ZCU-NTIS (MH-crR) Speaker Diarization System for the DIHARD 2018 Challenge

Zbyněk Zajíc[1], Marie Kunešová[1,2], Jan Zelinka[1,2], Marek Hrúz[1]

University of West Bohemia, Faculty of Applied Sciences
[1]NTIS - New Technologies for the Information Society and [2]Dept. of Cybernetics,
Univerzitní 8, 306 14 Plzeň, Czech Republic

Abstract. This paper describes the **"MH-crR"** diarization system developed by the team from the New Technologies for the Information Society (NTIS) research center of the University of West Bohemia (team "ZCU-NTIS"), for the First DIHARD Speech Diarization Challenge. Our system follows the currently-standard approach of segmentation, i-vector extraction, clustering and resegmentation. Additionally, we use an ANN-based domain classifier, which categorizes each conversation into one of 10 domains (the 9 corpora from the development set, plus "other"). This classification determines the specific system configuration (expected number of speakers, stopping criterion, etc.). This "MH-crR" version of the system is mostly identical to our "ws-crR" system, with the exception of the segmentation step. Here, the simple fixed-length segmentation was replaced by speaker change detection based on a convolutional neural network. This system achieves a DER of 27.12% and an MI of 8.31 bits on the evaluation set using gold segmentation (Track 1). In Track 2, DER of 46.14% and MI of 7.77 bits were achieved with our speech activity detector.

1 Differences from our other two systems

This document describes the entire "MH-crR" system, including parts that are identical to our other two systems. For convenience, here we list the parts which are different:

Compared to our most basic "ws-cr" system, both this and the "ws-crR" system perform resegmentation (section 3.6), use a different approach for SEEDLingS data (sections 2.5 and 3.7), and also include speech activity detection (sections 2.4, 3.2 and 4.5).

Additionally, the "MH-crR" system also uses a different segmentation - the simple fixed-length segmentation was replaced by speaker change detection based on a convolutional neural network (sections 2.6, 3.3, and 4.6). We also apply a weighing during the accumulation process for i-vector extraction (section 3.4).

Finally, the total execution times (section 4.7) are different for each system.

2 Data resources

2.1 i-Vector extraction for segment representation

Data for UBM:

- LibriSpeech (http://www.openslr.org/12/, training sets only),
- DIHARD Challenge Development Data (LDC2018E31),
- Czech Speecon database (ELRA-S0298, child voices only),
- UK English Speecon database (ELRA-S0215, child voices only),
- US English Speecon database (ELRA-S0233, child voices only),
- TIMIT Acoustic-Phonetic Continuous Speech Corpus (LDC93S1),
- CSR-I (WSJ0) Complete (LDC93S6A),
- CSR-II (WSJ1) Complete (LDC94S13A),
- AMI Meeting Corpus (http://groups.inf.ed.ac.uk/ami/download/),
- RT-03 MDE Training Data Speech (LDC2004S08),
- Santa Barbara Corpus of Spoken American English Part II (LDC2003S06)

Data for FA model:

- Czech Speecon database (ELRA-S0298, child voices only),
- UK English Speecon database (ELRA-S0215, child voices only),
- US English Speecon database (ELRA-S0233, child voices only),
- TIMIT Acoustic-Phonetic Continuous Speech Corpus (LDC93S1),
- CSR-I (WSJ0) Complete (LDC93S6A),
- CSR-II (WSJ1) Complete (LDC94S13A),

2.2 i-Vector extraction for the domain classifier

Data for UBM (only 2-3 hours of data from each corpus):

- LibriSpeech (http://www.openslr.org/12/, training sets only),
- DIHARD Development Data (LDC2018E31),
- AMI Meeting Corpus (http://groups.inf.ed.ac.uk/ami/download/),
- RT-03 MDE Training Data Speech (LDC2004S08),
- Santa Barbara Corpus of Spoken American English Part II (LDC2003S06)

Data for FA model:

- DIHARD Development Data (LDC2018E31),
- LibriSpeech (http://www.openslr.org/12/, training sets only)

2.3 Domain classifier (ANN):

- DIHARD Development Data (LDC2018E31),
- LibriSpeech (http://www.openslr.org/12/, training sets only),
  (10 randomly chosen 10 min recordings were used as additional LibriVox data)

For the training process, we excluded 1 randomly selected recording from each domain for testing the performance of the network. The final network was trained on all data, but with the same number of epochs.

2.4 Speech Activity Detector

The speech activity detector was trained only on the DIHARD development set.

2.5 Adult-Child classification

The adult GMM was trained on the same data as the general UBM for i-vector extraction (listed in sec. 2.1), with the exception of the Speecon child voices, which were instead used to train the child UBM. Both these models were further adapted on the SEEDLingS development data, which were manually divided into clean child and adult segments. For the child UBM, we also obtained several other short recordings of small children (6 children between 3 months and 3 years old, total length 6 min), which were added to the adaptation data.

2.6 CNN-based Speaker Change Detection

The network was trained only on the YouthPoint subset of the DIHARD development data.

3 Detailed description of algorithm

Our system [1–3] follows an i-vector-based approach, as introduced in [4–6]: First, each recording is divided into short segments and i-vectors are extracted. Then, a clustering method is used in order to determine which parts of the signal were produced by the same speaker. Finally, a GMM-based resegmentation is performed to refine the positions of boundaries between speakers. For the DIHARD Challenge, we have also introduced a domain classifier that determines the source of each recording and selects the most suitable system configuration.

3.1 Feature extraction

We used Linear Frequency Cepstral Coefficients (LFCCs), Hamming window of length 25 ms with 10 ms shift. There are 40 triangular filter banks linearly spread across the frequency spectrum, and 25 LFCCs are extracted. The resultant 50-dimensional feature vector ($D_f = 50$) also includes delta coefficients.

3.2 Speech activity detection

This particular system was used for both Track 1 and Track 2. In Track 1, information about speaker activity was received as gold speech segmentation. For Track 2, we used our speech activity detector (SAD), which is based on an enhanced version of the approach described in [7]. We use a neural network (depicted in Figure 1) consisting of three parts: The first part computes a spectral flux or other analogous spectral features and also contains a feature extractor. The second part computes a score from contextual information. And the last part functions as a decoder.

The first part is three standard CNN layers with ReLU activation function, each followed by a pooling layer with max pooling function, and ending with four standard fully-connected NN layers with sigmoid activation functions. The second part is a

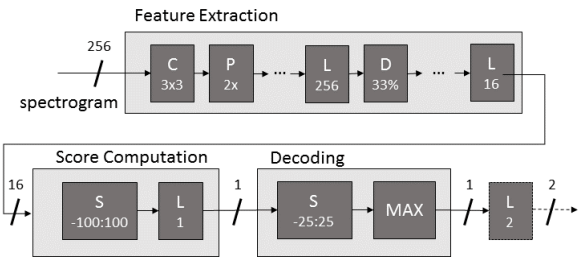

Fig. 1. The schema of the NN for SAD, where C = convolution, P = pooling, D = dropout, L = fully-connected and S = splicing layer.

splicing that makes a long-temporal window and one single neuron with linear activation function. The last part also uses a long-temporal window and it finds the maximal score in the window.

Our SAD processes the logarithm of amplitude spectrum that is computed from 512 samples long window with 160 samples step (hop). Thus, the spectrum has 256 features. Each CNN layer uses a 3x3 window and it has 6 kernels. Fully-connected layers in the second part have 256 neurons, except for the last layer, which has 16 neurons. The window splices 201 feature vectors. The window is symmetric, i.e. used time shifts are from -100 to +100. The window in the third process is also symmetric and the length of the window is 51.

In the training process, we added one layer with softmax activation function and we trained all parts simultaneously by means of cross-entropy criterion and SGD. Due to the long windows, we used batches of 128 randomly selected continuous parts of recordings, always 2000 spectral features long.

3.3 Segmentation

Each recording is split into multiple individual speech regions by breaking it on any non-speech longer than 0.5 s. These speech regions are then further split based on potential speaker changes detected by a Convolutional Neural Network (CNN) [2].

The CNN was trained as a regressor on spectrograms of acoustic signal, with reference information L about existing speaker changes. The value of the function L in time t is computed via the formula in Equation 1:

$$L(t) = \max\left(0, 1 - \frac{\min_i\left(|t - s_i|\right)}{\tau}\right), \tag{1}$$

where s_i is the time of i^{th} speaker change and $\tau = 0.6$ is the tolerance which models the level of uncertainty of the manual labeling.

The output signal P gives the probability of a speaker change at each given moment. Speaker changes are identified as peaks in the signal P (after normalization into the interval $< 0, 1 >$ for each conversation), using non-maximum suppression with a window size of 10 features. We also apply a threshold of 0.5 on the detected peaks in

order to remove insignificant local maxima. The signal between two detected speaker changes is considered as one segment. To ensure that each segment contains sufficient information about the speaker, we set the minimum duration of each segment at one second. Shorter segments are discarded and the decision about the speaker is left for the resegmentation step.

(The following descriptions of the CNN's architecture and training process are taken from [2].)

Architecture of the CNN: The architecture of the CNN consists of three convolutional layers with ReLU activation functions. Each convolution layer is followed by a max pooling layer and a batch normalization layer [8]. The last two layers are fully connected with sigmoid activation function. The architecture is summarized in Table 1. The shape of the filters in the first convolutional layer respects the usually rectangular shapes of the high energy speech harmonics in the spectrogram. This layer serves as a visual features detector. The output layer consist of just one neuron with sigmoid activation function.

Table 1. Summary of the architecture of the CNN.

Layer	Kernels	Size	Shift
Convolution	50	32 x 16	2 x 2
Max pooling		2 x 2	2 x 2
Batch Norm			
Convolution	200	4 x 4	1 x 1
Max pooling		2 x 2	2 x 2
Batch Norm			
Convolution	300	3 x 3	1 x 1
Max pooling		2 x 2	2 x 2
Batch Norm			
Fully Connected	4000		
Fully Connected	1		

Training of the CNN: We use a Binary Cross Entropy loss function in the training process. It is optimized by Stochastic Gradient Descent with batch size of 64 and we change the learning rate after a fixed number of steps. When the loss function is stabilized we use RMSProp algorithm for fine tuning of the network's weights.

3.4 Segment description

Each segment is represented by an i-vector derived from the supervector of accumulated statistics - zeroth and first statistical moments of data related to a UBM as a GMM

with $M = 1024$ components. The dimensionality of this supervector is reduced by Factor Analysis (FA) [9, 10] into $D_w = 100$ and we have used conversation-dependent Principal Component Analysis (PCA) [6] to reduce the dimension further into 3 or 9 (depending on the specific data - see Tab. 2).

Because we cannot be certain that each segment only contains the speech of a single speaker, not all data from a segment should contribute to the supervector equally. With CNN-based segmentation, we can reuse the output of the CNN (the probability of a speaker change in the signal) as an indication of the suitability of each frame. The part of the audio segment in time t with a high probability of a speaker change $P(t)$ is less appropriate to represent the speaker than a part with a small value of $P(t)$. Thus, we use the value of $1 - P(t)$ as a weighting factor of the signal during the accumulation process [3].

3.5 Clustering

Given i-vector representations of the extracted segments, we perform a clustering into sets of i-vectors describing different speakers. For clustering we mainly use agglomerative hierarchical clustering (AHC). The system starts with each i-vector in a separate cluster and then merges the closest pairs until it reaches a stopping point. The distance between two clusters is calculated as the average cosine distance between each pair of i-vectors. The stopping condition is a combination of maximum merging distance and a minimum and a maximum number of clusters: First, we perform AHC by merging the closest pairs of clusters until the lowest distance exceeds a specific threshold. If the resulting number of clusters is not within the expected range, we adjust the stopping point so that we reach either the minimum or maximum allowed number of clusters. If we are certain there are only 2 speakers in the conversation, we use the k-means algorithm instead (also with cosine distance). All mentioned parameters were selected on a per-corpus basis using the development set (see section 3.8).

3.6 Resegmentation

To make the final diarization more precise, we refine it by resegmentation. We compute GMMs over the feature vectors, one GMM for each speaker cluster. Then the whole conversation is redistributed frame by frame according to the likelihoods of the GMMs, filtered by a Gaussian window (length 75 ms with shift 50 ms) to smooth the peaks in the likelihoods. The number of GMM components depends on the amount of data in each cluster (1 GMM component per 2 segments in the cluster, rounded down to the nearest power of 2) and ranges between 1 and 64.

3.7 Adult-Child classification

As our standard AHC approach does not work very well on SEEDLingS data, we use a simplified alternate approach:

Only two speakers are expected in the recording - one child and one adult. We have prepared a separate UBM for children and for adults and we classify each frame of the

recordings as one of the two categories, using the same algorithm we use for resegmentation. After this adult-child classification, we also use a regular resegmentation using GMMs created from the specific conversation, as described in section 3.6.

3.8 Domain Classification

The domain classifier was implemented as a neural network in Keras, using TensorFlow. It was trained with one hidden layer (2048 neurons, tanh activation function) followed by 0.9 dropout and the output layer as softmax into 9 categories, batch size = 32, epochs = 25, categorical cross-entropy, "adam" optimizer. The remaining hyperparameters were left at default values.

The input to the classifier is a single i-vector calculated over the entire recording. It was extracted using a UBM with 512 components and a FA model with dimension 100. The network outputs the probability of each of the 9 corpora in the DIHARD development set.

Because of the presence of unseen corpora in the evaluation set, we have also added a threshold (= 0.5) on the output probability from the classifier and categorize lower-scoring recordings as "unknown domain".

The experimentally chosen parameters for each corpus in the development set are listed in Table 2 and were chosen as follows:

- The target number of speakers was observed directly from the development data (with the exception of VAST and SEEDLingS).
- PCA dimension reduction was selected from three options: reduction to dimension 3 or 9, or no reduction.
- The SAD threshold for speech/nonspeech was found as the setting with the lowest speech/nonspeech classification error (tested in increments of 0.05).
- The AHC stopping threshold was found as the setting with the lowest DER at the clustering stage - without resegmentation (tested in increments of 0.02).
- For VAST, lowest overall DER was achieved with only a single cluster in each recording (i.e. the system did not work well on these data).
- For SEEDLingS, we used the alternate adult-child classification approach described in section 3.7.

4 Hardware requirements

The main body of the system was implemented in Matlab. The code was not optimized with regards to execution time (e.g., intermediate results were saved to the disk).

The feature extractor, i-vector extractor and GMM adaptation were all in separate executables (C++), called from Matlab.

Domain classification, speech activity detection and the CNN-based speaker change detection were computed separately, using Theano (SAD) and Keras with TensorFlow.

Table 2. Experimentally chosen parameters for each corpus, system "MH-crR".

corpus	SAD Thresh.	Clustering	No. spk	AHC Thresh.	PCA dim
SEEDLingS	0.60	Adult/Child	2	-	-
SCOTUS	0.95	AHC	5-10	0.64	9
DCIEM	1.15	k-means	2	-	3
ADOS	1.10	k-means	2	-	3
YouthPoint	0.90	AHC	3-5	0.62	9
SLX	1.10	AHC	2-6	0.58	9
RT-04S	-0.55	AHC	3-10	0.76	9
LibriVox	1.00	-	1	-	-
VAST	0.55	-	1	-	-
other	0.80	AHC	3	-	9

4.1 Training of UBM models for i-vector extraction:

Hardware:

- CPU 8-core Intel Xeon E5-2650v2 2.60 GHz
- GPU 2x nVidia Tesla K20, 5GB, 1000 GFLOPS
- 4 GB RAM
- 10 GB required storage

Total training time: approx. 96 hours

4.2 Training of FA models for i-vector extraction:

Hardware:

- CPU 8-core Intel Xeon E5-2650v2 2.60 GHz
- 22 GB RAM
- 10 GB required storage

Total training time: approx. 48 hours

4.3 Training of UBM and FA models for the domain classifier:

Hardware:

- CPU Intel(R) Core(TM) i7 cpu - 1 core used, 3.07GHz
- GPU NVIDIA GeForce 1080 Ti, 11 GB VRAM, 11,340 GFLOPS
- 32 GB RAM
- 5.5 GB required storage

Total training time: approx. 5 hours

4.4 Training of the domain classifier ANN:

Hardware:

- CPU Intel(R) Core(TM) i7 cpu - 1 core used, 3.07GHz
- GPU NVIDIA GeForce 1080 Ti, 11 GB VRAM, 11,340 GFLOPS
- 32 GB RAM
- 5.5 GB required storage

Implemented in Keras, using TensorFlow.
Total training time: approx. 5 minutes

4.5 Training of the speech activity detector:

Hardware:

- CPU Intel(R) Core(TM) i7 cpu - 1 core used, 3.07GHz
- GPU NVIDIA GeForce GT 640, 2 GB, 691.2 GFLOPS
- 8 GB RAM
- 5.5 GB required storage

Implemented in Theano.
Total training time: approx. 10 hours

4.6 Training of the CNN for speaker change detection:

Hardware:

- CPU Intel(R) Core(TM) i7 cpu - 1 core used, 3.07GHz
- GPU NVIDIA GeForce 1080 Ti, 11 GB VRAM, 11,340 GFLOPS
- 32 GB RAM
- 5.5 GB required storage

Implemented in Keras, using TensorFlow.
Total training time: approx. 30 minutes

4.7 Execution times to process an average 10 minute recording:

Hardware (main system):

- CPU Intel(R) Core(TM) i7 cpu - 4 cores used, 3.07GHz
- GPU NVIDIA GeForce 1080 Ti, 11 GB VRAM, 11,340 GFLOPS
- 32 GB RAM
- 80 MB required storage

Execution time:

- Domain classification: $\sim$30 s
- Speech activity detection: $\sim$20 s
- Speaker change detection: $\sim$60 s
- Main system: 145 s
- **Total time:** $\sim$255 s

5 Acknowledgements

The work was supported by the project no. P103/12/G084 of the Grant Agency of the Czech Republic. Access to computing and storage facilities (CESNET LM2015042) is greatly appreciated. The authors would also like to thank their friends and colleagues who provided recordings of their children as additional training data.

References

1. Z. Zajíc, M. Kunešová, and V. Radová, "Investigation of Segmentation in i-Vector Based Speaker Diarization of Telephone Speech," in *Specom*. Budapest: Springer, 2016, pp. 411–418.
2. M. Hrúz and Z. Zajíc, "Convolutional Neural Network for Speaker Change Detection in Telephone Speaker Diarization System," in *ICASSP*. New Orleans: IEEE, 2017, pp. 4945–4949.
3. Z. Zajíc, M. Hrúz, and L. Müller, "Speaker diarization using convolutional neural network for statistics accumulation refinement," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, Stockholm, 2017, pp. 3562–3566.
4. G. Sell and D. Garcia-Romero, "Speaker Diarization with PLDA I-vector Scoring and Unsupervised Calibration," in *IEEE Spoken Language Technology Workshop*, South Lake Tahoe, 2014, pp. 413–417.
5. M. Senoussaoui, P. Kenny, T. Stafylakis, and P. Dumouchel, "A Study of the Cosine Distance-Based Mean Shift for Telephone Speech Diarization," *Audio, Speech and Language Processing*, vol. 22, no. 1, pp. 217–227, 2014.
6. S. Shum, N. Dehak, E. Chuangsuwanich, D. Reynolds, and J. Glass, "Exploiting Intra-Conversation Variability for Speaker Diarization," in *Interspeech*, Florence, 2011, pp. 945–948.
7. S. Thomas, S. Ganapathy, G. Saon, and H. Soltau, "Analyzing convolutional neural networks for speech activity detection in mismatched acoustic conditions," in *ICASSP*. Florence: IEEE, may 2014, pp. 2519–2523.
8. S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Arxiv*, vol. abs/1502.0, 2015.
9. P. Kenny and P. Dumouchel, "Experiments in Speaker Verification Using Factor Analysis Likelihood Ratios," in *Odyssey*, Toledo, 2004, pp. 219–226.
10. L. Machlica and Z. Zajíc, "Factor Analysis and Nuisance Attribute Projection Revisited," in *Interspeech*, vol. 2, Portland, 2012, pp. 1570–1573.