CONTEXT AND UNCERTAINTY MODELING FOR ONLINE SPEAKER CHANGE DETECTION

Hagai Aronowitz[1], Weizhong Zhu[2]

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

ABSTRACT

Speaker change detection is often addressed as a key component in speaker diarization systems. In this work we focus on online speaker change detection as a standalone task which is required for online closed captioning of broadcast television. Contrary to related works, we do not operate on frame-level features such as MFCC. Instead, we leverage state-of-the-art speaker recognition-based technology by modeling sequences of pretrained speaker embeddings (x-vectors) using a deep neural network. We explicitly address two types of uncertainties. The first one is uncertainty in embedding point estimate which is due to short and varying segment duration. The second type is uncertainty in which context segments are relevant to representing the speaker talking right before the hypothesized speaker change. We also show the robustness of affinity matrix-representation for speaker change detection. Our methods provide very significant accuracy improvements compared to several baselines including a recently published end-to-end system.

Index Terms— Online speaker change detection, affinity matrix, uncertainty modeling in deep learning, duration modeling

1. INTRODUCTION

Speaker change detection (SCD) is the task of locating all the positions in an audio stream in which the speaker identity changes from one speaker to another. SCD is commonly addressed as a component of speaker diarization ("who spoke when") systems.

As part of a speaker diarization pipeline, the requirements from SCD are usually quite modest, with a focus on a low misdetection rate, which comes with the price of extensive over-segmentation which may be mended later in the diarization pipeline, during the clustering step.

However, important tasks do exist in which the output of a speaker change detector is required, such as closed captioning (CC) of broadcast television for the hearing-impaired.

Starting July 2017, the United States Federal Communications Commission (FCC) requires television vendors to support speaker change detection and speaker recognition ("Captioning shall provide nonverbal information that is not observable, such as the identity of speakers,") [1]. Moreover, for broadcast news the requirement is near real-time CC. In this work we address online speaker change detection in the context of low-latency CC for broadcast news.

The task we address is given an online speech-to-text (STT) system that produces very short textual fragments (often containing a single word or two), we need to decide whether a speaker change mark ">>" should precede the current text fragment which is displayed in near real-time. Note that the decision should be based on the current audio segment (corresponding to the textual fragment) and previous segments only.

Our main contributions in this paper are as follows. First, we model a sequence of audio embeddings (contrary to frame level features) for the SCD task, and do so using an affinity matrix representation, which is both accurate and robust to domain mismatch. Second, we address embedding uncertainty (due to short and varying duration) within the framework of a neural network. Third, we address uncertainty in context relevancy within the framework of a neural network.

More generally, we propose a framework for handling input uncertainty in speaker change detection, which can be applicable to other related tasks such as speaker recognition and diarization.

The rest of the paper is organized as follows: Section 2 provides an overview of related work. Section 3 describes our proposed systems. Section 4 describes our approach for handling input uncertainty. Section 5 reports the experiments and results. Finally, Section 6 concludes the paper.

2. RELATED WORK

Prior to reviewing related work, we note that speaker diarization systems may do without SCD, by segmenting the audio stream into very short segments, either evenly spaced or using a speaker activity detector (SAD), and passing the segments to a clustering sub-system [2-4]. Thus, the related work is limited than expected. Note however that accurate SCD is expected to improve diarization.

Speaker diarization systems that explicitly implement SCD, very often follow the following framework. First, potential speaker change points are specified, for instance by sampling the audio stream in resolution of ~0.5-1 seconds. Then, for each potential speaker change point the preceding and following audio segments (and possibly also the union of them) are parameterized. Each pair of such parameterizations is then scored. In the following subsections we elaborate on both steps. Then, we describe two methods that deviates from this common framework.

2.1. Audio segment parameterization

A common approach is to extract frame level features such as MFCC and parameterize the segment using a Gaussian [5, 6]. Recent approaches leverage speaker recognition techniques and parameterize segments using anchor models [7], speaker factors [8], or i-vectors [8].

Lately, deep learning has been successfully used for segment parameterization. The x-vector architecture [3] embeds a stack of MFCC frames (corresponding to 2 seconds of speech) into an x-vector using time-delay neural network (TDNN) layers followed by temporal pooling. [9] use long-short term memory (LSTM) with

triplet loss-based training to embed stacks of MFCC frames (2 seconds long). [2] also use LSTM to embed stacks of log-Mel-filter-bank energies of up to 400ms. [10] utilize a Siamese architecture with a multi-layer Fully Connected (FC) network to embed 500ms stacks of MFCCs using a bottleneck layer.

2.2. Scoring

Gaussian representations have been traditionally scored using KL-distance [5] and BIC [6] which crudely assume that each speaker's low-level features distribute normally. Probabilistic discriminant analysis (PLDA) [31, 8] has been used to score speaker factors and i-vectors. A discriminatively trained metric has been used in [26] to score x-vectors. [11] used a Convolutional Neural Network (CNN) to directly verify whether a 1.4-seconds spectrogram is centered near a speaker change.

2.3. SCD as a sequence labeling task

The authors of [13] take a sequence $x_1,\ldots,x_T$ of frame-level features (MFCC+Δ+ΔΔ+ΔEnergy+ΔΔEnergy) and train a network based on two bi-directional LSTMs followed by three dense layers to predict labels $l_1,\ldots,l_T$ where l_t denotes whether frame t is a speaker change. The duration of the sequences processed is 200 frames (3.2 seconds).

2.5. End-to-End system

Recently [27], the task of SCD in broadcast news was addressed using an end-to-end approach. Given two adjacent segments parameterized by PLP+Δ+ΔΔ features, a Siamese network is used to extract embeddings which are then concatenated and fed into the classifier which is a fully-connected network. The Siamese network consists of three BLSTM layers followed by two fully connected tanh layers. Thus, variable length sequences are mapped into fixed dimensional embeddings. The embedding layers are pretrained using the triplet loss. After pretraining, the end-to-end network is jointly trained with a loss targeting to maximize SCD classification accuracy. The system was evaluated on Hub4. We contrast our experimental results to this system using the same scoring script.

3. ONLINE SPEAKER CHANGE DETECTION

Our pipeline starts with an STT-based temporal segmentation of the audio stream into short segments (1.5s in average). These segments are embedded into either i-vectors or x-vectors followed by centering, Within Speaker Covariance Normalization (WCCN) [14] and length-normalization. We parameterize a given segment t with matrix X_t by stacking the k preceding embeddings as columns $X_t = [x_{t-k,\ldots,} x_t]$ where x_i denotes the embedding extracted for segment i, and k is the number of context embeddings used to parameterize a segment (in this paper k is 9). The process of temporal segmentation and embedding sequence parameterization is depicted in Figure 1.

Alternatively, we may parameterize a given segment t with matrix A_t of stacked columns $A_t = [a_{t-k,\ldots,} a_t]$ where a_i denotes a vector of affinities between embedding x_i and a set of context-embeddings. In this paper we choose the context-embeddings for A_t to be embeddings $\{x_{t-k},\ldots, x_t\}$, and use the cosine similarity as the affinity measure. This leads to Equ. (1):

$$A_t^{m,n} = \langle x_{t-k+m}, x_{t-k+n} \rangle \quad (1)$$

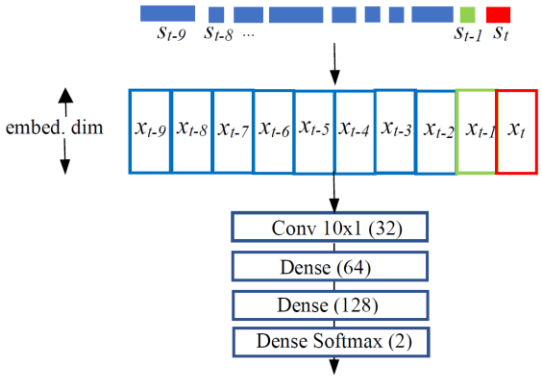

Figure 1: Architecture of the CNN network for the sequence of embeddings. Number of nodes/kernels is in parentheses.

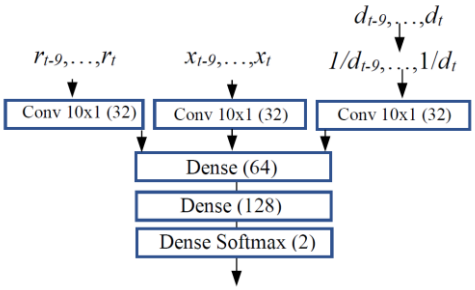

Figure 2: Handling input uncertainty in the CNN framework.

where m and n denote the row and column indices, respectively.

We define label l_t to be a binary variable indicating whether segments t and t-1 associate to different speakers. Given input matrices $\{X_t$ or $A_t\}$ and labels $\{l_t\}$, we train neural networks to predict the label of each matrix using a cross entropy loss. In evaluation phase, we predict the labels (speaker changes) using the corresponding matrices (each matrix predicts a single label).

3.1. Temporal segmentation

The IBM Cloud STT Service [15] is used for segmenting the audio stream into short segments, as described in [16]. The service runs in real time and with low-latency. For a coming speech audio signal, the service performs deep learning-based SAD, followed by STT decoding. The service outputs textual fragments (a single word or more) with the corresponding endpoints of the segment.

3.2. Audio embeddings

We evaluate both i-vectors and x-vectors. The i-vector extractor is described in detail in [17]. We use separate 8KHz and 16KHz 64-dimensional i-vector extractors. WCCN was trained at the short segment level rather than the session level.

We evaluate two 16KHz-based 512-dimensional x-vector extractors. The first one was downloaded from [28] and is based on the work described in [18]. We denote it by Kaldi-x-vector. The x-vectors are produced by a feed-forward DNN. The first layers of the network are at frame level with a time delay architecture. A statistics pooling layer aggregates over frame-level output vectors of the DNN and computes their means and standard deviations. Segment-level embeddings are extracted from the statistics pooling layer.

For the second x-vector extractor (denoted by IBM-x-vector) we used the scripts and code from [28] but retrained using more data as described in section 5.

3.3. Systems

The first baseline system is based on a simple cosine distance between the pair of embeddings $\{x_{t-1},x_t\}$. The second baseline system is based on the corresponding PLDA [31] score. We evaluated both CNN and LSTM-based neural networks and obtained better results with the CNN architecture. We use rectified linear units unless noted otherwise, and mini-batch size is 128. The sequence of embeddings is of length 10.

CNN network for stacks of embeddings (CNN)
The network is detailed in Fig. 1. The convolution layer processes each embedding coordinate independently (across the context). The next 3 layers are dense. Other configurations have been investigated but underperformed. Dropout with probabilities of 0.25 and 0.5 is applied after the first and second dense layers

CNN network for affinity matrices (AF-CNN)
The CNN network described in Figure 1 turned out to be optimal also for classifying affinity matrices. The only difference is that the network is fed with a 10x10 affinity matrix defined in Eq. (1), instead of a stack of embeddings.

4. HANDLING UNCERTAINTY

4.1. Modeling embedding uncertainty

It has been shown previously by Kenny et al. [29] that under the i-vector framework, when scoring short-duration segments it is best to propagate i-vector uncertainty into PLDA scoring. We propose to do the same with our CNN-network. Contrary to [29], we use segment duration as an indication on x-vector uncertainty and feed it to the neural network.

Fig. 1. illustrates a sequence of 10 segments of varying durations, where the last two segments are short. Therefore, the confidence of the classification process should be lower than the case where the last 2 segments are long. Furthermore, the importance of the rest of the context segments is higher due to the large uncertainty accounted for the short segment s_{t-1}.

The theory motivating our architecture is as follows. Given a sequence of observed embeddings of length T, $x_1,\ldots,x_T$, we assume that x_t is drawn from a normal distribution with mean $\hat{x}_t$ and covariance matrix $\Lambda_t = \Lambda/d_t$ where d_t is the duration of segment t. Application of a linear function $f = \sum_t a_t x_t + b$ (such as a convolution layer without the nonlinearity), leads to $\mathrm{Cov}(f) = \Lambda\sum_t(a_t^2/d_t)$.

Following the above analysis, given durations $\{d_1,\ldots,d_T\}$ we first compute the reciprocal of the durations (we verified empirically that not using the reciprocal is suboptimal), and then feed the resulting sequence into a convolutional layer with the same configuration as used for the actual embeddings (but with separate learnt parameters due to a_t^2 replacing a_t). Finally, we concatenate the output of both CNNs and feed to the dense layers. The network is shown in the middle and r.h.s. of Fig. 2.

4.2. Modeling context relevancy

Looking back at Fig. 1, segments s_{t-1} and s_t (in green and red) are relevant to reaching a decision. A preceding segment $s_{t'}$ (in blue) may be of less relevance in case of a speaker change between $s_{t'}$ and s_{t-1}. The corresponding SCD scores for the context segments may indicate such relevancy. We first compute relevance probabilities $r_1,\ldots,r_T$ according to Equ. (2), where p_{SCD} is the softmax output of the baseline neural network (Section 3). We then feed the resulting sequence into a convolutional layer similarly as done for modeling embedding uncertainty. The network is shown in Fig. 2 (l.h.s.).

$$r_{T-1} = r_T = 1$$
$$r_{t<T-1} = r_{t+1} \cdot p_{SCD}(t+1) \quad (2)$$

5. EXPERIMENTS AND RESULTS

5.1. Datasets

We focus our experiments on a subset of the Hub4 broadcast news corpus [19, 20] containing 288 English shows sampled at 16KHz. The dataset is partitioned as detailed in Table 1. We further provide additional experiments on two additional datasets. LDC CALLHOME (CH) English Corpus [21] contains 113 two-speaker phone calls sampled at 8KHz and is partitioned into as detailed in Table 1.

Furthermore, a dataset named IBM-BN consisting of 130 hours of broadcast news in English (sampled at 16KHz) from 2017-2018 was collected and annotated internally in IBM. The dataset is partitioned as detailed in Table 1.

The tuning datasets (from Hub4 and CH) are used for assessing overfitting of the networks, for choosing optimal configurations for the networks, and for setting decision thresholds.

Table 1. *Datasets specifications (duration in hours)*

	Train	Tune	Test
Hub4	133	12	36
CH	70	2	6
IBM-BN	100	10	10

5.2. Metrics and optimization criteria

We use F-measure which is the harmonic average of recall and precision and is a common performance measure for SCD. A hypothesized change point is considered correct if it is within a "forgiveness collar" of a reference change point. We use a forgiveness collar of 0.25 seconds. As our SCD task is aligned to the STT task, we label the STT segments according to reference speaker labels (obtained from the ground truth) and use these labeled STT segments as ground truth.

For training the neural networks we minimize the total number of errors (either speaker change misdetection or false positives).

5.3. IBM-x-vector training

Our x-vector extractor shares the same DNN structure as described by Snyder et al. in [19, 29]. Training data is a combination of GALE training data, with some additional IBM internal wideband resources, and data extracted from the OpenSLR [30] and VoxCeleb1 [23]. It gives us a total of 173k recordings with 9.3k speakers. The training examples are 2–4 second speech segments

extracted from the input recordings. We train the DNN for 15 epochs with a minibatch size of 64.

5.4. Embeddings

For 8KHz data (CH) we used our 8KHz i-vector extractor. For 16KHz data (Hub4, IBM) we used our 16KHz i-vector extractor, Kaldi-x-vector and IBM-x-vector extractors. Table 2 reports an analysis of using different embedding extractors on Hub4. The results indicate the superiority of the x-vector extractors over the i-vectors extractor, with the IBM-x-vector obtaining significantly better results than the Kaldi-x-vector.

5.5. PLDA training

The PLDA backend for SCD was trained on 4.4k recordings with 3.7k speakers, from Hub4 training dataset and IBM internal wideband resources.

5.6. Systems

We evaluated our systems on Hub4 with the IBM-x-vector extractor. The results are reported in Table 3. Since the End-to-end work [27] was tested on a small subset of Hub4 (10 files), we also evaluated our best system on the same subset (denoted by test10). Results indicate the superiority of using a well-trained x-vector extractor and the benefit of using the CNN-based network to classify the 10-segment context. We also see significant improvements using uncertainty modeling.

We evaluated our systems also on the IBM-BN dataset, either using the Hub4 models (*mismatched*) or retraining on both Hub4 and IBM-BN-train (*matched*). Results are reported in Table 4 and indicate similar improvements for the *matched* condition (as for Hub4 experiments). For *mismatched*, the uncertainty modeling techniques seem to be less effective.

5.7. Cross domain experiments

We conducted cross-domain experiments in which a system is trained on CH (8KHz) and tested on Hub4 (16KHz) and vice-versa. Obviously, the CNN-embedding-based system did not perform very well. We evaluated the affinity matrix-based CNN system based on 16KHz i-vectors (Hub4) and tested on affinity matrices based on 8KHz i-vectors (CH) and vice-versa.

Table 5 reports the results which indicate that a CNN on the affinity matrix is robust to domain mismatch even when different embedding extractors are used for training and testing. For the i-vector extractor, accuracy was insensitive to cross-domain training (insignificant degradation for HUB4 testing). When using IBM-x-vector for the 16KHz HUB4 data and Kaldi-x-vectors for the 8KHz CH data, an absolute of 0.04 degradation was observed for HUB4 testing, and an improvement of 0.02 for CH testing.

5.8. Coverage and purity

The authors of [13] (reviewed in subsection 2.3 above) report results in terms of coverage and purity. We configured the NIST scoring tool [32] to compute these measures for our best system on Hub4 (Table 3a, F-measure=0.91). Our coverage is 90.5% and purity is 96.5%, This compares to the results in [13] of 84.4% coverage and 91.0% purity. Note that the evaluation datasets are different.

Table 2. *F-measure results as function of embedding extractor. Results are on Hub4 (16KHz)*

Method	Cosine F-measure	CNN-10 F-measure
i-vectors	0.70	0.83
Kaldi-x-vectors	0.73	0.85
IBM-x-vectors	0.80	0.88

Table 3. *F-measure results for Hub4 using IBM-x-vector*

Method	F-measure
Cosine	0.80
PLDA	0.82
CNN (Embeddings)	0.88
CNN (Affinity)	0.87
CNN (Embeddings) + Duration	0.90
CNN (Embeddings) + Relevancy	0.90
CNN (Embed.) + Duration + Relevancy	0.91
CNN (Embeddings) on test10	0.89
End-to-End on test10 [27]	0.53

Table 4. *F-measure results for IBM-BN using IBM-x-vectors. Training: mismatched= Hub4 matched= Hub4+IBM-BN-train*

Method	F-measure mismatched	F-measure matched
Cosine	0.75	0.75
PLDA	0.76	0.76
CNN (Embeddings)	0.80	0.81
CNN (Affinity)	0.81	0.81
CNN (Embeddings) + Duration	0.82	0.83
CNN (Embeddings) + Relevancy	0.81	0.82
CNN (Embed.) + Dur. + Relev.	0.82	0.84

Table 5. *F-measure results for cross domain experiments (matched reference performance is in parentheses). i-vector and x-vector extractors are 8Kz and 16Kz -based for CH and Hub4 respectively.*

Embedding	Method	CH→HUB4 F-measure	HUB4→CH F-measure
i-vector	**CNN Affinity**	0.78 (0.79)	0.73 (0.73)
x-vector	**CNN Affinity**	0.83 (0.87)	0.64 (0.62)

6. CONCLUSIONS

In this work we propose to do online speaker change detection by extracting speaker recognition-based embeddings (x-vectors) and modeling sequences of these embeddings. In our experiments, LSTM was significantly outperformed by CNN based approaches.

We propose to explicitly propagate input uncertainty information into the backend neural network. We address two types of uncertainties. The first one is uncertainty in embedding point estimate which is due to short and varying segment duration. The second type is uncertainty in relevancy of context embeddings.

Finally, we show in our experiments that the affinity matrix representation can be useful for speaker change detection and is quite robust to domain mismatch.

Overall, we obtained an F-measure of 0.91 and 0.84 on Hub4 and IBM-BN respectively, which significantly outperformed the PDLA baseline (0.82, 0.76) and the recently published end-to-end system (0.53 on Hub4-test10). In terms of coverage-purity we obtained (90.5%/96.5%) compared to other reported work (84.4%/91.0% on a different Broadcast dataset).

7. REFERENCES

[1]"Electronic Code of Federal Regulations: §79.1 Closed captioning of televised video programming", Available online: https://www.ecfr.gov/cgi-bin/retrieveECFR?gp=&SID=8d61c92a7719edd7584f6de60288be70&mc=true&n=sp47.4.79.a&r=SUBPART&ty=HTML#se47.4.79_11

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

[3] "Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge", G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, Matthew Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, S. Khudanpur, in Proc. *Interspeech* 2018

[4] H. Aronowitz, Y. Solewicz, O. Toledo-Ronen, "Online Two Speaker Diarization", in Proc. *Speaker Odyssey*, 2012.

[5] M. A. Siegler, U. Jain, B. Raj, and R. M. Stern, "Automatic segmentation, classification and clustering of broadcast news audio," in Proc. *DARPA Speech Recognition Workshop*, 1997.

[6] S. Chen and P. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the Bayesian information criterion," in Proc. *DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[7] H. Aronowitz, "Trainable Speaker Diarization", in Proc. *Interspeech*, 2007.

[8] B. Desplanques, K. Demuynck, and J.-P. Martens, "Factor analysis for speaker segmentation and improved speaker diarization,", in Proc. *Interspeech*, 2015.

[9] H. Bredin, "TristouNet: Triplet Loss for Speaker Turn Embedding," in Proc. *ICASSP*, 2017.

[10] S. H. Yella, A. Stolcke, and M. Slaney, "Artificial neural network features for speaker diarization," in Proc. *SLT*, 2014.

[11] M. Hr´uz, Z. Zaj´ıc, "Convolutional neural network for speaker change detection in telephone speaker diarization system", in Proc. *ICASSP*, 2017.

[12] M. Collet, D. Charlet, F. Bimbot, "Speaker tracking by anchor models using speaker segment cluster information," in Proc. *ICASSP*, 2006.

[13] R. Yin, H. Bredin, C. Barras, "Speaker Change Detection in Broadcast TV Using Bidirectional Long Short-Term Memory Networks", in Proc. *Interspeech*, 2017

[14] A. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for SVM-based speaker recognition", in Proc. *Interspeech*, 2016.

[15] IBM cloud speech to text service. Available online: https://www.ibm.com/watson/services/speech-to-text/

[16] D. Dimitriadis, P. Fousek, "Developing On-Line Speaker Diarization System", in Proc. *Interspeech*, 2017.

[17] W. Zhu, J. Pelecanos, "Online Speaker Diarization using Adapted I-Vector Transforms", in Proc. *ICASSP*, 2016.

[18] D. Snyder, D. Garcia-Romero, D. Povey, S. Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification", in Proc. *Interspeech*, 2017.

[19] "1996 English broadcast news speech (HUB4)", https://catalog.ldc.upenn.edu/LDC97S44.

[20] "1997 english broadcast news speech (HUB4), "https://catalog.ldc.upenn.edu/LDC98S71.

[21] A. Canavan, D. Graff, and G. Zipperlen, CALLHOME American English Speech LDC97S42. Web Download. Philadelphia: Linguistic Data Consortium, 1997.

[22] LibriVox, "LibriVox: Free public domain audiobooks," http://www.librivox.org, Accessed October 2018.

[23] A. Nagrani, J.S. Chung and A. Zisserman, "VoxCeleb: a large-scale speaker identification dataset", in Proc. *Interspeech*, 2017.

[24] J. Ajmera, I. McCowan, and H. Bourlard, "Robust Speaker Change Detection", in IEEE *signal processing letters*, 2004.

[25] S.H. K. Parthasarathi, M. Magimai-Doss, D. Gatica-Perez, H. Bourlard, "Speaker Change Detection with Privacy-Preserving Audio Cues", *Idiap-RR-23-2009*, 2009.

[26] "Speaker diarization using deep neural network embeddings", "D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree", in Proc. *ICASSP* 2017.

[27] L. Sarı, S. Thomas, M. Hasegawa-Johnson and M. Picheny, "Pre-training of speaker embeddings for low-latency speaker change detection in broadcast news", in Proc. *ICASSP* 2019.

[28] VoxCeleb Xvector System 1a. http://kaldi-asr.org/models/m8

[29] P. Kenny, T. Stafylakis, P. Ouellet, M.J. Alam and P.Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration", in Proc. *ICASSP*, 2013.

[29] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors: Robust embeddings for speaker recognition", in Proc. *ICASSP*, 2018.

[30] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books", in Proc. *ICASSP*, 2015.

[31] S. Prince and J. Elder, "Probabilistic linear discriminant analysis for inferences about identity", in Proc. ICCV, 2007.

[32] https://www.nist.gov/itl/iad/mig/tools