Jaccard Error Rate (JER) is a metric that was inspired by the Jaccard Index. It is the sum of missed speech plus false alarm divided by the total speech time. TFA is the total system speaker time not classified as the reference speaker, and TMISS is the total reference speaker time not associated with the system speaker.

$$\mathrm{JER}_{\mathrm{ref}} = \frac{T_{\mathrm{FA}} + T_{\mathrm{MISS}}}{T_{\mathrm{SPEECH}}}$$

Figure 2.3: Jaccard Error Rate

2.7 Datasets

2.7.1 Callhome

CALLHOME American English Speech is a telephone conversation dataset developed by Linguistic Data Consortium (LDC). It consists of 120 unscripted, 30 minutes long telephone conversations between native English speakers. The dataset was compiled in 1997 when 8 kHz was the de facto frequency for such datasets. Though the newer datasets such as VoxCeleb have been sampled at 16 kHz, CALLHOME remains a very important dataset in Diarization tasks. The current pipeline uses the CALLHOME dataset to reproduce the currently existing results.

2.7.2 Free ST Chinese Mandarin Corpus (SLR38)

This is a much smaller dataset compared to the others. Hence it is often used as a dataset to first get the proof of concept. It is only 8.2GB. The SLR38 dataset is curated by Surgingtech and has 855 speakers, with each speaker having exactly 120 utterances. It was recorded in silence, in an indoor environment using cellphones and therefore is a good model to be used for diarization as other larger datasets have been prepared under similar conditions.

2.7.3 DIHARD

DIHARD dataset is the first dataset that was made using real world scenarios. It is a dataset that most accurately depicts the real world. Unlike the CALLHOME dataset which was made under controlled environments, this is better because we can test on real world data.

2.7.4 VoxCeleb

This is a dataset that was curated from YouTube. It is one of the largest datasets. It is available in two forms VoxCeleb1 and VoxCeleb2. The audio utterances are 16kHz. The utterances of speakers are available in both audio and video. When combined, there are over seven thousand speakers, one million utterances, and two thousand hours of recordings. Many of the baseline models are trained on this dataset and this dataset is considered as a standard.

Chapter 3

Approach

3.1 BaseLine Model

Currently the CALLHOME dataset is being used to test the baseline model. The current pipeline consists of mainly three different components.

- Speaker Embedding: The x-vectors are chosen for the speaker representations
- BackEnd: BiLSTM model to generate the similarity matrices
- Clustering: Spectral Clustering

The speaker embedding is the x-vectors. The generation of x-vectors has been discussed in the earlier segments. In the pipeline, the CALLHOME dataset is used. We use a pretrained model (in kaldi) to generate the x-vectors in the CALLHOME dataset. For generating the similarity matrix which is the second step in the pipeline, we use a BiLSTM model. Usually, for this module, we can use either use cosine similarity, PLDA or LSTM.

The architecture is two 256-dimensional Bi-LSTM layers followed by a 64-dimensional FCNN layer and a 1-dimensional sigmoid output layer. It takes as input the n 128-dimensional x-vector embeddings and predicts the Similarity Matrix row-wise. Using the batch size as 'n' in our case was very memory intensive. Hence we performed the training in batches. This mini-batching makes it practical to run in low memory GPU environments.

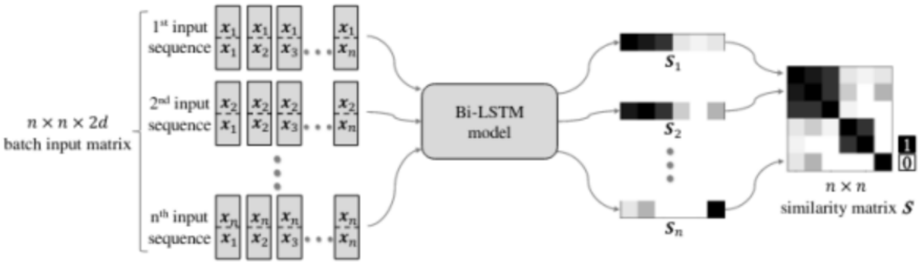

Figure 3.1: Bi-LSTM model for generating the similarity martrix

Finally after the similarity matrix is generated, we have used spectral clustering to obtain the final results. The best cparam for which the DER was lowest was 0.99473. The CALLHOME dataset was used to run the baseline model locally. The x-vectors were extracted using a pretrained model. After this, the BiLSTM model was run locally for 100 epochs using batch training. This was followed by spectral clustering. The best DER obtained was 7.3

3.2 Speaker Verification

The final goal of the project is to reduce the final error for speaker diarization. One of the most important steps in speaker diarization is the speaker embeddings. In traditional speaker diarization systems, the main focus in the speaker embedding area is the embeddings of the speakers. However, in this case, the focus was mostly on background noise. Identifying what kind of background noise is very important because we can precisely remove the unnecessary parts if we know what the background noise is classified as. In most speaker diarization systems the Speech Activity Detection portion also known as SAD, is the module which is responsible for the process of segregating the audio from the unwanted or background noise. If this module is not very good, speaker embedders which are only trained to work on pure speech data will not work very well. The task of speaker verification which was most popularized by the famous “OK Google” verification is the task of verifying that a particular part of the speech was spoken by that speaker. The most famous application of this is the process of voice matching. There are mainly two major categories of speaker verification.

- The first category is called text dependent speaker verification also known as TD-SV. In this there is a phonetic constraint on the transcript of both enrolment and verification utterance part of the speech.
- The second category is called text independent speaker verification which is also known as TI-SV. In TI-SV there are not any phonetic constraints. For our case, we will be discussing TI-SV.

Hence the secondary goal of this thesis is to perform speaker verification on the data that is not speaker speech. We use various kinds of background noise and other unwanted noise to detect and see if the current standard of speaker verification models works well on non-speech data.

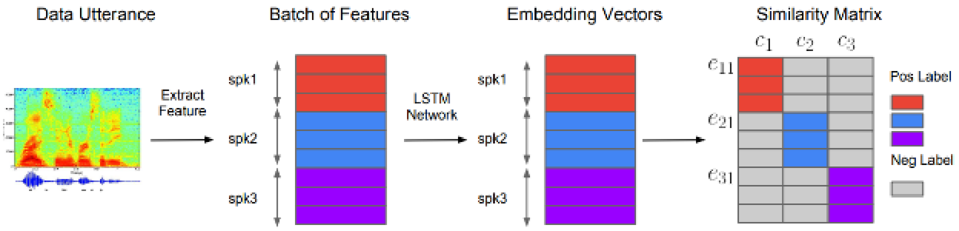

Figure 3.2: Speaker Embeddings

3.3 Resemblyzer

One of the most famous speaker verification models is a paper titled “Generalized End to End loss for speaker verification”. The model that is described in this paper is the foundation for Google’s “Ok Google” speaker verification functionality. Unlike the original tuple-based end to end loss this paper introduces generalized end to end loss which from now on will be referred to as GE2E loss. For GE2E, the training method utilizes a large number of speaker utterances at once. To be more precise, N*M utterances are used to train. In this ‘N’ depicts the total number of speakers. ‘M’ depicts the number of utterances for each speaker. Therefore, each position in the matrix determines the utterance for that corresponding speaker. This matrix is then fed into the neural network. This network is an LSTM layer. The embedding is the normalized version of the final layer. The centroid of each speaker is then calculated as the centroid of all the utterances for that speaker. After this, a similarity matrix is generated. This matrix is the size of N*N. Each element of the matrix denotes the similarity of the normalized embeddings to the corresponding centroid. The similarity is calculated as follows:

$$\mathbf{S}_{ji,k} = w \cdot \cos(\mathbf{e}_{ji}, \mathbf{c}_k) + b,$$

Figure 3.3: Similarity index

Here, w and b are parameters that are learnable. Eij represents the normalized speaker embeddings. Ck is the corresponding centroid for that speaker. The major difference between both the originally defined tuple loss and the GE2E loss is that the original loss used only a linear method where the distance between the embedding vector and the centroid was minimized only for one centroid. In the case of GE2E, the goal is to minimize the distance between the centroid of current speaker and maximize the distance to other speakers.

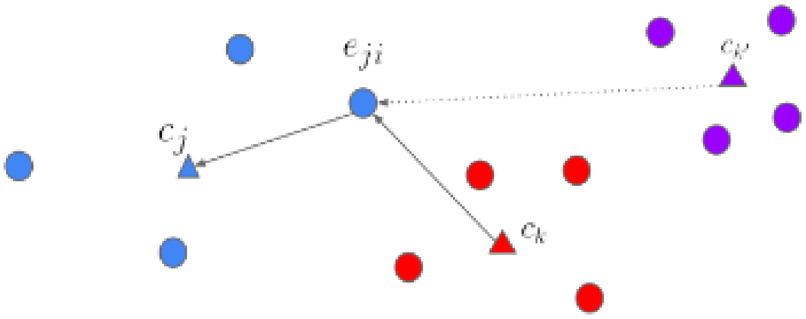

Figure 3.4: GE2E Loss

3.4 UMAP and T-SNE projections

UMAP (Uniform Manifold Approximation and Projection) is a technique that performs dimensionality reduction. It is a scalable algorithm that can be applied to real world data to visually see higher dimensional data on a two dimensional plane. T-SNE (T-distributed Stochastic Neighbor Embedding) is another similar algorithm that performs dimensionality reduction. The main goal of both these algorithms is to visualize the higher dimensional data on a two dimensional graph. The design is mainly to preserve the local structure and group neighboring data points. The goal is to visually see the heterogeneity of the data. Although they show the separation between the clusters, these algorithms do not guarantee that the distances between the clusters are preserved correctly.

3.5 Datasets used for experiments

LibriSpeech : This is a dataset that is based on the LibriVox's public domain of audiobooks. The main goal of this dataset is to help in the training and testing of ASR (automatic speech recognition) systems. A subset of 10 speakers were used to perform testing on the resemblyzer.

TIMIT dataset: This is a speaker dataset that consists of 6300 sentences in which 630 speakers each speak 10 sentences. The speakers are from different regions meaning they have different dialects. There are a total of eight dialects which are further broken down by sex.

MUSAN dataset : Musan Dataset is a dataset that contains three different kinds of background noise. The categories are :

- Music
- Noise
- Speech

The main goal of this dataset is to see whether the current model can properly distinguish between speech and non speech data. Later we have made our own dataset with multiple different categories of non speech data. This dataset is used to train speech embedders to achieve better results in speaker diarization.

Chapter 4

Test and Experiments

4.1 LibriSpeech Dataset

The following are the results for models trained on LibreSpeech Data. The UMAP projections show us how well the embeddings are seperated.

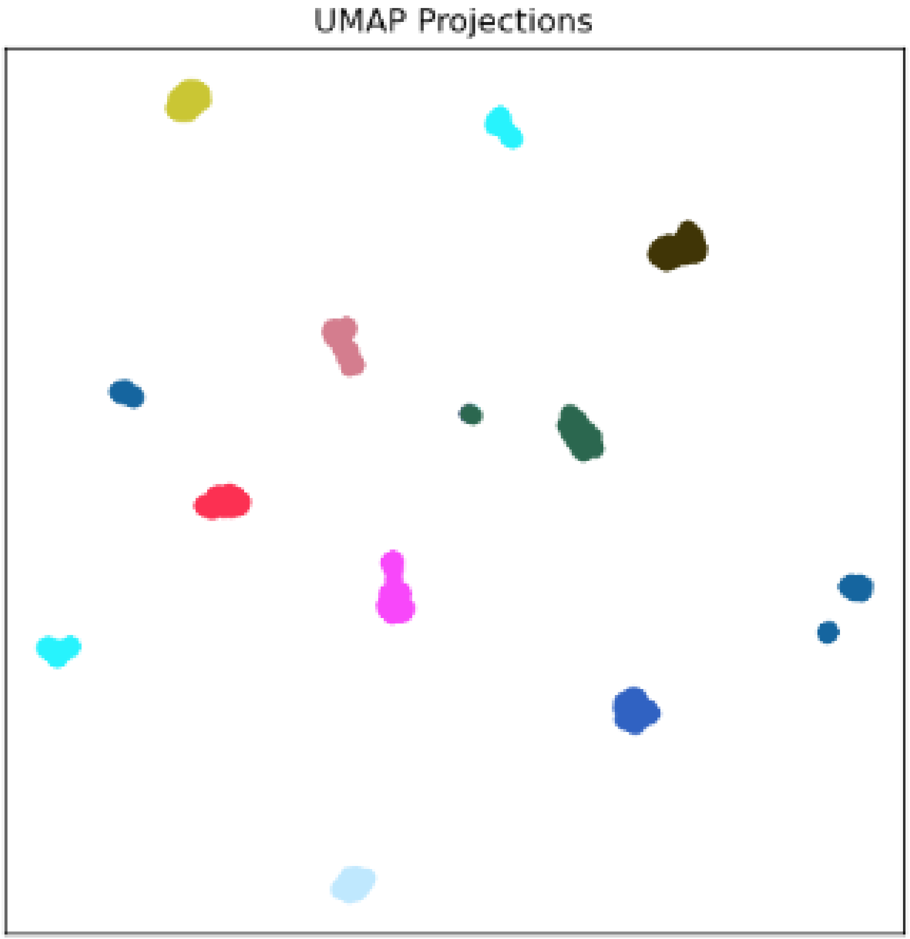

Figure 4.1: Resemblyzer trained and tested on LibriSpeech Data

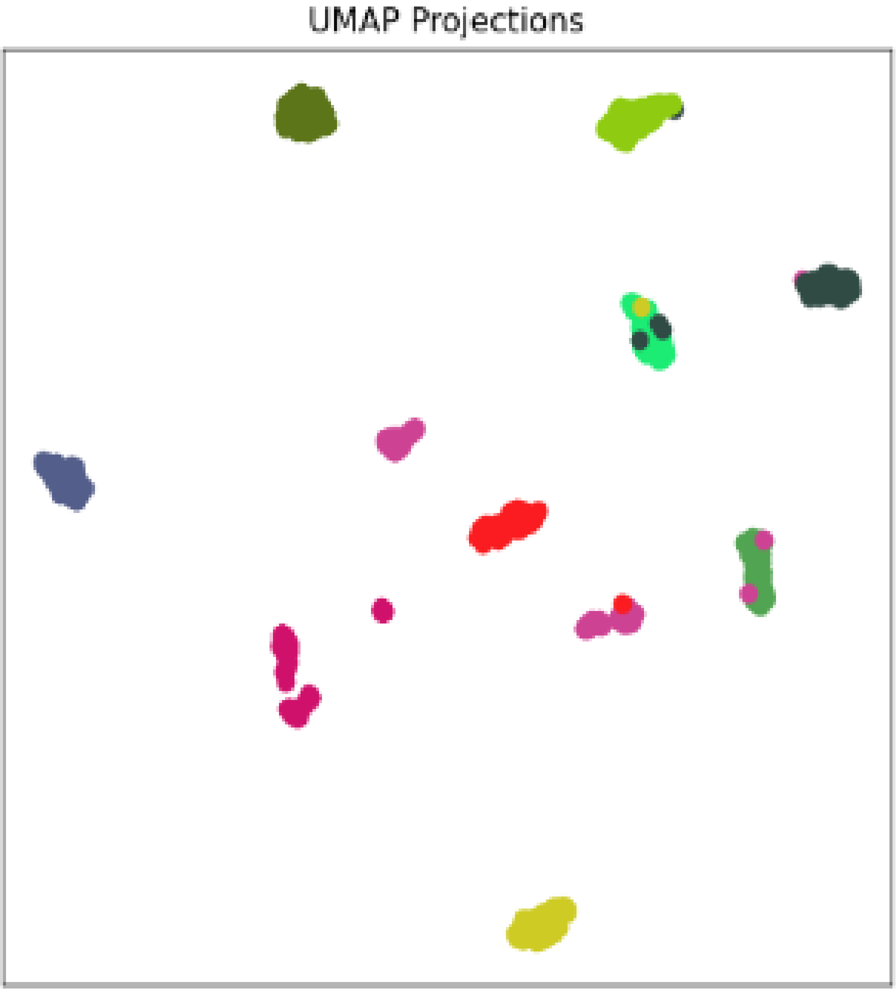

Figure 4.2: X-vector trained and tested on LibriSpeech Data

Further experiments were done using the resemblyzer embedder. It was trained on three different kinds of datasets and tested on LibriSpeech Dataset. The following are the results:

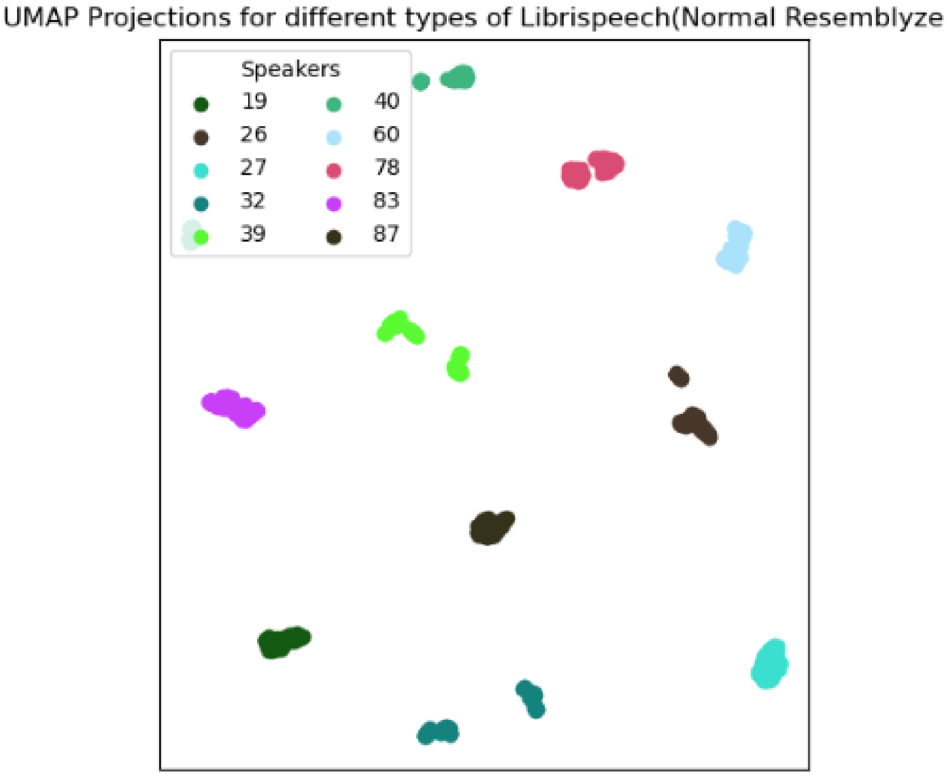

Figure 4.3: Resemblyzer trained and tested on LibriSpeech Data

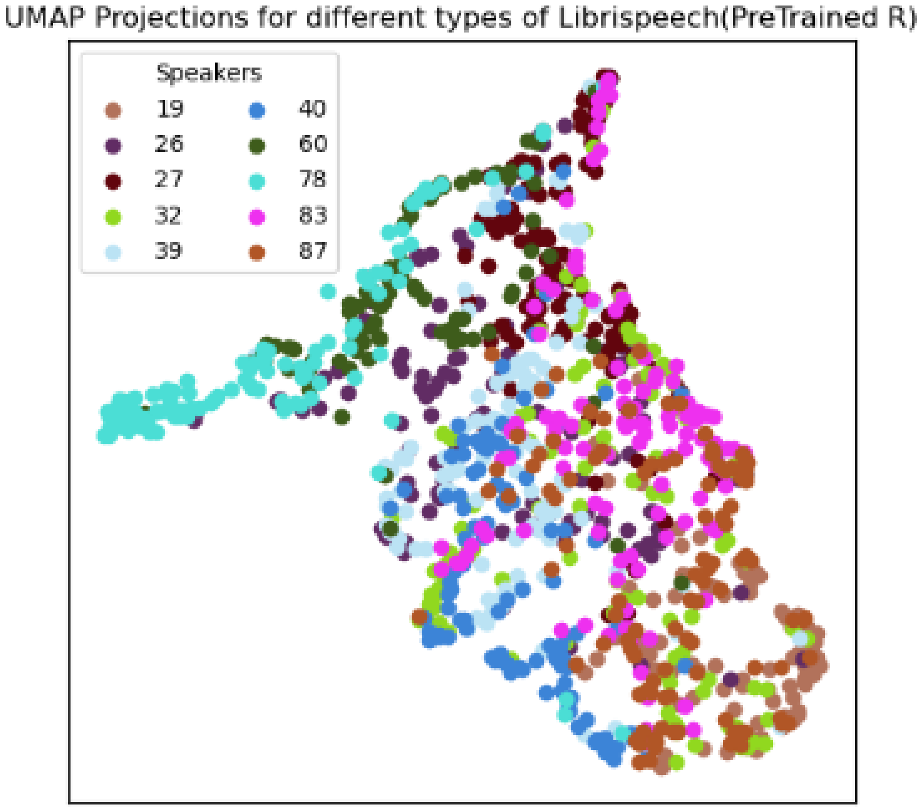

Figure 4.4: Resemblyzer trained on TIMIT and tested on LibriSpeech Data

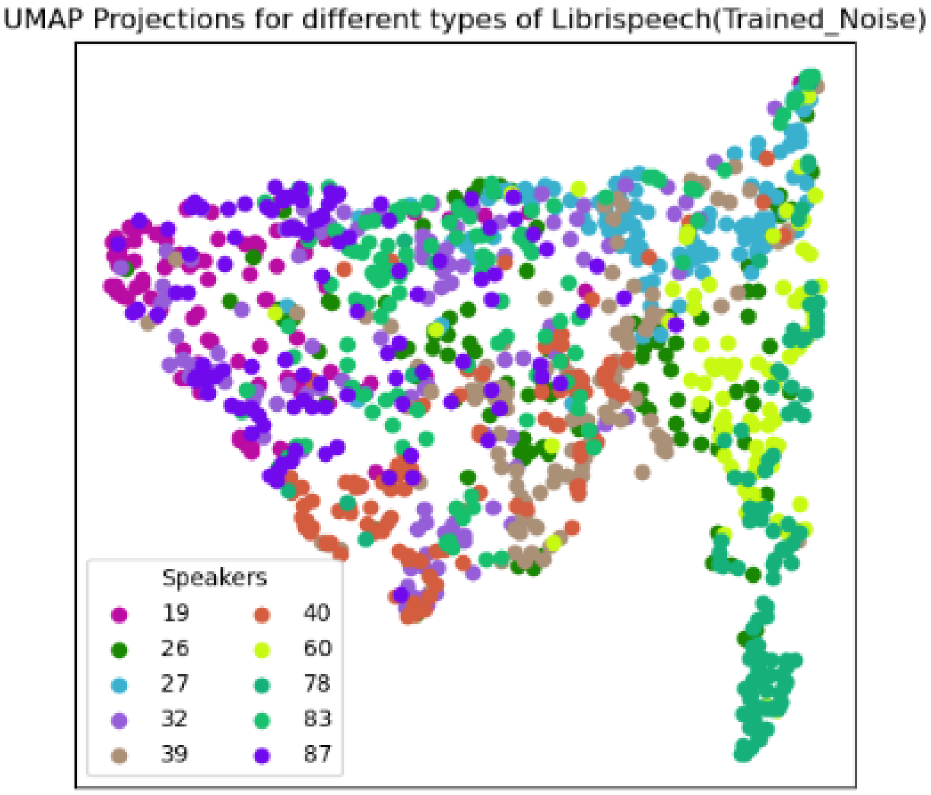

Figure 4.5: Resemblyzer trained on custom noise dataset and tested on LibriSpeech Data

4.2 Musan Dataset

The following is the UMAP projection when resemblyzer is used on the MUSAN dataset

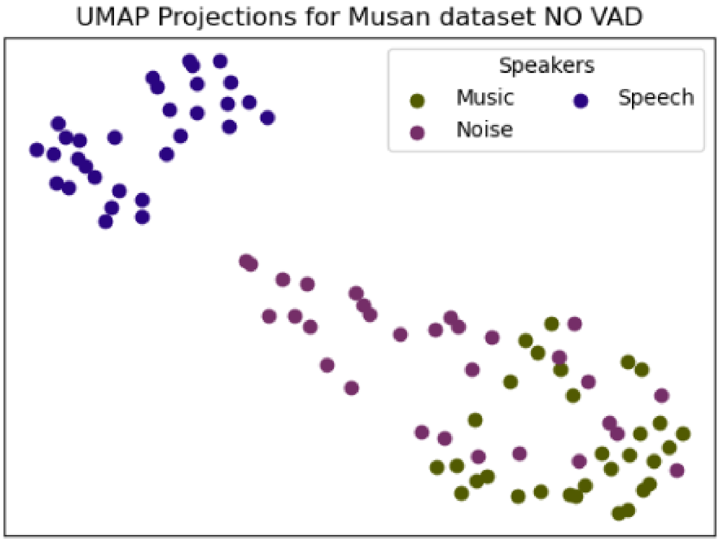

Figure 4.6: Resemblyzer tested on MUSAN dataset

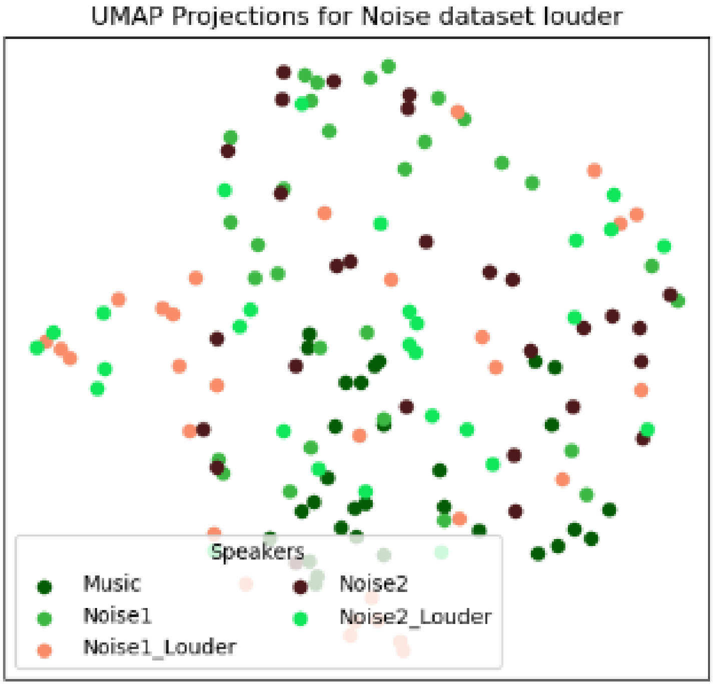

Figure 4.7: Resemblyzer tested on custom MUSAN dataset with louder noise

4.3 Noise Dataset

A custom noise dataset has been curated with nine categories as explained in the previous chapters. The training was done on various datasets and it was tested on the custom noise dataset. The following are the results:

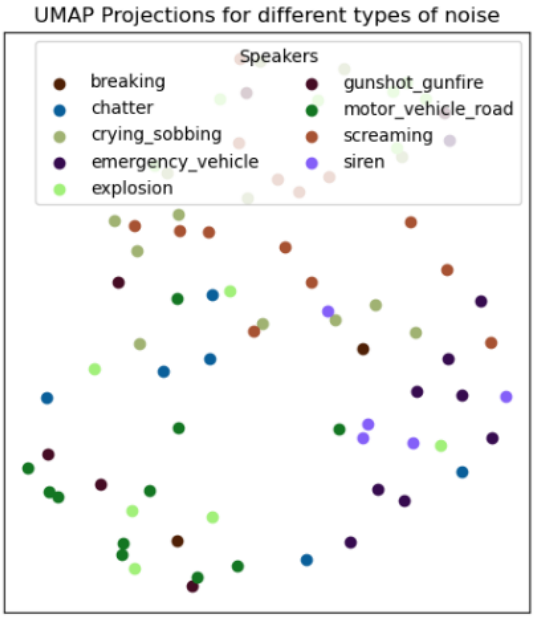

Figure 4.8: Resemblyzer trained on LibriSpeech and tested on Noise

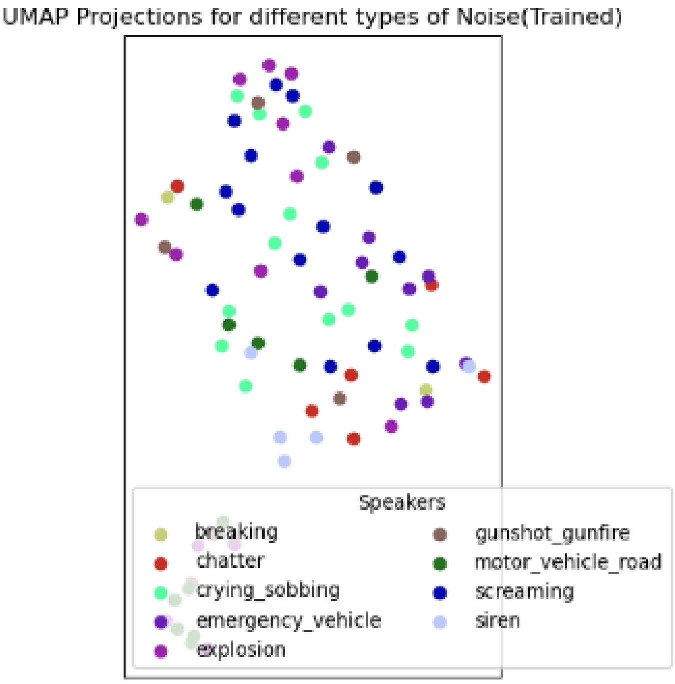

Figure 4.9: Resemblyzer trained on TIMIT and tested on Noise

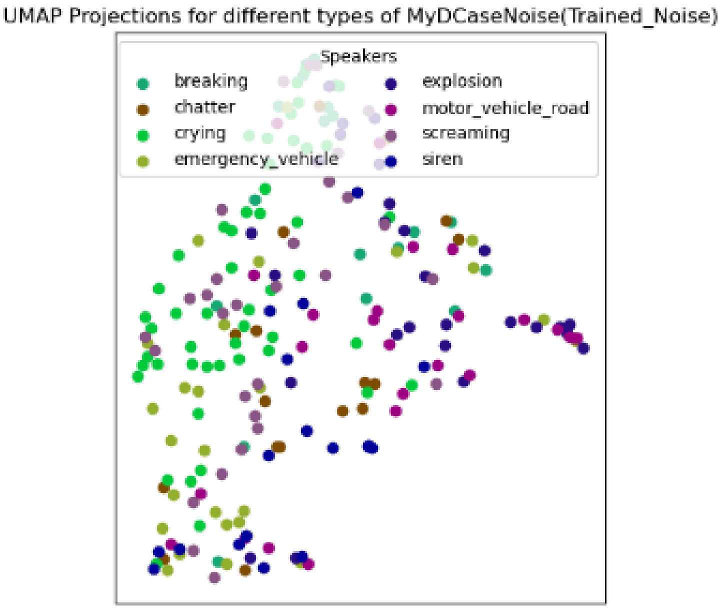

Figure 4.10: Resemblyzer trained on custom noise and tested on Noise

Chapter 5

Conclusion and Recommendations

5.1 Future Work

The current pipeline is disjoint in the sense that half of it is done using kaldi while the other half is done using Pytorch. The x-vectors are generated through kaldi, but the similarity matrices through pytorch. The current state-of-the-art in Deep Learning is evolving in and around Python, particularly, because it is open source and makes prototyping easy. Further, the large community maintained frameworks such as PyTorch and Tensorflow are available to make Deep Learning computations easy and extensible. If both the embedding and the similarity matrices can be trained in the same language it is useful for everyone. Experiments have been performed to see how well the current speaker embedder models will perform on non speech data. This is especially important because if we can properly identify what category of non-speech data and make the appropriate embeddings this will help reduce the overall loss for the speaker diarization. As of now, the model trained on custom noise dataset is not very well in the UMAP clustering. Hence the next step is to tweak hyperparameters to reach convergence on the noise dataset. Generating good embedders for background noise can be very helpful for speaker diarization.

Bibliography

[1] SE Tranter, K Yu, DA Reynolds, G Evermann, DY Kim, and PC Woodland. An investigation into the the interactions between speaker diarisation systems and automatic speech transcription. 2003.

[2] Tae Jin Park, Naoyuki Kanda, Dimitrios Dimitriadis, Kyu J Han, Shinji Watanabe, and Shrikanth Narayanan. A review of speaker diarization: Recent advances with deep learning. *Computer Speech & Language*, 72:101317, 2022.

[3] Dan Ellis. Speech separation in humans and machines. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 1–13, 2005.

[4] Tian Gao, Jun Du, Li-Rong Dai, and Chin-Hui Lee. Densely connected progressive learning for lstm-based speech enhancement. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5054–5058. IEEE, 2018.

[5] Tomohiro Nakatani, Takuya Yoshioka, Keisuke Kinoshita, Masato Miyoshi, and Biing-Hwang Juang. Speech dereverberation based on variance-normalized delayed linear prediction. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(7):1717–1731, 2010.

[6] Jongseo Sohn, Nam Soo Kim, and Wonyong Sung. A statistical model-based voice activity detection. *IEEE signal processing letters*, 6(1):1–3, 1999.

[7] Thilo Pfau, Daniel PW Ellis, and Andreas Stolcke. Multispeaker speech activity detection for the icsi meeting recorder. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU'01.*, pages 107–110. IEEE, 2001.

[8] Samuel Thomas, Sriram Ganapathy, George Saon, and Hagen Soltau. Analyzing convolutional neural networks for speech activity detection in mismatched acoustic conditions. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2519–2523. IEEE, 2014.

[9] Gregory Gelly and Jean-Luc Gauvain. Optimization of rnn-based speech activity detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):646–656, 2017.

[10] Neville Ryant, Mark Liberman, and Jiahong Yuan. Speech activity detection on youtube using deep neural networks. In *INTERSPEECH*, pages 728–731. Lyon, France, 2013.

[11] Herbert Gish, Man-Hung Siu, and Jan Robin Rohlicek. Segregation of speakers for speech recognition and speaker identification. In *icassp*, volume 91, pages 873–876, 1991.

[12] Matthew A Siegler, Uday Jain, Bhiksha Raj, and Richard M Stern. Automatic segmentation, classification and clustering of broadcast news audio. In *Proc. DARPA speech recognition workshop*, volume 1997, 1997.

[13] Perrine Delacourt and Christian J Wellekens. Distbic: A speaker-based segmentation for audio data indexing. *Speech communication*, 32(1-2):111–126, 2000.

[14] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1435–1447, 2007.

[15] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5329–5333. IEEE, 2018.

[16] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59. IEEE, 2013.

[17] Rama Doddipatla, Norbert Braunschweiler, and Ranniery Maia. Speaker adaptation in dnn-based speech synthesis using d-vectors. In *INTERSPEECH*, pages 3404–3408, 2017.

[18] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4052–4056. IEEE, 2014.

[19] Kyu J Han, Samuel Kim, and Shrikanth S Narayanan. Strategies to improve the robustness of agglomerative hierarchical clustering under data source variation for speaker diarization. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1590–1601, 2008.

[20] Huazhong Ning, Ming Liu, Hao Tang, and Thomas S Huang. A spectral clustering approach to speaker diarization. In *Ninth International Conference on Spoken Language Processing*, 2006.

[21] Jonathan G Fiscus, Jerome Ajot, Martial Michel, and John S Garofolo. The rich transcription 2006 spring meeting recognition evaluation. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 309–322. Springer, 2006.

Appendix A

(Code Here)

Appendix B

(Code Here)