# Speaker Diarization Using Continual Learning

**Dibyojyoti Sinha(180244)**
dibyo@iitk.ac.in

**Garvit Bhardwaj(180262)**
garvitbh@iitk.ac.in

**Shivam Tulsyan(180723)**
shivtuls@iitk.ac.in

## 1 Abstract

We describe a way to leverage joint clustering and feature learning methods for the task of speaker diarization. We use Online Deep Clustering[1] to continually learn and as well as continually adapt our diarization system in an unsupervised way. Due to this, the system can also be trained using large and free data streams like YouTube and news broadcasts. Before performing continual unsupervised learning, we propose initialising the system's weights by pre-training it with a suitable amount of labelled data. This is done to direct the unsupervised training to distinguish between speakers. Since to separate different speakers, our model clusters the continually learned speaker representations, we need to assume that speakers take a turn speaking (no overlap).

## 2 Introduction

Speaker Diarization is the task of segmenting and co-indexing audio recordings by a speaker. The goal is to co-index segments that are attributed to the same speaker. In simpler terms, it is the task to find out "who spoke when".

Diarization has many applications in speaker indexing, retrieval, speech recognition with speaker identification, diarizing meeting and lectures. With the increasing number of broadcasts, meeting recordings and voice mail collected every year, speaker diarization has received much attention from the speech community. An essential application of speaker diarization is the cocktail party problem [2]. This problem can be transcribed as a simple question asking, "How do we recognize what one person is saying when others are speaking at the same time?". The root of the cocktail party problem is that the human voices present in a noisy social setting often overlapped in frequency and time. Thus, these signals represent sources of direct acoustic interference and "energetic masking" that can impair the perception of speech. Diarization models can be used to solve this problem efficiently.

The speaker boundaries obtained by diarization can be used to improve acoustic speech recognition accuracy significantly. A typical diarization system usually consists of four components:

- Dividing the input audio into small segments that are assumed to have a single speaker, we filter out any non-speech fragments(pauses in this case).
- Embedding extraction, where features such as MFCC's, i-vectors[3] etc. are extracted.
- Clustering, wherein the number of speakers is determined and extracted embedding are clustered into these speakers.
- Final step is re-segmentation(which is sometimes optional) wherein clustering results are further refined to produce more accurate results.

Preprint. Under review.

In the clustering steps mentioned above models like mean shift [4], K-Means[5], spectral clustering[6] and supervised Bayesian non-parametric models can be used. For re-segmentation, recent advancements have been implementing Viterbi[7] and factor analysis subspace[8].

State of the art models usually require large data-sets to train their embeddings and clustering modules. However, the availability of training set can be restricted and limited. Many of these models can't be applied to out-of-distribution environments.

To tackle these problems, we want the system to learn continually. Here the examples are available one by one, and the agents learn to detect the speaker's identity on the fly. Doing this we can ensure that a realistic speaker diarization system can:

- not require user data before deployment
- allow new user to be detected in real time
- run efficiently without pre-training on large amounts of data.

## 3 Problem Statement

To perform speaker diarization using continual unsupervised representation learning. We aim to get a new high-performance continual speaker diarization system which works faster and better than real-time ones. It performs all the major tasks of a speaker diarization system, but the within segment speaker change detection.

## 4 Dataset and Pre-Processing

The dataset that we have used is the ICSI meeting corpus[9] [10]. It is an audio data set consist originally recorded on close-talking microphones. We extracted the ground truth labels for the speech segments corresponding to different speakers(which speaker spoke in a particular segment) to evaluate our model's performance from a given annotation file.
To train the gender-dependent GMM models, audios from the first three meetings are used in a supervised fashion and evaluated on the data from the fourth meeting.
All audio data were transformed into 26-dimensional feature vectors consisting of 13 MFCC coefficients and their first derivatives.

## 5 Baseline Model

In the baseline model[11] we implemented a forever learning-based system for continual speaker diarization. Based on the observation that most speaker changes occur during the non-speech regions[12], we assume that each segment belongs to a single speaker. In this, we have used a standard model for voice activity detection[13], and implemented a set of modules including novelty detection(determining whether a speaker is new or old) and classification for the identity and gender of speakers. Across each module, a set of GMM modules is shared(male and female speakers, and each speaker). We have pre-trained two GMMs to determine the gender of speakers in a supervised manner.
There are a variable number of speaker GMM's. Block arrows show how modules share these models. Unsegmented audio is fed to the voice activity detection module, which outputs speech start and endpoints. For each frame, the non-speech and speech (the better one from the two GMMS) likelihoods are passed through two separate median filters and the frame's label (speech / non-speech) is assigned by comparing the output of the filter. Then, a simple logic decides segments start and endpoints taking into account such requirements as the minimum segment length (MSL), maximum pause in segment (MPS) and maximum speech in pause (MSP).

Now during the speaker diarization process, for the current speech segment, as soon as the start point is decided, the novelty detection module accumulates likelihoods from all the GMM's. This time called decision time is the system latency time. Then based on these, it is decided whether the speech segment comes from a new speaker or the already existing one(old speaker). This is a

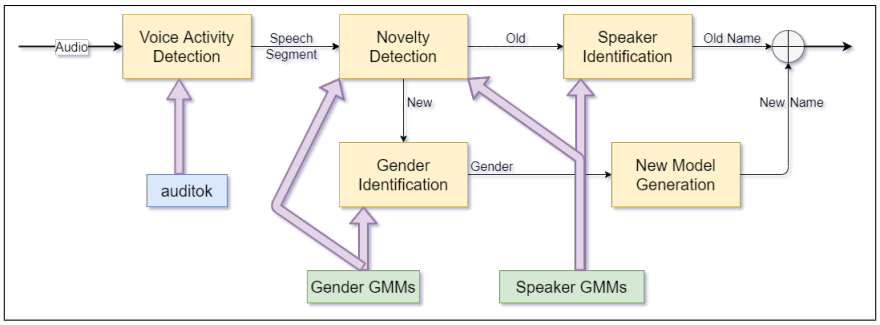


Figure 1: Block diagram of the system

typical hypothesis testing problem, where the standard solution is the likelihood ratio test. It is formulated as follows

$$X \in \begin{cases} \omega_o & L(X) > \theta \\ \omega_1 & L(X) < \theta \end{cases} \tag{1}$$

where $X = \{x_i\}$, i=1,...., DL is a decision length speech segment, here $\omega_0$ corresponds to the hypothesis of an old speaker, and $\omega_1$ for the new speaker.
There are various ways to define $p(X|\omega_i)$. Considering the available set of GMMs, a straightforward approach is to define them as:

$$p(X|\omega_0) = P_{sp} = \max_{\lambda_j \in \Lambda} p(X|\lambda_j) \tag{2}$$

$$p(X|\omega_1) = P_{gen} = \max(p(X|\lambda_{male}), p(X|\lambda_{female})) \tag{3}$$

where $\Lambda = \{\lambda_j\}$ is the current set of speaker GMMs. An alternate approach to define $p(X|\omega_i)$

$$p(X|\omega_1) = P_{ave} = \frac{1}{n-1}(\sum_j p(X|\lambda_j) - P_{sp}) \tag{4}$$

Here $n = |\Lambda|$ is the size of the speaker set, and the quantity is defined as the average of all model likelihoods of the speaker set except for the winning one. The final likelihood ratio is

$$L(X) = \frac{P_{sp}^2}{P_{gen} P_{ave}} \tag{5}$$

When a new speaker is encountered, their gender is identified, and then, from the corresponding gender GMM, a new GMM is spawned by copying its parameters.
This GMM is continually learned using the speech segment data, and from this point, it is used to represent the new speaker. This is used to produce all the individual speaker models. In the case of an old speaker, s/he is identified, and the corresponding GMM is again continually learned.

This step is the one that allows the whole system to operate continually and makes it different from all other systems. The main algorithm for off-line GMM parameter estimation is the Expectation-Maximization (EM) algorithm[14]. The online EM[15] converges faster than the standard EM, but even a few iterations could increase too much the computational load for a real-time system. On the other hand, given an infinite number of data drawn from the same distribution, the online EM can be considered as a stochastic approximation[16]. In practice, this means that as long as there is enough data, model parameters can be approximated in a single pass.

The learning rate $\eta(t)$ (where t is time) in this should satisfy the following conditions:

$$\lim_{t \to \infty} \eta(t) = 0 \qquad \sum_{t=1}^{\infty} \eta(t) = \infty \qquad \sum_{t=1}^{\infty} \eta^2(t) < \infty \tag{6}$$

thus the commonly used function for $\eta(t)$ is:

$$\eta(t) = \frac{1}{at + b} \qquad 1 > a > 0 \tag{7}$$

Here b sets the learning speed for newer samples, representing the forgetting speed of past samples. This algorithm allows fast and inexpensive continual learning of the system GMM's.

## 6 Online Deep Clustering

Joint clustering and various feature learning techniques have shown commendable performance in unsupervised learning tasks. The training schedule, which alternates between feature clustering and network parameter updates, leads to unstable learning of speaker embeddings. To address this issue, we propose Online Deep Clustering (ODC)[1], which performs clustering and network updates concurrently rather than alternatively.

The critical insight we gained is that the cluster centroids should evolve steadily to keep the classifier stable.

Initially, we take the MFCC vector representation of a speaker and feed it into a deep network to obtain embeddings. The clustering will be performed on the obtained embeddings. The loss function used in the deep network's training is the generalized end to end(ge2e) loss [17].

### 6.1 GE2E Training

GE2E[17] training is based on processing a large number of utterances at once, in a batch that contains N speakers and M utterances from every speaker on average. Each batch has a size $N \times M$. Each feature vector $x_{ji}(1 \leq j \leq N$ and $1 \leq i \leq M)$, we feed the features into an LSTM network. A linear layer is connected to the last LSTM layer as an additional transformation of the last frame response of the network. We denote the output of the entire neural network as $f(x_{ji}; w)$, w denotes the parameters of the neural network(including LSTM and linear layers). The embedding vector is defined as the L2 norm of the network output:

$$\boldsymbol{e}_{ji} = \frac{f(\boldsymbol{x_{ji}}, \boldsymbol{w})}{||f(\boldsymbol{x_{ji}}, \boldsymbol{w})||_2} \tag{8}$$

Here $e_{ji}$ represents the embedding vector of the $j_{th}$ speaker's $i_{th}$ utterance. The centroid of the embedding vectors from the jth speaker $[e_{j1}, ..., e_{jM}]$ is defined as $c_j$ as follows:

$$c_k = E_m[\boldsymbol{e}_{km}] = \frac{\sum_{m=1}^{M} \boldsymbol{e}_{km}}{M} \tag{9}$$

The similarity matrix $S_{ji,k}$ is defined as the scaled cosine similarities between each embedding vector $e_{ji}$ to all the centroids $c_k(1 \leq j, k \leq N$ and $1 \leq i \leq M)$ :

$$\boldsymbol{c}_j^{(-i)} = (1+\epsilon) \cdot \boldsymbol{c}_j - \epsilon \cdot \boldsymbol{e}_{ji} \qquad \text{where } \epsilon \text{ is a small number, eg. } 0.01 \tag{10}$$

$$\boldsymbol{S}_{ji,k} = \begin{cases} w \cdot \cos\left(\boldsymbol{c}_{ji}, \boldsymbol{c}_j^{(-i)}\right) + b & \text{if } k = j. \\ w \cdot \cos\left(\boldsymbol{c}_{ji}, \boldsymbol{c}_k\right) + b & otherwise. \end{cases} \tag{11}$$

where w and b are learnable parameters. We constrain the weight to be positive w > 0, because we want the similarity to be larger when cosine similarity is larger. GE2E builds a similarity matrix that defines the similarities between each $\boldsymbol{e_j}$, and all centroids $\boldsymbol{c_k}$

Given an embedding vector $\boldsymbol{e_{ji}}$, all centroids $\boldsymbol{c_k}$, and the corresponding similarity matrix $\boldsymbol{S}_{ji,k}$. We put a softmax on $\boldsymbol{S}_{ji,k}$ for k=1,...,N that makes the output equal to 1 iff $k = j$, otherwise make the output equal to 0. The loss on each embedding vector $\boldsymbol{e}_{ji}$ could be defined as:

$$L(\boldsymbol{e}_{ji}) = -\boldsymbol{S}_{ji,j} + log \sum_{k=1}^{N} \exp \boldsymbol{S}_{ji,k} \tag{12}$$

$$L_{ge2e} = \sum_{j,i} L(\boldsymbol{e}_{ji}) \tag{13}$$

### 6.2 ODC terminology

We specifically design and maintain two dynamic memory modules: samples memory, which stores labels and embeddings for samples, and centroids memory. The samples memory contains N nearest speakers to the centroid for every cluster.

### 6.3 Clustering Algorithm

We use an unsupervised clustering algorithm to assign labels to the feature vectors outputted by the model. Consider the following algorithm to cluster vectors

**Algorithm 1:** Hybrid Speaker Clustering

**Initialize** $\theta_U$, $\theta_L$
**for** ( *every new embedding* $\mathcal{E}$ ) **do**
  j $\leftarrow \arg\underset{\text{i}}{min}$ cos_Dist($\mathcal{E}$, $\mathcal{C}_\text{i}$)
  min_D $\leftarrow$ cos_Dist($\mathcal{E}$, $\mathcal{C}_\text{j}$)

  G1 $\leftarrow$ G($\mathcal{E}$ is merged with $\mathcal{C}_\text{j}$)
  G2 $\leftarrow$ G($\mathcal{E}$ is assigned to a new cluster)

  **if** ( *min_D* $< \theta_L$ ) **then**
    Merge $\mathcal{E}$ with cluster $\mathcal{C}_\text{j}$
  **else if** ( *min_D* $> \theta_U$ ) **then**
    Assign $\mathcal{E}$ to a new cluster
  **else if** ( *G1* $<$ *G2* ) **then**
    Merge $\mathcal{E}$ with cluster $\mathcal{C}_\text{j}$
  **else**
    Assign $\mathcal{E}$ to a new cluster

  **Update Parameters Accordingly**
  **return** label

where $\theta_U$ and $\theta_L$ are the upper and lower thresholds of the low confidence region, the region beyond this limits is high confidence region. And G is the within-cluster dispersion scaled by the number of clusters for selection of clusters for new embeddings, and is denoted by :

$$G(\cdot) = \mid \sum_{j=1}^{c} (N_j \Sigma_j) \mid \sqrt{c} \tag{14}$$

where $\mid \cdot \mid$ denotes the determinant, c is the number of clusters, $N_j$ and $\Sigma_j$ is the number of embeddings and co-variance matrix corresponding to a cluster $j$.

This is a combination of K-means and Dispersion based clustering. Firstly, for every new speaker embedding, its closest cluster is determined by computing its cosine distance from each existing cluster centroid. Then the minimum distance among these is stored, and we work upon confidence intervals to justify our hypothesis.

#### 6.3.1 Confidence

Confidence metric which tells us whether we need to do simple leader follower based clustering[18] or dispersion based clustering[19]. If the confidence is high, we directly perform clustering based on the leader for the current embedding, assigning it to the nearest centroid. However if the confidence is low, then dispersion of this embedding will be calculated, based on which the clustering will be performed.

#### 6.3.2 Role of Thresholds

If the cosine distance obtained is lower than $\theta_L$ then the embedding belongs to the high confidence region, thus it is assigned to its closest cluster. Alternatively, if this distance is higher than $\theta_U$, then again the embedding belongs to the high confidence region and the embedding is assigned to a new cluster. If the cosine distance is between $\theta_L$ and $\theta_H$, then the embedding belongs to the low confidence region and hence, its dispersion will be used to allocate a cluster to it.

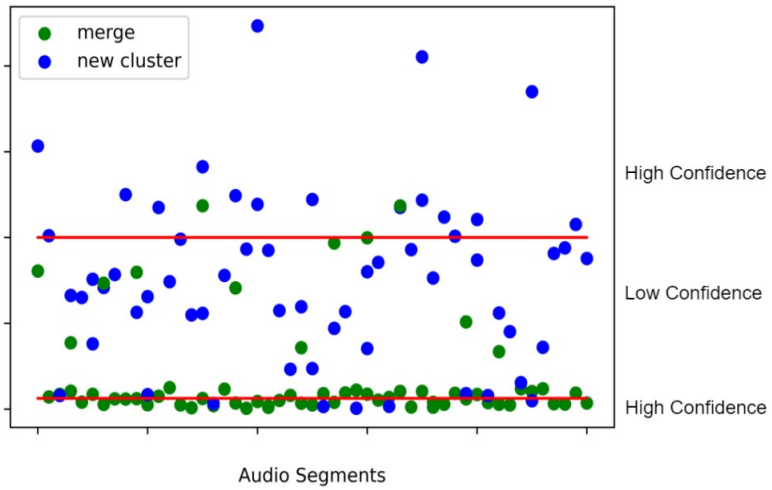


Figure 2: Confidence regions based on cosine distance

#### 6.3.3 Dispersion Based Clustering

Dispersion is analogous to entropy for an embedding. We calculate dispersion[19] of the clusters corresponding to the following cases :

- G1: Dispersion corresponding to merging the embedding to its closest cluster
- G2: Dispersion corresponding to assigning a new cluster to this embedding

Then these two dispersion values are compared. The operation performed is corresponding to the one having lower value.

### 6.4 Lifelong Training

The deep network used is a three-layer stacked BiLstm[20] with a final global averaging layer. This model takes input an equivalent duration of 2 seconds of audio as MFCC features and outputs a single feature vector.
We have pre-trained this model to distinguish between 18 speakers from the first few shows in the ICSI meeting corpus. The pretraining process was two-step. First, the model is trained using cross-entropy loss by adding an l2-normalization, affine and softmax layer after the global averaging layer. Next, we remove the affine and softmax layers and continue training the model using triplet loss [21].
The deep network is adapted on the fly by using the labels from the clustering step. Due to machine limitations, we cut the one-hour audio recordings into 10-minute segments and feed these segments sequentially to the model. We had used the Generalized End to End Loss function[17] and RMSprop optimizer[22] to adapt the trainable parameters. Also, we regularized the weights using the original pre-trained weights such that the adapted weights are near the original weights.

$$L = L_{ge2e} + \lambda \cdot \sum_{w_{model}} (w_{new} - w_{pre\_trained})^2 \tag{15}$$

The above update helps to reduce catastrophic forgetting. Moreover, we also maintain a memory which stores the nearest input features to the detected speaker centroids and their predicted labels to back-propagate on these stored samples.

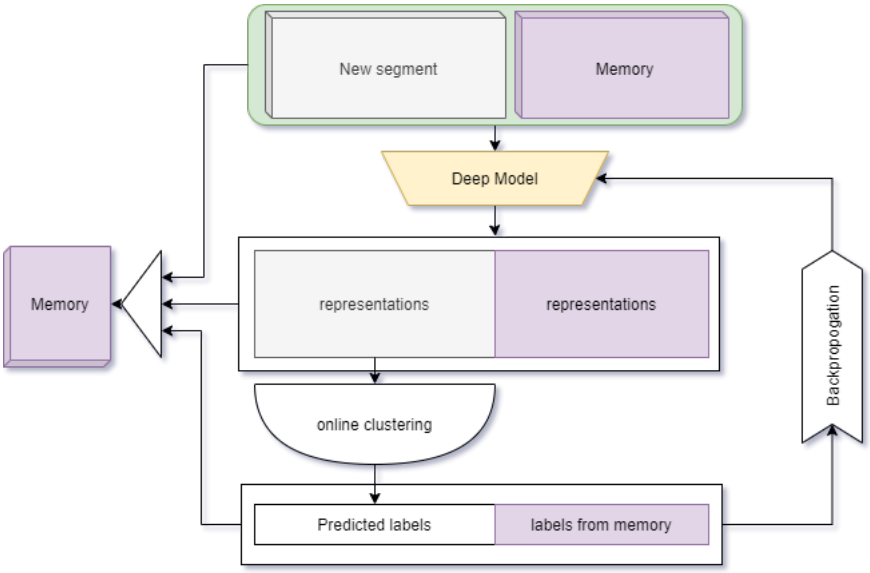


Figure 3: Model Architecture

We keep expanding the memory by adding such input features and their predicted labels. When the memory exceeds a certain size, we start removing the oldest sample for each new sample added.

### 6.5 Loss Re-weighing

To avoid the training from collapsing into a few enormous clusters, we implement a DC model with loss re-weighing. We empirically find that the performance remained unchanged when the weight follows $w_c \propto \frac{1}{\sqrt{N_c}}$, where $N_c$ denotes the number of samples in class c. Hence, we adopt the same technique for ODC. With this, samples in smaller clusters contribute more towards backpropagation, pushing the decision boundary farther to accept more potential samples.

## 7 Evaluation Scheme

The evaluation metric is the speaker diarization error rate (DER)[23]. The DER is a time-weighted sum of miss errors, false alarms and speaker errors. Since we have used a separate model for Voice activity Detection, the DER is modified to show only speaker errors. This gives us the final metric as

$$\text{DER} = \frac{\text{confusion time}}{\text{total reference time}} \tag{16}$$

where confusion time denotes the total time across the segments where our model predicted an incorrect speaker.

## 8 Results

We have evaluated the performance for the model under lifelong training on 376 audio segments of 10 minutes (called Batches) each, which is about 60 hours of audio. Below is a graph plotting the DER as the lifelong learning model is being adapted on the fly.

| Model | DER (%) |
|---|---|
| online learning GMM (baseline) | 58.9 |
| lifelong learning ODC | 42.5 |

We can see that after the 150th batch the DER shows a decreasing trend. The average DER is 0.4246 (42.46%)

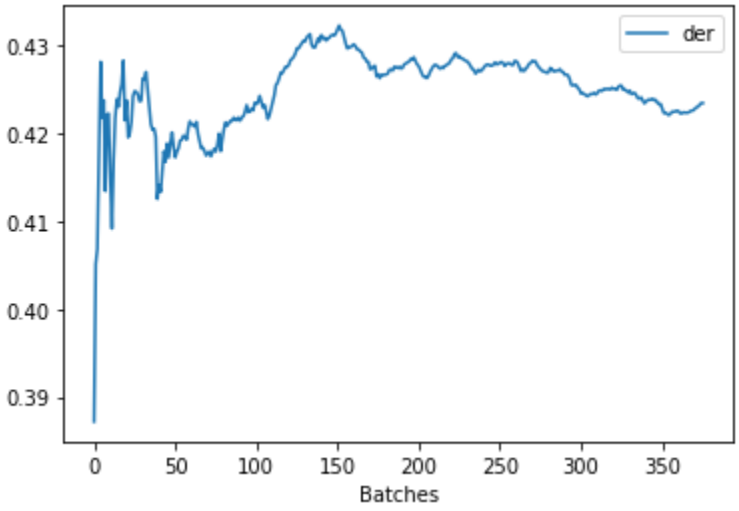


Figure 4: Cumulative DER across batches

## References


[1] Xiaohang Zhan et al. "Online deep clustering for unsupervised representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6688–6697.

[2] Simon Haykin and Zhe Chen. "The cocktail party problem". In: *Neural computation* 17.9 (2005), pp. 1875–1902.

[3] Daniel Garcia-Romero and Carol Y Espy-Wilson. "Analysis of i-vector length normalization in speaker recognition systems". In: *Twelfth annual conference of the international speech communication association*. 2011.

[4] Yizong Cheng. "Mean shift, mode seeking, and clustering". In: *IEEE transactions on pattern analysis and machine intelligence* 17.8 (1995), pp. 790–799.

[5] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. "The global k-means clustering algorithm". In: *Pattern recognition* 36.2 (2003), pp. 451–461.

[6] Andrew Ng, Michael Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 14 (2001), pp. 849–856.

[7] G David Forney. "The viterbi algorithm". In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[8] Gregory Sell and Daniel Garcia-Romero. "Diarization resegmentation in the factor analysis subspace". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 4794–4798.

[9] Adam Janin et al. "The ICSI meeting corpus". In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).* Vol. 1. IEEE. 2003, pp. I–I.

[10] Rajdip Dhillon et al. *Meeting recorder project: Dialog act labeling guide*. Tech. rep. INTERNATIONAL COMPUTER SCIENCE INST BERKELEY CA, 2004.

[11] Konstantin Markov and Satoshi Nakamura. "Never-ending learning system for on-line speaker diarization". In: *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*. IEEE. 2007, pp. 699–704.

[12] Daben Liu and Francis Kubala. “Fast speaker change detection for broadcast news transcription and indexing”. In: *Sixth European Conference on Speech Communication and Technology*. 1999.

[13] Amine SEHILI. *auditok*. `https://github.com/amsehili/auditok`. 2021.

[14] Lei Xu and Michael I Jordan. “On convergence properties of the EM algorithm for Gaussian mixtures”. In: *Neural computation* 8.1 (1996), pp. 129–151.

[15] Masa-aki Sato and Shin Ishii. “On-line EM algorithm for the normalized Gaussian network”. In: *Neural computation* 12.2 (2000), pp. 407–432.

[16] CS Kubrusly and J Gravier. “Stochastic approximation algorithms and applications”. In: *1973 IEEE conference on decision and control including the 12th symposium on adaptive processes*. IEEE. 1973, pp. 763–766.

[17] Li Wan et al. “Generalized end-to-end loss for speaker verification”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4879–4883.

[18] Richard O Duda, Peter E Hart, and David G Stork. “Pattern classification 2nd ed”. In: *John Willey & Sons Inc* (2001).

[19] Hubert Jin, Francis Kubala, and Rich Schwartz. “Automatic speaker clustering”. In: *Proceedings of the DARPA speech recognition workshop*. Citeseer. 1997, pp. 108–111.

[20] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. “Bidirectional LSTM networks for improved phoneme classification and recognition”. In: *International conference on artificial neural networks*. Springer. 2005, pp. 799–804.

[21] Sergey Novoselov et al. “Triplet Loss Based Cosine Similarity Metric Learning for Text-independent Speaker Recognition.” In: *Interspeech*. 2018, pp. 2242–2246.

[22] Thomas Kurbiel and Shahrzad Khaleghian. “Training of deep neural networks based on distance measures using RMSProp”. In: *arXiv preprint arXiv:1708.01911* (2017).

[23] Olivier Galibert. “Methodologies for the evaluation of speaker diarization and automatic speech recognition in the presence of overlapping speech.” In: *INTERSPEECH*. 2013, pp. 1131–1134.