Discriminative Neural Clustering

Technical Milestone Report

James Thompson, jhrt2
Emmanuel College

January 2022

1 Abstract

This project aims to improve work done recently in the Speech Research Group on Discriminative Neural Clustering (DNC), a novel supervised clustering method applied to speaker diarisation. The proposed improvements will reduce the restrictions on input data that the original model required and provide better performance. Progress so far has seen the implementation of a new framework that augments data from the AMI corpus 'on-the-fly', ie. on each iteration of training, reducing storage costs. The new implementation will allow more additions to be made with relative ease, such as a transition to window-level clustering (rather than utterance-level) and the ability to handle overlapping speakers.

2 Introduction

2.1 Speaker Diarisation

Speaker diarisation is the task of determining 'who spoke when' in a conversation [1]. The typical pipeline is given in Figure 1. The input is a continuous stream of audio which first undergoes Voice Activity Detection (VAD), where regions containing speech are labelled. Next, Change Point Detection (CPD) identifies the points where the speaker changes, giving a sequence of speaker-homogeneous segments. Each segment is converted to a series of speaker embeddings, in this case d-vectors [2] which are the output of a hidden neural network layer for classifying speakers. These 32-dimensional d-vectors encapsulate information about the speaker's voice over a window of 200 frames (2s), with each window shifted 10ms from the start of the previous window. In the current implementation, the mean is then calculated for all the d-vectors from one segment, giving one d-vector per segment. Clustering is then performed on the d-vectors to group segments close in the embedding space. Each cluster corresponds to a speaker label and so the final output is a sequence of speaker labels for the inputted audio stream. The focus of this project is the clustering phase.

2.2 Spectral Clustering

A commonly used unsupervised clustering algorithm, which is used as a baseline in this project, is spectral clustering [3]. First an affinity matrix, A, is produced which gives the cosine similarity between speaker embeddings. After applying some refinement operations,

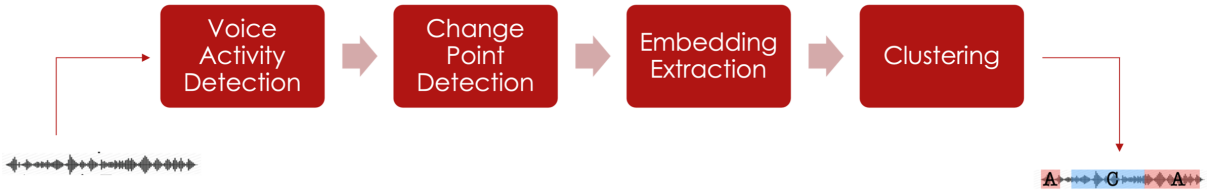

Figure 1: Speaker diarisation pipeline

dimensionality reduction is performed via eigen-decomposition with the number of clusters given by:

$$\tilde{k} = \arg\max_k \frac{\lambda_k}{\lambda_{k+1}}$$

where the eigenvalues, λ_k, are in descending order. Finally, k-means clustering is done on new embeddings formed by taking the eigenvectors, v_k, corresponding to the $\tilde{k}$ highest eigenvalues and replacing the ith segment embedding with the corresponding dimension for each eigenvector, $e_i = [v_{1i}, ..., v_{\tilde{k}i}]$.

2.3 Discriminative Neural Clustering

Discriminative Neural Clustering (DNC) is a supervised clustering algorithm introduced by Li et al. [4] based on the Transformer model [5]. Rather than being given a pre-defined distance measure (in the way spectral clustering uses cosine similarity), DNC learns how to determine the similarity of embeddings by training on meetings from the AMI corpus [6]. The probability of a speaker label, y_i, which is the cluster associated with segment, $\boldsymbol{x_i}$, is conditioned on all previous labels, $y_{0:i-1}$ and the entire input sequence $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top$:

$$P(y_{1:N}|\boldsymbol{X}) = \prod_{i=1}^{N} P(y_i|y_{0:i-1}, \boldsymbol{X})$$

The encoder-decoder model is defined as follows:

$$\boldsymbol{H} = \textsc{Encoder}(\boldsymbol{X})$$

$$y_i = \textsc{Decoder}(y_{0:i-1}, \boldsymbol{H})$$

where $\boldsymbol{H}$ is a hidden representation of $\boldsymbol{X}$ which captures the similarity between segments from the same speaker. The encoder consists of a multi-head self-attention layer followed by a feed-forward layer, and the decoder consists of a multi-head self-attention layer, a multi-head source-attention layer and a feed-forward layer. These are defined as follows where $\text{MHA}(\tilde{\boldsymbol{X}})$ is a Multi-Head Attention layer acting on the output of the previous layer ($\boldsymbol{X}$ for the first layer):

$$\text{MHA}(\tilde{\boldsymbol{X}}) = \text{concatenate}(\textsc{Head}_1(\tilde{\boldsymbol{X}}), ..., \textsc{Head}_H(\tilde{\boldsymbol{X}}))\boldsymbol{W_o}$$

$$\textsc{Head}_h(\tilde{\boldsymbol{X}}) = \begin{cases} \textsc{Attention}(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}}) & \text{for self-attention} \\ \textsc{Attention}(\tilde{\boldsymbol{X}}, \boldsymbol{H}) & \text{for source-attention} \end{cases}$$

$$\textsc{Attention}(\boldsymbol{\Phi}, \boldsymbol{\Psi}) = \text{softmax}\left(\frac{(\boldsymbol{\Phi W_q})(\boldsymbol{\Phi W}_k)^\top}{\sqrt{D}}\right)(\boldsymbol{\Psi W}_v)$$

D is the dimension of the embeddings and the parameters to be learnt are in the $\boldsymbol{W}$ projection matrices.

2.4 Data Augmentation

There are 147 meetings in the training dataset of the AMI corpus however significantly more data is required to train the DNC model. Therefore, three augmentation techniques are used:

Sub-sequence randomisation takes a meeting and truncates it to get a shorter meeting. The speaker labels may change depending on if there is a different first speaker (labelled '1').

Input vectors randomisation first samples a speaker label sequence, $y_{1:N}$. Then each cluster is assigned a new speaker identity. This can be from across all meetings (called global randomisation) or restricted to the same meeting as the speaker label sequence (meeting randomisation). Finally, each segment, $\boldsymbol{x_i}$, is sampled from the set of d-vectors associated with the corresponding speaker identity.

Diaconis augmentation makes use of the fact that d-vectors are L_2-normalised and so all lie on the unit hypersphere. It is performed on top of the other methods and rotates an entire meeting to a different region of the hypersphere, preserving the distances between d-vectors [7].

2.5 Planned Improvements

Several improvements to the original DNC implementation are proposed for this project. First, the input format is to be changed to allow further developments in the future. Originally, meetings were input as a series of separate speaker-homogeneous segments, however the new implementation is to take in a sequence of d-vectors for an entire meeting. While at first, segmentation data will be provided in another file, in the future this new input format will make it easier to incorporate VAD and CPD into the system.

Currently there is a large storage cost associated with saving the augmented data for the model to work on. This will be solved by introducing on-the-fly augmentation where a mini-batch of augmented data is produced on each iteration in training, and discarded after use, rather than producing a large batch of augmented data beforehand.

Window-level clustering is the next addition where speaker labels will be assigned to each window-level d-vector rather than averaging d-vectors for a whole segment. This is another reason for the new input format and is necessary if VAD and CPD are to be imported.

A longer-term aim is to enable the system to identify two speakers talking at the same time, something the original implementation could not handle. A stepping-stone towards this is to allow multiple-channel input. At the moment, the audio streams from multiple microphones are combined into one channel using Beamforming [8]. Creating two Beamform channels pointing in opposite directions would provide more information to the model, improving performance in the single speaker case and when there are overlapping speakers.

3 Current progress and results

Of the planned improvements to the original model, the project has got as far as having a working implementation of on-the-fly augmentation. This is promising as the aim was to be at this stage by the end of Michaelmas Term.[1]

3.1 Input format

The first stage was to write new functions for loading the meeting data from *.scp*, *.ark* and *.rttm* files containing d-vectors and segmentation information. Four dictionaries are then produced: the first two are used for global and meeting input vectors randomisation respectively while the others are used for sub-sequence randomisation. The global d-vector dictionary has speaker identity as key and a list of all d-vectors associated with that speaker from across all meetings as its value. The meeting d-vector dictionary is similar but has a d-vector dictionary for each meeting. The other two dictionaries contain the sequences of d-vectors (segmented meetings dictionary) and speaker labels (segmented speakers dictionary) for each meeting.

3.1.1 Spectral Clustering

Spectral clustering was performed on the data loaded using the new functions to check that everything was working as expected by comparing performance to the original implementation. At first, the accuracy (the percentage of correctly labelled segments) was lower than anticipated. This unearthed a problem with the new data files that had been provided: the d-vector extraction had been trained using data normalised at the segment level whereas the data I was using had been normalised at the meeting level. This discrepancy was causing clustering to be less effective than expected. After changing the data files to be normalised at the segment level, an accuracy of 63% was achieved which matched that from the original implementation.

3.2 Data Augmentation

New functions that could deal with the new data loading formatting were written for each type of data augmentation. As an interim stage, a function was written to produce a batch of augmented data and store it in *.json*, *.ark* and *.scp* files in the same format as the original DNC. This would allow spectral clustering and DNC to be run on the new augmented data to check it was working as expected before moving to on-the-fly augmentation. Running spectral clustering on the augmented data gave accuracies in the region 60-65%, similar enough to the result for real data to suggest that the augmented data is from a similar distribution to the real data.

DNC was then run on augmented data using different combinations of augmentation techniques to check each was working properly. The Diarisation Error Rates (DER, the percentage of incorrectly labelled time) are given in Table 1. They are largely similar to those achieved in the original DNC paper, eg. 19.14% for sub-meeting randomisation compared to 20.19% originally. The slightly improved performance can be explained by the fact that currently each augmented meeting is based on a randomly sampled real meeting, whereas previously each meeting was augmented in turn. A different random seed

[1]The code is available at https://github.com/jamest08/newDNC

would lead to a slightly different DER. Diaconis augmentation and global input vectors randomisation currently perform slightly worse than before. The fact that each randomisation method has a similar DER with and without Diaconis augmentation suggests it is not behaving as expected. This problem is currently being investigated but has not prevented progress in other areas. At the time of writing, it has just been noticed that the original model rotated a meeting on every iteration rather than just at the beginning. This is likely to explain the problem as when producing one batch with the new functions, the meetings are only rotated once. This should not affect on-the-fly augmentation since the batch function is called every iteration and so we would effectively have a new rotation each time, but would affect the testing phase where one large batch is produced.

Randomisation	w/o Diac-Aug	w/ Diac-Aug
Sub-sequence	19.14	19.20
Input vectors (global)	19.58	19.09
Input vectors (meeting)	18.85	19.62

Table 1: % DERs for different augmentation techniques with meeting length of 50 segments on the eval set, trained on 735,000 augmented meetings

3.3 On-the-fly Augmentation

To allow data to be augmented on-the-fly, the function that produced a batch of augmented data was adapted to turn it into a generator. This yields a mini-batch of augmented data when called. The iterator and training functions from ESPNet [9] were adapted to call this generator function rather than iterating over a fixed batch of data. The system works properly and can train the network without needing to input a pre-augmented data file. Results are promising and match those achieved by the original implementation without the large storage costs. Performance may be improved further by experimenting with the number of epochs and mini-batch size.

4 Future work

4.1 Window-level Clustering

The next step is to move to clustering at the window level rather than the segment level. This means that rather than taking one averaged d-vector per segment and an associated speaker label, each individual d-vector will have a speaker label and be clustered individually. The bulk of the work for this was already done when changing the way data was inputted and formatted. The only real change to make is to no longer take the mean of d-vectors for a segment, and then some minor edits to the dictionary-producing functions. New test data will also need to be produced however this can be done with existing functions. Depending on performance, meeting-length may need to be restricted to maintain useful DERs.

4.2 Multiple Speakers

Moving to multiple speakers is a bigger step. First, the model will be adapted to allow a two channel input. There will be two d-vectors per window rather than one and the

network will be edited to allow two inputs. At this stage it will be useful to strip the required transformer code away from the rest of ESPNet to give a more standalone system.

There are a few design choices for dealing with overlapping speakers - due to time constraints it is likely that the number of overlapping speakers will be restricted to two. One option is to have two output nodes: a new one will flag if there is overlapping speech, and if there is, the two most probable speaker labels will be taken from the other node, rather than just one. Now there will need to be two speaker labels for each window, one of which can be NONE if there is only a single speaker.

5 Conclusion

While the original DNC showed that it was possible to use a neural network for clustering speaker embeddings, it was fairly limited by the restrictions required for input data. This project began by rebuilding the input pipeline to allow more sophisticated additions to be made. On-the-fly augmentation has been a success and will allow the model to be trained more easily and with reduced storage cost. Going forward, the project is well positioned to deliver several further improvements to DNC. This will put it a step closer to competing with the existing state-of-the-art and being applied usefully to real-world diarisation problems.

References

[1] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, *A review of speaker diarization: Recent advances with deep learning*, 2021. arXiv: 2101.09624 [eess.AS].

[2] G. Sun, C. Zhang, and P. C. Woodland, "Combination of deep speaker embeddings for diarisation," *Neural Networks*, vol. 141, pp. 372–384, Sep. 2021, ISSN: 0893-6080.

[3] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, *Speaker diarization with lstm*, 2018. arXiv: 1710.10468 [eess.AS].

[4] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, *Discriminative neural clustering for speaker diarisation*, 2020. arXiv: 1910.09703 [eess.AS].

[5] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[6] I. Mccowan, J. Carletta, W. Kraaij, *et al.*, "The ami meeting corpus," *Int'l. Conf. on Methods and Techniques in Behavioral Research*, Jan. 2005.

[7] P. Diaconis and M. Shahshahani, "The subgroup algorithm for generating uniform random variables," *Probability in the Engineering and Informational Sciences*, vol. 1, no. 1, pp. 15–32, 1987.

[8] X. Anguera, C. Wooters, and J. Hernando, "Acoustic beamforming for speaker diarization of meetings," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2011–2022, 2007.

[9] S. Watanabe, T. Hori, S. Karita, *et al.*, *Espnet: End-to-end speech processing toolkit*, 2018. arXiv: 1804.00015 [cs.CL].