New Advances in Speaker Diarization

Hagai Aronowitz[1]*, Weizhong Zhu*[2]*, Masayuki Suzuki*[3]*, Gakuto Kurata*[3]*, Ron Hoory*[1]

[1]IBM Research AI, Haifa, Israel
[2]IBM Research AI, Yorktown Heights, New York, USA
[3]IBM Research AI, Tokyo, Japan

{hagai@il, zhuwe@us, szuk@jp, gakuto@jp, hoory@il}.ibm.com

Abstract

Recently, speaker diarization based on speaker embeddings has shown excellent results in many works. In this paper we propose several enhancements throughout the diarization pipeline. This work addresses two clustering frameworks: agglomerative hierarchical clustering (AHC) and spectral clustering (SC).

First, we use multiple speaker embeddings. We show that fusion of x-vectors and d-vectors boosts accuracy significantly. Second, we train neural networks to leverage both acoustic and duration information for scoring similarity of segments or clusters. Third, we introduce a novel method to guide the AHC clustering mechanism using a neural network. Fourth, we handle short duration segments in SC by deemphasizing their effect on setting the number of speakers.

Finally, we propose a novel method for estimating the number of clusters in the SC framework. The method takes each eigenvalue and analyzes the projections of the SC similarity matrix on the corresponding eigenvector.

We evaluated our system on NIST SRE 2000 CALLHOME and, using cross-validation, we achieved an error rate of 5.1%, going beyond state-of-the-art speaker diarization.

Index Terms: speaker diarization, agglomerative hierarchical clustering, spectral clustering, uncertainty modeling, short utterances, number of clusters estimation

1. Introduction

Speaker diarization is the process of partitioning an input audio stream into clusters of segments according to speaker identity ("who spoke when"). Speaker diarization systems often consist of the following components: speech segmentation, speech embedding and clustering.

In the speech segmentation component, voice activity detection (VAD) is usually employed to remove the non-speech part. Each resulting segment may be further divided into smaller segments using any method that ensures that only a single speaker exists in most segments.

The speech embedding component extracts segment-based features such as i-vectors [1] or x-vectors [2]. The features are used to score the similarity of pairs of segments or pairs of clusters.

The clustering component groups the segments into hypothesized speakers, using a similarity scoring function defined on the extracted embeddings. One popular clustering scheme is Agglomerative Hierarchical Clustering (AHC) [3] which is a bottom-up approach, and a second popular clustering method is Spectral clustering (SC) [4].

In this work we address several components of the diarization pipeline. For the speech embedding component, we propose a fusion of multiple embeddings, namely, fusion of x-vectors and d-vectors [5].

For the similarity scoring of clusters which is fundamental in AHC, and for similarity scoring of segments which is key in SC, we propose training neural networks that integrate both acoustic similarity and duration information that indicates uncertainty in embeddings due to short durations.

Finally, we propose a novel approach to estimate the number of clusters. For each eigenvalue, we analyzre the temporal responses of the corresponding eigenvector on the SC similarity matrix.

The rest of the paper is organized as follows. Section 2 describes our baseline systems. Section 3 describes our contribution in the speech embedding component. Section 4 describes our contributions in similarity scoring. Section 5 introduces our contributions in the clustering component. Section 6 reports the experiments and results. Finally, Section 7 concludes the paper.

2. Baseline Systems

We consider two baseline systems, where both systems share the speech segmentation and speech embedding components, and differ only in the clustering component.

The first baseline system is based on AHC and the second is based on SC. We first describe the two shared components. Next, we describe the AHC-based system and the SC-based system.

2.1. Speech Segmentation

Our deployed system uses automatic speech recognition (ASR) to perform speech segmentation [6]. The benefits of using ASR are twofold. First, ASR can accurately remove the non-speech part. Second, the speaker diarization output is fully aligned with the ASR output. Because we wanted to compare our method to other works, we used in this work Oracle VAD to filter non-speech and create evenly-spaced overlapping segments of size 2.4s with 50% overlap, as done in [5].

2.2. Speech embedding

We use x-vectors as the baseline embeddings. These x-vectors are currently the most widely used method for deep learning-based speaker embedding. The x-vector architecture feeds a stack of Mel-frequency cepstral coefficients (MFCC) frames into a time-delay neural network (TDNN). The first layers are TDNN, followed by a time-pooling layer which accumulates statistics on the time frame-based mean and standard deviation of the segment. On top of the pooling layer, a single x-vector segment-level embedding is trained discriminatively on speaker labels. A detailed description can be found in [2].

2.3. AHC-based clustering

AHC starts by assigning each segment into a unique cluster. Then, in each iteration, AHC finds the most similar pair of clusters and merges them into a single cluster. The process stops according to a predefined threshold on the cluster pair similarity.

Our implementation starts with a pre-processing step of unit length normalization of the embeddings followed by removal of the session-dependent embedding mean. We then apply session-dependent principal component analysis (PCA) [7, 8] in the embedding space to further enhance speaker separability and deemphasize within-speaker variability (we expect only the top eigenvectors to account for speaker information).

Pairwise dot products are then computed on the dimensionality reduced embeddings. In each iteration AHC finds the most similar pair of clusters according to the dot product, and merges them together. When merging clusters *a* and *b* into a merged cluster *x*, we simply average our estimates on pairwise dot products. For any cluster *c:*

$$c \cdot x := \tfrac{1}{2}(c \cdot a + c \cdot b) \qquad (1)$$

A more detailed description of our implementation is reported in [6, 3, 9].

2.4. SC-based clustering

Our SC-based system is inspired by the system described in [5]. Given *n* observed embedding vectors, spectral clustering is based on the similarity (also called affinity) matrix (*n* x *n*). We conduct the following steps to estimate the number of clusters *k*:

1. Compute 10 largest eigenvalues, $\lambda_1,\ldots,\lambda_{10}$ of the similarity matrix
2. Search for the number of clusters *k* in the range of [2, …, 10]
3. Stop if the normalized eigengap $\frac{\lambda_{i+1}-\lambda_i}{\lambda_1}$ is below a threshold

Given the estimated number of clusters *k*, we stack the top-*k* eigenvectors into a *n* x *k* matrix and apply k-means on the *n* matrix rows.

3. Multiple Speech Embeddings

Our baseline systems use x-vectors for speech embedding, similar to other systems such as [10, 11]. In [5, 12] LSTM-based d-vectors proved their effectiveness for speaker diarization. In our diarization frameworks, we evaluated the fusion of x-vectors and d-vectors.

The d-vectors we use are different from Google's d-vectors [5] in the following manner. First, we employ frame-stacking and frame-skipping for MFCC frames. Second, we applied additive angular margin (ArcFace) loss [13] to train the model. A detailed description of our d-vector extractor and training setup can be found in [12].

For multiple-embedding experiments, we concatenate the x-vector and d-vector embeddings, which was found to be comparable to fusion in the similarity score domain.

4. Similarity Scoring with Neural Networks

Our baseline similarity scoring function is the dot product, which is equivalent to the cosine similarity for unit length normalized vectors. For AHC, we investigated methods for learning a similarity function between clusters, while for SC, we investigated methods for learning a similarity function between segments.

4.1. Cluster similarity for AHC

The accuracy of AHC results is highly dependent on a high-quality similarity function between pairs of clusters. We propose the following technique to augment the plain cosine similarity. We feed both the cosine similarity and the accumulated durations of both clusters into a neural network which is designed to estimate the utility of merging the two clusters into one. We name this architecture c-d-d (cosine-duration-duration).

Note, that a given cluster does not necessarily consist of a single speaker, because the clustering process is imperfect. Therefore, even given the ground truth speaker labels, the quality of a merge is not well defined.

We feed the durations of the clusters to the network, to better handle small clusters. As shown recently [14] in the context of speaker change detection, embedding uncertainty due to short audio segments can be handled when segment durations are fed to a neural network jointly with the acoustic information.

We use the architecture shown in Figure 1 to score the utility of merging a given cluster. During the clustering process, we find the top-*N* (*N*=100*)* pair candidates using the baseline scoring method (cosine similarity) and rescore the top-*N* pair candidates using the neural network.

We train the neural network by running the AHC pipeline on speaker labeled training data. Given a clustering iteration, we collect the top-*N* scoring pairs and, for each one, create a training example for the neural network. The cosine similarity and cluster durations are trivially given.

To set the target output of the neural network, we create for each cluster a histogram of the speakers that correspond to the cluster. For example, for a session with three labeled speakers, the histogram of a cluster contains three frequencies that correspond to the percentage of speech spoken by each of the three speakers in that cluster. We define the target output as the cosine similarity between the pair of associated histograms.

We use the mean squared error loss function with the Adam optimizer to train the NN.

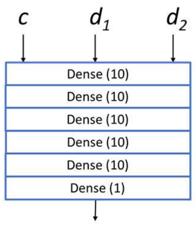

Figure 1: *Clusters-pair similarity scoring given cosine similarity (c) and clusters durations (d_1, d_2). Rectified linear units (*ReLU*) are used in all layers except for the last layer which uses a sigmoid. Network is named c-d-d.*

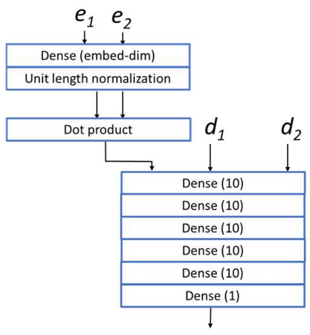

Figure 2: *Segments-pair similarity scoring given embeddings (e_1, e_1) and segments durations (d_1, d_2). Rectified linear units (*ReLU*) are used in all layers except for the first dense layer for the embeddings input which is linear, and the last layer which uses a sigmoid. The embedding dimension is denoted by embed-dim. The network is named e-e-d-d.*

4.2. Segments similarity for SC

A key component in SC is the ability to score the similarity of a pair of segments. We investigated two architectures.

The first architecture is the c-d-d network described in the previous subsection, namely the cosine similarity and segments durations are fed to the network described in Figure 1.

The second architecture is depicted in Figure 2. It combines the architecture from Figure 1 with a sub-network that replaces the cosine similarity input with a learnt similarity function that operates directly on the segment embeddings. We name this network e-e-d-d. The embedding-based learnt similarity function is a dot product between the outputs of a simple sub-network that performs a linear transformation followed by unit length normalization.

5. Estimating the number of speakers

Estimating the number of clusters is a crucial step in SC. The large eigenvalues of the similarity matrix usually correspond to speakers, and the small eigenvalues correspond to within-speaker variability (noise). However, in practice, it is often hard to find the right cutoff point, as there may be one or two borderline eigenvalues for which it is difficult to distinguish between an eigenvalue that corresponds to an actual speaker or corresponds to noise. In the following subsections we propose two methods for improving the estimation of the number of speakers

5.1. Deemphasizing short segments

Our goal is to deemphasize the impact of short segments on the process of estimating the number of speakers in SC, since the embeddings extracted from them are less reliable. We do this by applying the following on the similarity matrix. We scale each off-diagonal element of the matrix by $\frac{d_i+d_j}{2\cdot\max_k d_k}$, where d_i and d_j are the durations of segments i and j respectively and $\max_k d_k$ is the duration of the longest segment.

5.2. Temporal responses analysis

Ideally, it is expected that each top eigenvector of the similarity matrix corresponds to one or two speakers. Multiplying the similarity matrix with this eigenvector results in a vector we name the temporal response. Observing the absolute values of the components of the temporal response, we expect to get large values in coordinates corresponding to segments that belong to the speaker associated with the eigenvector. In case of two speakers associated to the eigenvector, one of the speakers will induce large positive values and the other will induce large negative values.

In case of an eigenvector that is not associated to a speaker, we expect the temporal response to be noisy.

We define an eigenvalue to be positive-dominant in segment *j* if the eigenvector has a positive projection on row *j* of the similarity matrix and the magnitude of the projection is maximal with respect to the magnitudes of the projections of other eigenvalues on row *j*. We define the term negative-dominant correspondingly.

For every speaker in the session, we expect to have an eigenvalue that is either positive-dominant or negative-dominant in all the rows that correspond to the speaker.

Therefore, for every eigenvalue we count separately the number of positive-dominant rows and negative-dominant rows and compare each one of them to a threshold (minimal number of segments). Each such count that exceeds the threshold accounts for a newly detected speaker.

The outline of our proposed method is as follows. *k* is the number of eigenvalues (10 in our setup).

1. Compute the temporal response matrix R: R=AE where A is the *n* x *n* similarity matrix and E is a *n* x *k* matrix stacking *k* top eigenvectors
2. Clear counter c_r for *r*=1,…,2*k*
3. For each row *i*
   a) Find maximal absolute value in row *i* in R → with index *j*
   b) If $R_{i,j}>0$ increase counter c_j, otherwise increase c_{j+k}
4. Count the number of counters c_r that exceed a threshold → Estimated number of speakers

6. Experiments

6.1. Embedding extractors

We adopt the publicly available x-vector extractor [15] for our experiments. This TDNN based x-vector was trained for the speaker diarization task based on an augmented Switchboard and NIST SREs.

The d-vector extractor is a 2-layer LSTM network with 768 hidden nodes. We applied 8 frame-stacking and 8 frame-skipping to input MFCCs. A mean pooling over time was applied to the hidden states of the final LSTM layer followed by a single linear layer to form 64-dimensional d-vectors. Full details on the training scheme can be found in [12].

6.2. Speaker diarization dataset

We used the NIST SRE 2000 CALLHOME dataset for our evaluation. The dataset contains 500 utterances in total, in 6 languages: English, Chinese, Japanese, Arabic, German, and Spanish. The number of speakers in the recordings, ranges from 2 to 7 speakers.

For training the systems, we use 5-fold cross validation as done in other works.

6.3. Experimental setup

We follow the common practice of using Oracle VAD to filter non-speech and create evenly spaced overlapping segments of size 2.4s with 50% overlap. We used the standard forgiveness collar of 0.25s and report diarization error rates (DER).

6.4. Deemphasis of short segments for SC

Table 1 reports our experiments with deemphasizing short segments (Subsection 5.1) under the SC framework with the x-vector based baseline system. The results indicate a relative error reduction of 7%. The rest our SC-based experiments that are reported in Table 3 use this technique.

6.5. Multiple speech embeddings

Results for using x-vectors, d-vectors and the concatenation of both embeddings are reported in Tables 1 and 2 for AHC and SC respectively. Relative error reduction for using the concatenation compared to using x-vectors only is 4% for AHC and 25% for SC.

6.6. Similarity scoring with neural networks

The c-d-d architecture was evaluated for AHC and SC. The results indicate a 6% relative error reduction (compared to the cosine similarity) for AHC (Table 2), and 8% for SC (Table 3).

The e-e-d-d architecture was evaluated for SC. The result which is reported in Table 3 indicates a 12% relative error reduction compared to the cosine similarity.

6.7. Temporal responses analysis

In order to improve the estimate of the number of speakers in SC, we fuse two estimates by averaging them. The first estimate is the baseline normalized eigengap-based method. The second estimate is based on the temporal response analysis (Subsection 5.2).

In case of a non-integer average, the estimate is rounded towards the baseline estimate. The result is presented in Table 3 and indicates a 4% relative error reduction compared to using only the baseline estimate.

Table 1. *DER results for the SC-based baseline system: analysis of the effect of short segments deemphasis*

Method	DER
No deemphasis of short segments	8.6
With deemphasis of short segments	8.0

Table 2. *DER results for the AHC-based system*

Method	DER
x-vectors	8.4
d-vectors	9.4
x-vectors + d-vectors	8.1
x+d-vectors + DNN c-d-d	7.6

Table 3. *DER results for the SC-based system. All experiments include deemphasis of short segments*

Method	DER
x-vectors	8.0
d-vectors	7.9
x-vectors + d-vectors	6.0
x+d-vectors + DNN c-d-d	5.5
x+d-vectors + DNN e-e-d-d	5.3
x+d-vectors + DNN e-e-d-d + Temporal analysis	5.1

Table 4. *DER results for recent works on NIST-2000 CALLHOME*

Work	DER
Diaez et al. [16]	9.0
Zhang et al [17] [1]	7.6
Mcree et al. [18]	7.1
Liu et al. [19]	6.6
Huang et al. [20]	6.5

7. Conclusions

In this work we address two popular speaker diarization schemes: agglomerative hierarchical clustering (AHC) and spectral clustering (SC). We propose several general advances that are applicable to both schemes, and some advances that are applicable to only one of them.

First, we use multiple speaker embeddings. We show that fusion of x-vectors and d-vectors boosts accuracy significantly.

Second, we train neural networks to leverage both acoustic and duration information for scoring similarity of segments or clusters. Third, we handle short duration segments in SC by deemphasizing their effect on setting the number of speakers.

Finally, we propose a novel method for estimating the number of clusters in the SC framework. The method takes each eigenvalue and analyzes the temporal responses of the corresponding eigenvector on the SC similarity matrix.

We evaluated our system on NIST SRE 2000 CALLHOME and, using cross-validation, we achieved a DER of 5.1% with the SC-based system, compared to the 8.6% baseline (41% relative error reduction), going beyond state-of-the-art speaker diarization (see Table 4).

[1] not comparable due to use of VAD (not Oracle)

8. References

[1] N. Dehak et al., “Front-end factor analysis for speaker verification, in *IEEE Transitions on Audio, Speech, and Language Processing*, 2011.

[2] D. Garcia-Romero et al., “Speaker diarization using deep neural networks embeddings.”, in *IEEE International conference on Acoustics, Speech, and Language Processing*, 2017.

[3] W. Zhu and J. Pelecanos., “Online speaker diarization using adapted i-vector transform” in *IEEE International conference on Acoustics, Speech, and Language Processing*, 2016.

[4] A.Y. Ng, M. I. Jordan and Y. Weiss, "On spectral clustering: analysis and an algorithm", in *Advances in Neural Information Processing Systems,* 2002.

[5] Q. Wang, C. Downey, L. Wan, P. A. Mansfield and I. L. Moreno, “Speaker Diarization with LSTM”, in Proc. *ICASSP*, 2018.

[6] D. Dimitriadis and P. Fousek, “Developing on-line speaker diarization system” in *InterSpeech,* 2017.

[7] H. Aronowitz, "Unsupervised Compensation of Intra-Session Intra-Speaker Variability for Speaker Diarization", in Proc. *Speaker Odyssey*, 2010.

[8] S.H. Shum et al., “Unsupervised method for speaker diarization” An integrated and iterative approach,” in *IEEE Transitions on Audio, Speech, and Language Processing*, 2013.

[9] K. Church et al., “Speaker diarization: perspective on challenges and opportunities from the theory to practice,” in *IEEE International conference on Acoustics, Speech, and Language Processing*, 2017.

[10] G. Sell et al., “Diarization is Hard: Some Experiences and Lessons Learned for the JHU team in the Inaugural DIHARD Challenge”, in *Proc*. Interspeech, 2018.

[11] M. Diez et al., “Bayesian HMM based x-vector clustering for Speaker Diarization”, in Proc. *Interspeech*, 2019.

[12] Y. Higuchi, M. Suzuki, and G. Kurata, “Speaker embeddings incorporating acoustic conditions for diarization,” in Proc. *ICASSP*, 2020.

[13] J. Deng, J. Guo, N. Xue and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in Proc. *CVPR*, 2009.

[14] H. Aronowitz, W. Zhu, “Context and Uncertainty Modeling for Speaker Change Detection”, in Proc. *ICASSP*, 2020.

[15] https://kaldi-asr.org/models/m6

[16] M. Diez et al., “Speaker diarization based on Bayesian HMM with eigenvoice priors”, in *Proceedings of Odyssey*, 2018.

[17] A. Zhang et al., “Fully supervised speaker diarization,” in Proc. *ICASSP*, 2019.

[18] A. McCree at el., “Speaker Diarization using leave-one-out Guassian PLDA clustering of DNN embeddings”, in Proc. *InterSpeech,* 2019.

[19] Q. Lin et al., “LSTM based similarity measurement with spectral clustering for speaker diarization”, in Proc. *Interspeech,* 2019.

[20] Z. Huang, S. Watanabe, Y. Fujita, P. García, Y. Shao, D. Povey, and S. Khudanpur, “Speaker Diarization with Region Proposal Network”, (2020), ArXiv, abs/2002.06220.