EE698R: Speaker Diarization and Transfer Learning

Shivam Kumar
170668

Yash Mittal
170818

Prateek Varshney
170494

Abstract

In this paper we describe our model for the speaker diarization problem and explain how one can leverage Transfer Learning to quickly learn a model at the expense of negligible performance loss as compared to a fully trained one. Given the input utterances and their speaker identity labels, we extract embeddings from short audio segments and use these embeddings to segregate the speaker segments within the input source. Building upon this model, we have focused on transfer learning and manually adapting over various datasets so as to make our model more generic. We have also focused on improving the DER along with experimenting with different embedding generation networks.

1 Introduction

Speaker Diarization refers to partitioning an input audio stream into segments based on the speaker identity, i.e., it tells us *"who spoke when"* in a multi-speaker setting. For example, we want to know how many speakers are there and when did each speaker start and end speaking. Speaker Turn Analysis, Multi-media information retrieval, etc. are just some of the domains where Speaker Diarization is applied extensively.

Most existing Speaker Diarization Systems consist of multiple relative independent components, namely:

1. Speech Segmentation Module: Inputs audio into short segments which are of uniform size (assumed to have a single speaker).
2. Voice Activity Detection Module: Detects the presence or absence of human speech.
3. Audio Embedding Extraction Module: Extracts specific features such as i-vectors [4][3], MFCCs [2], Mel Spectrum from the segmented sections.
4. Clustering Module: Determines the number of speakers and assigns speaker identities to clusters of extracted audio embeddings.
5. Re-segmentation Module [5]: Refines the Clustering output by imposing additional constraints.

In this report, we provide details for our speaker diarization models. We also leverage Transfer Learning on top of an LSTM-based text-independent speech embedding model, which is then passed to a parametric clustering algorithm to obtain a speaker diarization system.

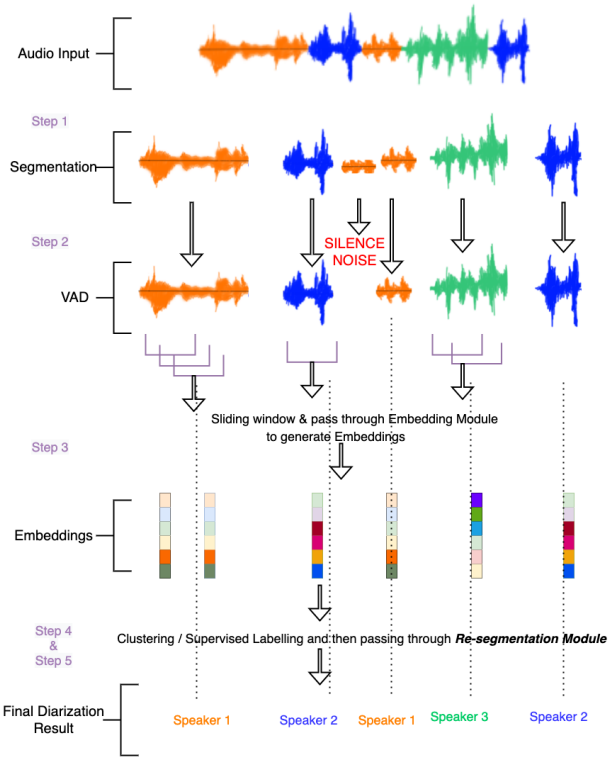


Figure 1: Approach

2 Data Description

We are currently using two datasets from different domains for our project:

- **AMI dataset**: The AMI Meeting Corpus is a multi-modal data set and consists of meeting recordings. Specifically, we use the subset of meetings recorded in English using three different rooms with acoustic properties. The number of speakers in each subset varies from 1-4 (labelled A,B,C and D) with the words spoken by each speaker during the meeting stored as $(..., StartTime, EndTime, SpeakerId, ...)$ stored in XML files which we use to segment the input audio wav file. Currently we are using three different meeting subsets of AMI dataset namely ES, IS and TS.

Blur σ	Train DER	Test DER
0.25	0.1733	0.6143
0.5	**0.1464**	**0.5575**
0.75	0.1825	0.5613
1.00	0.4617	0.5644

(a) Resemblyzer + Spectral Clustering

Model	Train DER	Test DER
AMI-CNN	0.32	0.37
AMI-LSTM	0.287	0.304
Hindi-BiLSTM	0.114	0.238

(b) Embedding Architectures

Model	Train DER	Test DER
TL Variant 1	0.41	0.45
TL Variant 2	0.41	0.29
TL Variant 3	0.42	0.28

(c) Transfer Learning Variation

Table 1: DER analysis of various models

- **Custom made Hindi-English dataset**: This dataset was created through obtaining various Hindi news discussion. The number of speakers varies from 4-7 (labelled as different alphabets). The input audio were Mono channel wav file with 16 bit sample width. They were annotated in CSV format with structure $(AudioFileName, StartingTime, Duration, Speaker\_ID)$.

3 Proposed Approach

Wan et. al. developed an LSTM network [1] to create speaker embeddings for both text-dependent and text-independent speaker verification [7]. As a baseline, we use a pre-trained instance of their model (trained on fixed-length segments extracted from a large corpus) as the Embedding module for our Speaker Diarization system. We then experiment with a *CNN architecture*, a *LSTM Model* (on AMI-Corpus Dataset, henceforth called AMI-CNN and AMI-LSTM), a *BiLSTM Model* (on Hindi-English Dataset, henceforth called Hindi-English-BiLSTM) and 3 different variants of a *BiLSTM model* (using *Transfer Learning*, henceforth referred to as *BiLSTM-TF*) with *Triplet Loss Function* as an Embedding module. In our pipeline (Figure 1), the input audio signals are segmented into uniform small chunks using the ground truth labels (each chunk containing approximately only one speaker), a VAD is used to select only those segments which contain speech, then raw level features (MFCC and log-Mel Spectrum Features) are extracted from each frame as the network input for the embedding model (BiLSTM-TF/LSTM/CNN). The embedding model then produces speaker embeddings for each input sequence. Thus this reduces arbitrary length audio streams into a sequence of fixed size embeddings. We then apply a suitable clustering algorithm to these embeddings to determine both the total number of unique speakers in the entire audio input as well as assign each audio chunk to a specific speaker.

3.1 Preprocessing

The audio wav files consists of various speakers with corresponding text annotation file denoting the timestamps of all the distinct speakers along with their speaker id. It is loaded into the system using Librosa library. We then extract the respective time stamp by parsing the annotation file, split the wav file into small chunks and concatenate them according to their speaker id. In this process, we also remove the noise if the consecutive chunks doesn't belong to annotation file.

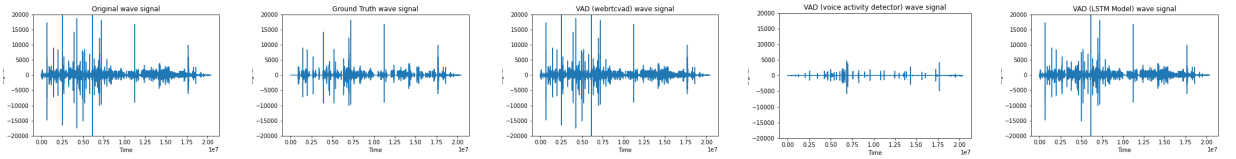
For the testing phase, as we do not know which speaker said when (i.e. what parts of the wav file actually contain utterances), we split the audio into 1 second long chunks and run the denoising + low level feature extraction as before. We use the following routines for each of the models:

- **AMI-CNN Model (self-implemented):** we use a self-implemented Mel-log spectrum and MFCC feature extractor as well as a denoiser to remove the silence parts and speech noise
- **AMI-LSTM Model (Resemblyzer):** we use the log-melspectrum of the wav chunks as the input vectors (features) to the Embedding module.
- **5 LSTM based Models (self-implemented):** For the AMI-LSTM and all the 3 variants of BiLSTM-TL we use Librosa's MFCC and MFCC-Delta feature extractors. For the Hindi-English-BiLSTM model we use both Librosa's MFCC and MFCC-Delta feature extractors as well as Pyannote's Embedding generation function.

These raw features of the split wav chunks are then used as the input for the embedding module to generate the embeddings for each speech segment, after the corresponding speech segments have been chosen by VAD.

3.2 Voice activity detection (VAD)

For detecting presence or absence of human speech we use Voice activity detection (VAD) technique. In our project we incorporated VAD by three different methods.



(a) Complete Audio (b) Ground Truth (c) WebRTC-VAD (d) Log Mel Spectrum (e) LSTM based Model

Figure 2: Comparing output of Different implementation of VAD (non-speech segments assigned 0)

- **WebRTC-VAD:** A GMM based voice activity detector which uses GMM models of speech and non-speech sounds with input features as log energies of six frequency bands between 80Hz-4000Hz.
- **Voice Activity Detector (self implemented):** Uses normalised energies (computed using log Mel Spectrum and MFCC Features) and a user-input threshold to determine bands of voice activity.
- **LSTM based Model (self implemented):** Trained on input files (similar to Embedding module), except that it assigns each speech/non-speech segment as label 1/0 respectively and treats VAD as a binary classification problem.

3.3 Embeddings

After removing noise we create the embedding vector using the Embedding Module (Resemblyzer[8]/AMI-CNN/AMI-LSTM/Hindi-English-BiLSTM/BiLSTM-TF). These embedding vectors are such that they give the similarity between the two speakers based on the distance or cosine similarity between them.

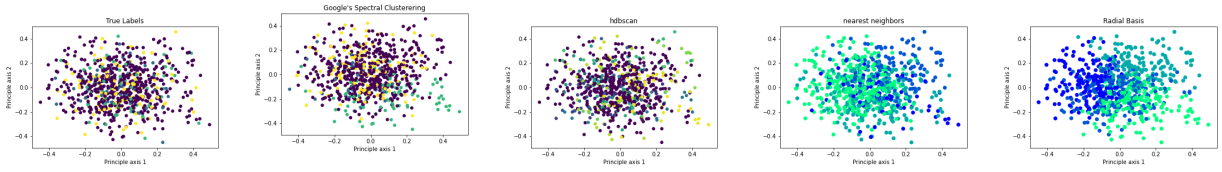
- **Resemblyzer:** we obtain a summary vector of 256 values (embedding) which summarises the characteristics of the voice spoken in the audio chunk.
- **AMI-CNN (self coded):** We use the model architecture: input (160 x 64) $\rightarrow$ 2D Convolution (3,3) $\rightarrow$ Relu activation $\rightarrow$ 2D Maxpool (2,2)) $\times 2 \rightarrow$ Dense Layer (32) $\rightarrow$ Relu activation $\rightarrow$ Dense Layer (16). Therefore, the output embedding is a 16 floating values long vector.
- **AMI-LSTM (self coded):** We use three LSTM Layers (768 nodes large) followed by 2 Dense Layers with tanh activation and Dropout of 0.01. The 256 long output vector is normalised using l2 distance metric.
- **Hindi-English-BiLSTM (self coded):** We use two BiLSTM Layers (128 nodes large) followed by 2 Time Distributed Dense Layers (32 nodes large) with tanh activation and Dropout of 0.01.

3.4 Clustering Algorithm

Next, we cluster the embeddings obtained to obtain the speaker Labels. Since we are working on predefined Datasets with Transfer Learning as the variant, we choose Offline Clustering Methods to assign the speaker labels to the input embeddings. Note that Offline Clustering Methods usually have more hyperparameter choices ('knobs') which is exactly what we need to fine-tune over the given dataset having pre-trained over another dataset (Transfer Learning). Since the Clustering output decides both the number of speakers and segment-wise labels, the overall Diarization performance depends critically on the clustering module. We therefore work with a number of Clustering Algorithms:

- **Sklearn's Spectral Clustering Algorithm**: It is useful when a measure of the center and spread of the cluster is not a suitable description of the complete cluster. This method requires the exact number of expected speakers.
- **K-Means++ Clustering (self coded)**: Starts with smarter initialization of the centroids and improves the quality of the clustering as compared to Vanilla K-Means. But also requires the exact number of expected speakers beforehand.
- **Google's Spectral Clustering Algorithm**: Performs eigen-decomposition over the Affinity Matrix and uses K-Means++ on the features in the eigen-vector space. Allows for min/max number of expected cluster as an input.
- **Hierarchical Agglomerative Clustering**: **HDBSCAN** is an extension of DBSCAN which converts it into a hierarchical clustering algorithm. For our model we set min_cluster_size to 4 and tune other parameters manually.
- **MeanShift Clustering:** It is very much similar to K-means clustering apart from one major fact that it does not require specifying the number of clusters. Due to this, we choose this algorithm for clustering the embeddings.

The Figure 3 shows the cluster outputs for the AMI-CNN model for each clustering algorithm



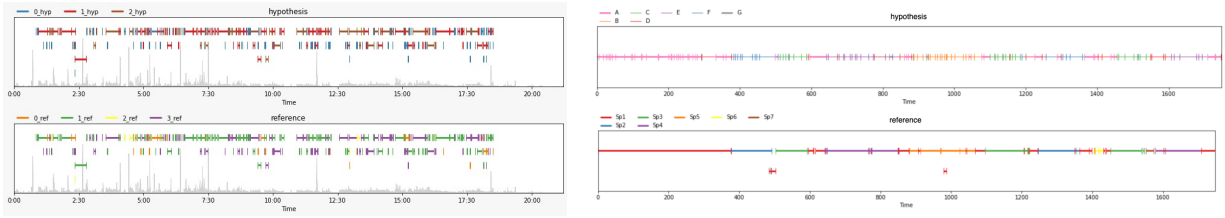
(a) True Labels (b) Google's Spectral Clustering (c) HDBSCAN (d) Nearest Neighbors (e) Radial Basis

Figure 3: Clustering of embedding vector using different algorithm

3.5 DER Calculation

We then compare it against the given speaker id (ground truth) to get results using Diarization Error Rate. To calculate the DER we invoke the Pyannote's metric library.

Figure 4 provides an easy visualisation of the output diarization, referred to as Hypothesis, ("who spoke when") against the ground truth, referred to as Reference. Diarization Error Rate (**DER**) results are shown in Table 1.



(a) Train (DER: 0.114)

(b) Test (DER: 0.24)

Figure 4: Plots of Hypothesis & Reference for BiLSTM model

4 Our Project Variant: Transfer Learning

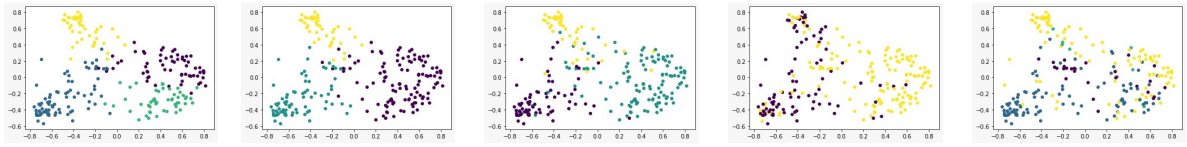
Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem [6]. The intuition being that a model trained on a large and general enough dataset will effectively serve as a generic model of the entire particular domain and these pre-learned feature maps will avoid training from scratch a large model on a large dataset. We integrate Transfer Learning by first pre-training the BiLSTM model on the Hindi-English Dataset and then fine-tuning over the AMI-Corpus Dataset via the following (**self-implemented**) variants:

1. **Variant 1: Feature Extraction** We pass data sample's raw features (MFCC) through the Hindi-English-BiLSTM Model to obtain "refined" features since the representations learned by the pre-trained network will extract "meaningful" features. The "refined dataset" obtained this way is then used to train a new Embedding Module (AMI-LSTM) from scratch. This is similar to passing the dataset through a sequence of 2 models aligned one after the other.
2. **Variant 2:** Here we combine the above 2 models into one. The idea being that the BiLSTM layers of the pre-trained network already contain features generically useful for creating embeddings. However the dense layers are specific to the original dataset. Therefore, here we freeze the weights of the BiLSTM layers of the Hindi-English-BiLSTM Model, remove and replace the TimeDistributed Dense Layers with one LSTM + Simple Dense Layers and retrain the model using MFCC features of the AMI-Corpus Dataset, thereby enabling only the training of the top layers.
3. **Variant 3:** This is very much like Variant 2 except that we unfreeze the BiLSTM layers as well, i.e., we train the "pre-trained" model (after replacing the Dense Layers) end to end on the current dataset and finetune it accordingly.

We observe that Variant 2 seems to perform best and freezing the BiLSTM layers enables us to save significantly on the training time as compared to Variants 1 and 3. Diarization Error Rate (**DER**) results are shown in Table 1c.

4.1 HyperParameters "Knob" Tuning for Domain Adaptation

Our approach has many hyperparameter "knobs" which one can change to finetune and improve the performance of the Transfer Learning Models Obtained above. For example, during our experiments, we found that the performance of the spectral clustering in terms of Number of Speakers identified was sensitive to the value of Sigma (σ) used in the Gaussian Blur operation used (Figure 5). Here we show how the output embedding clustering changes on different values of Sigma. So after training on one dataset we can finetune Sigma manually to better adapt our Model on different dataset and obtain better results. Other such knobs are include leaf size (number of points in leaf node of the tree) alpha (distance scaling parameter), cluster selection epsilon(distance threshold), etc. in HDBSCAN, number of Dense Layers and Dense Layer Nodes, Number segmentation window size, MFCC normalization flag, etc.



(a) $\sigma = 0.25$ (b) $\sigma = 0.5$ (c) $\sigma = 0.75$ (d) $\sigma = 1$ (e) Ground Truth

Figure 5: Spectral Plots with their respective Gaussian Blur Sigma Values ("Knob" Tuning)

5 Conclusion and Future Work

In this report, we provide the results of our experiments for a Speaker Diarization with a Transfer Learning Flavour. We present our findings on various VADs, Pre-processing Algorithms, Embedding Modules and Clustering Algorithms. We also empirically demonstrate the potential of Transfer Learning in improving the computation time for training an end to end Speaker Diarization System at very negligible accuracy cost, and present cases where it even surpasses non-Transfer Learning Model performances. Our Speaker Diarization Demo Files on a real recording from a TV and our Transfer Learning variant demo videos can be viewed at Demo_Part1.mp4 and Demo _Part2.mp4 respectively.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[2] P. Kenny, D. Reynolds, and F. Castaldo. "Diarization of Telephone Conversations Using Factor Analysis". In: *IEEE Journal of Selected Topics in Signal Processing* 4.6 (2010), pp. 1059–1070. DOI: 10.1109/JSTSP.2010.2081790.

[3] G. Sell and D. Garcia-Romero. "Speaker diarization with plda i-vector scoring and unsupervised calibration". In: *2014 IEEE Spoken Language Technology Workshop (SLT)*. 2014, pp. 413–417. DOI: 10.1109/SLT.2014.7078610.

[4] M. Senoussaoui et al. "A Study of the Cosine Distance-Based Mean Shift for Telephone Speech Diarization". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1 (2014), pp. 217–227. DOI: 10.1109/TASLP.2013.2285474.

[5] 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)*. 2015, pp. 4794–4798. DOI: 10.1109/ICASSP.2015.7178881.

[6] Stevo Bozinovski. "Reminder of the First Paper on Transfer Learning in Neural Networks, 1976". In: *Informatica* 44 (Sept. 2020). DOI: 10.31449/inf.v44i3.2828.

[7] Li Wan et al. *Generalized End-to-End Loss for Speaker Verification*. 2020. arXiv: 1710.10467 [eess.AS].

[8] Resemble-Ai. *https://github.com/resemble-ai/Resemblyzer*.