



The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP7704

Real-time Speaker Recognizer

Submitted in partial fulfillment of the requirements for the admission to  
the degree of Master of Science in Computer Science

By

YU Chuan

3035562485

Dr. Beta C.L. Yip

Date of submission: 10/04/2020

## **Abstract**

Speaker recognizer is a system that recognizes and labels the speakers in an audio file or live stream. In addition to help save the audio data in well-structured way, it could improve the accuracy of acoustic speech recognition (ASR). In this dissertation, we first use the voice activity detector (VAD) developed by Google to exclude the non-human speech. After we get the voiced audio data, we extract the spectrogram from the audio data via short-time Fourier transform (STFT) and then split the audio data into non-overlapping segments which contain 40 frames corresponding to 415ms which is also the resolution of speaker recognizer. Based on the spectrogram, we utilize the Net “Vector of Locally Aggregated Descriptors” (NetVLAD) embedding extraction technique to obtain the embeddings of each segment. Then we apply offline spectral clustering algorithm or online Links clustering algorithm to cluster the segments according to their embeddings and get the speaker label of each segment. Finally we use the pyannote.metrics python module to evaluate our system. The diarization error rate (DER) of our offline speaker recognizer system is 26% and the percentage of the most important part Confusion is only 8.7% while the DER of our online speaker recognizer is 38% and the percentage of the Confusion part is 20.5%.

## **Declaration**

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed . . . . .

## **Acknowledgments**

First and foremost, I would like to express my deepest appreciation to my supervisor, Dr. Beta C.L. Yip. It was a great experience to work with him. Dr. Beta C.L. Yip supervised me for one year and taught me kinds of techniques and theories to carry out the research and to present the research works as clearly as possible. It was a great honor to work and study under his guidance. Without his patient guidance and continuous support, I could not have achieved this stage and finished my dissertation.

My great gratitude also goes to my second examiner Dr. H.F. Ting. It was an impressive experience to learn a lot of knowledge and suggestions related with my dissertation project especially how to think critically.

I also appreciate my classmates and friends in the Computer Science Department, those who helped me a lot in my study and life in Hong Kong. I also appreciate the department for providing me such an excellent academic environment to fulfill myself in the Master of Science program study. Thank Ms. Priscilla Lam for her kind help during my study.

Finally, I would like to express my appreciation and gratitude to my parents for their consistent support. And also thank Miss. Le YU who accompanies me all the time and help me overcome any difficulties with

caring and loving me.

## Table of Contents

<b>Abstract</b> .....	i
<b>Declaration</b> .....	ii
<b>Acknowledgments</b> .....	iii
<b>Table of Contents</b> .....	v
<b>1. Introduction</b> .....	1
<b>2. Analysis of problem</b> .....	6
<b>3. Theoretical principle</b> .....	9
<b>3.1. Voice activity detector</b> .....	9
<b>3.2. Audio preprocessing</b> .....	11
<b>3.2.1. Waveform</b> .....	14
<b>3.2.2. Spectrum</b> .....	14
<b>3.2.3. Spectrogram</b> .....	14
<b>3.2.4. Mel-spectrogram</b> .....	15
<b>3.3. Net “<i>Vector of Locally Aggregated Descriptors</i>” embedding     extraction</b> .....	15
<b>3.3.1. Bag of figure</b> .....	16
<b>3.3.2. NetVLAD network architecture</b> .....	17
<b>3.3.3. Datasets</b> .....	20
<b>3.3.4. Equal error rate</b> .....	20
<b>3.4. Links clustering algorithm</b> .....	22
<b>3.4.1. High-dimensional approximation</b> .....	22

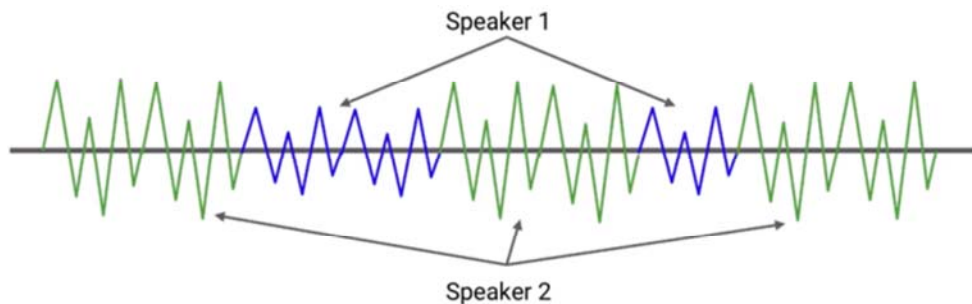
3.4.2.	Estimated distribution .....	23
3.4.3.	Internal representation .....	24
3.4.4.	Links algorithm .....	25
3.5.	Spectral clustering algorithm .....	28
4.	Model of investigation.....	31
4.1.	Mel-frequency cepstral coefficients .....	31
4.1.1.	Cepstral analysis .....	31
4.1.2.	MFCC.....	33
4.2.	LSTM.....	34
4.2.1.	Generalized End-to-End loss.....	35
4.2.2.	Model architecture .....	37
4.3.	K-means.....	38
4.3.1.	K-means algorithm .....	38
4.3.2.	Limitations of K-means .....	39
5.	Design and construction of software system .....	41
5.1.	Overview.....	41
5.2.	VAD .....	43
5.3.	Preprocessor.....	45
5.3.1.	Speech feature analysis .....	46
5.3.2.	Speech segmentation .....	46
5.4.	NetVLAD.....	47
5.4.1.	GhostVLAD .....	47

5.4.2.	Detailed parameters setting.....	47
5.5.	Spectral clustering .....	48
5.6.	Links clustering .....	48
6.	Experimental result.....	51
6.1.	Evaluation metrics.....	51
6.2.	VAD impact on DER.....	54
6.3.	Offline speaker diarization result.....	55
6.4.	Online speaker diarization result .....	57
7.	Discussion and future works .....	61
7.1.	Embedding extraction algorithm selection.....	61
7.2.	Multi-persons speech .....	63
7.3.	K-means vs. Spectral clustering. ....	64
7.4.	Decrease Missed Detection and False Alarm .....	65
7.5.	Training NetVLAD embedding algorithm by ourselves .....	67
7.6.	Increase the labeled audio file number .....	68
8.	Conclusions .....	69
9.	Reference.....	72



## 1. Introduction

Speaker recognizer is a system that recognizes and labels the speakers in an audio file or live stream. The process of recognizing and labeling the speakers is also called speaker diarization. More concretely, speaker diarization is the process of partitioning an input audio stream into homogeneous segments according to the speaker identity [1]. It aims to answer the question “who spoke when”.

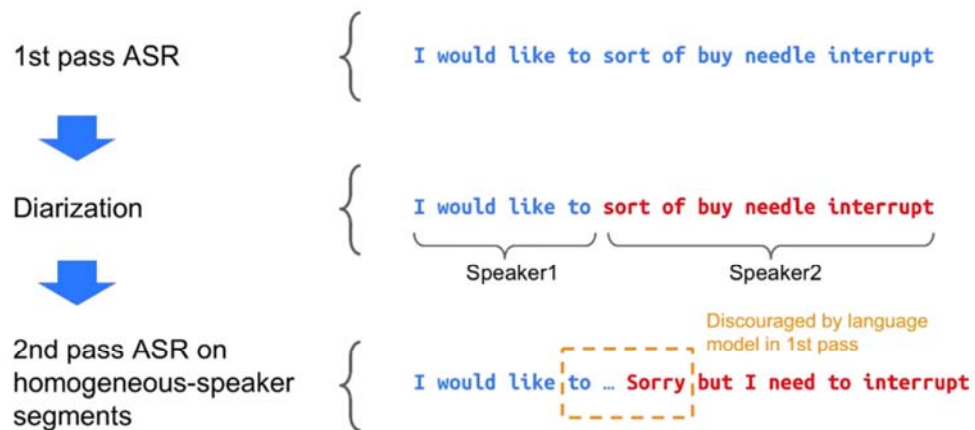


**Figure 1: Function of Speaker Recognizer. This figure is adopted from Youtube video “Google’s Diarization System: Speaker Diarization with LSTM” and modified by ourselves.**

For example, in Figure 1, given an audio file, speaker recognizer can distinguish how many speakers speaker and when the speakers start and stop talking.

There are lots of applications of speaker recognizer. In the area of medical recording, speaker recognizer could help separate doctor and patient conversation in order to storage the audio data in more well-structured way. It could also distinguish different speaker utterances to automatically generate

notes for meetings based on speaker identity. In addition, it could separate customers and sales audio records in the call center to improve the efficiency of analysis.



**Figure 2: Hero application of speaker recognizer. This figure is adopted from Youtube video “Google's Diarization System: Speaker Diarization with LSTM” and modified by ourselves.**

More importantly, the hero application of speaker recognizer is to improve the acoustic speech recognition (ASR). ASR is the process of converting speech to text. As illustrated in Figure 2, without speaker recognizer or diarization, the text converted by ASR may make no sense. With the help of diarization, ASR could convert different speaker speech to text separately. These speaker boundaries could help improve the accuracy of ASR.

A speaker recognizer usually consists of five components:

(1) Voice activity detection

Voice activity detector (VAD) is a module used in audio signal processing in which absence or presence of human speech is detected. VAD is used to exclude the non-human speech for further processing.

(2) Speech feature analysis

The audio data is normally the waveform of a signal. In order to acquire more information such as frequency, the waveform is transformed to spectrogram or Mel-spectrogram for further analysis.

(3) Speech segmentation,

The input audio signal is segmented into short segments that are supposed to have a single speaker and the length of segment is also the resolution of speaker recognizer.

(4) Audio embedding extraction

In mathematics, embedding is a mapping  $f: X \rightarrow Y$  which has two properties: injective and structure-preserving (if  $X_1 < X_2$ , then  $Y_1 < Y_2$ ).

For audio embedding, it aims to map the audio data to  $n$ -dimensional vectors for further analysis.

It is reported that specific features such as Mel-frequency cepstral

coefficients (MFCCs)[2], speaker factors[3], or i-vectors[4-6] are extracted from the audio data.

(5) Clustering,

The number of speakers is determined and audio embeddings are clustered into these speakers based on speaker identity.

In recent years, with the rapid development of neural network, more and more neural network based audio embedding extraction techniques are proposed in speaker verification (SV) area [7-11]. They are reported to significantly outperform previously state-of-the-art techniques such as i-vectors.

SV is the process of verifying whether an audio utterance belongs to a specific speaker based on speaker's known utterances which is also called enrollment utterances, with applications such as voice match.

Based on the restrictions of the enrollment and verification process, SV usually fall into one of two categories: text-dependent speaker verification (TD-SV) and text-independent speaker verification (TI-SV). In TD-SV, enrollment and verification utterances transcript is phonetically constrained such as "Hey, Siri", which means speaker embeddings are extracted from

specific keywords [12, 13]. While in TI-SV, there are no restrictions on the transcript of the enrollment or verification utterances. Obviously, speaker recognizer requires TI-SV audio embedding extraction techniques which work on arbitrary speech.

In this dissertation, we combine Net “*Vector of Locally Aggregated Descriptors*” (NetVLAD) audio embedding extraction algorithm, Links online clustering algorithm to develop our own real-time speaker recognizer system. In addition to the real-time speaker recognizer, we also take advantage of spectral clustering algorithm to develop our offline speaker recognizer which means the speaker labels are produced when the whole speech audio file is available.

## **2. Analysis of problem**

In this section, we analyze the aims and difficulties of components included in the real-time speaker recognizer and will illustrate how we overcome these difficulties in the following sections. As we mentioned in the section 1, the key components of the whole speaker recognizer are the voice activity detector (VAD), preprocessor, audio embedding extraction and clustering algorithm.

VAD aims to classify the audio data as speech or non-speech. Usually, VAD first extracts the feature vectors from each frame (Frame refers to a very short time such as 25ms) from the audio data and uses the classification rules to determine whether it is the speech or non-speech. The difficulty of VAD is how to improve its accuracy based on the result of each frame. In the section 5.2, we illustrate that we use the deque data structure to cache the fix number of frames of audio data and determine the final result based on the percentage of the results of all the cached frames.

The audio data is normally the waveform of a signal. In order to acquire more useful information in the audio such as frequency, the waveform is transformed to spectrogram or Mel-spectrogram for further analysis. This is the main function of the preprocessor. In this dissertation, we obtain the spectrogram from the audio data via short-time Fourier transform (STFT).

Another function of preprocessor is to segment the speech. The input audio signal is segmented into short segments that are supposed to have a single speaker. The time length of each segment is 415ms which covers 40 frames. The detailed feature analysis and speech segmentation will be explained in Section 5.3.

The purpose of audio embedding extraction is to extract the unique properties of each segment (Segment refers to several continuous frames). The unique property usually is a 1-D feature vector which has the ability to represent the corresponding segment. Lots of embedding extraction techniques is proposed so far. Some of them are traditional ones such as Mel-frequency cepstral coefficients (MFCC) while some of them are derived from deep neural networks. How to choose the appropriate one which could represent the segment best is one of the difficulties in this part. In section 3.3, we will explain the principle of the embedding extraction technique we use in our speaker recognizer. And in section 4, we will propose other two embedding extraction techniques and compare them in section 7.

After we get the feature vector of each segment, we need to use the clustering algorithm to determine how many speakers there are and which segment belongs to which speaker. The dimensionality of our feature vector is 512. When the dimensionality increases, data becomes increasingly sparse in the

space that it occupies which means the critical factor distance in most of clustering algorithms (such as K-means) becomes less meaningful. We overcome this difficulty via two algorithms proposed in 2018 and illustrate the principle of these two algorithms in the section 3.

Finally, we will integrate these components into our speaker recognizer system and use one of common evaluation strategies (Diarization error rate) to evaluate our system.



### **3. Theoretical principle**

In Section 3, we introduce all the basic knowledge, theoretical principle and detailed algorithms related to our real-time and offline speaker recognizer including basic audio preprocessing, voice activity detector principle, Net “*Vector of Locally Aggregated Descriptors*” embedding extraction algorithm[14], Links online clustering algorithm[15] and spectral offline clustering algorithm[1]. Due to the development property of this dissertation, we focus on the speaker recognizer building. Therefore the algorithms we use are almost proposed by other groups and we re-implement these algorithms in our speaker recognizer.

In addition to the algorithms we choose in our speaker recognizer system, there are lots of other algorithms related to the embedding extraction and the clustering. We also introduce some typical algorithms and principle in Section 4 and discuss the advantages and disadvantages of these algorithms in Section 7.

Section 3 and Section 4 only introduce the principles and algorithms piece by piece. We will integrate them and build our system in Section 5. The detailed parameters and the workflow will also be included in Section 5.

#### **3.1. Voice activity detector**

Voice activity detector (VAD) is a module used in audio signal processing in which absence or presence of human speech is detected[16]. More specifically, the absence of human speech includes not only the silence or quiet audio segments but also noise or any sound not related to human speech such as ambulance sirens.

The typical workflow of a VAD algorithm is as follows[16]:

- (1) First a noise reduction may be included in VAD.
- (2) Then some speech features are extracted from a segment of the audio signal. The segment is also called frame in the VAD. One frame is often referred to a fixed every short time in audio signal processing such as 20ms.
- (3) Finally a classification rule is applied to classify the segment as non-speech or speech. This classification rule is usually based on whether a specific value exceeds a threshold.

The Web Real-Time Communications (WebRTC) project is a free and open project that aims to provide applications used in browsers and mobiles with RTC capabilities via simple APIs (adopted from <https://webrtc.org/>). It is reported that the VAD developed by Google for the WebRTC project (in the following dissertation, we use webrtcvad for short) is one of the best VADs which is available, fast, and free (adopted from

<https://github.com/wiseman/py-webrtcvad>).

The webrtcvad uses the Gaussian mixture model (GMM) to classify the speech and noise via corresponding probabilities. The formula of GMM is as follows:

$$P(x_k|Z, r_k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_k - u_z)^2}{2\sigma^2}} \quad (1)$$

Where  $x_k$  is the feature vector which is derived from the energy of six frequency bands (80~250 Hz, 250~500 Hz, 500~1000 Hz, 1000~2000 Hz, 2000~3000 Hz, 3000~4000Hz),  $r_k$  is the parameter set of mean  $u_z$  and variance  $\sigma^2$ ,  $Z$  is a bit in which 0 represents noise and 1 represents speech.

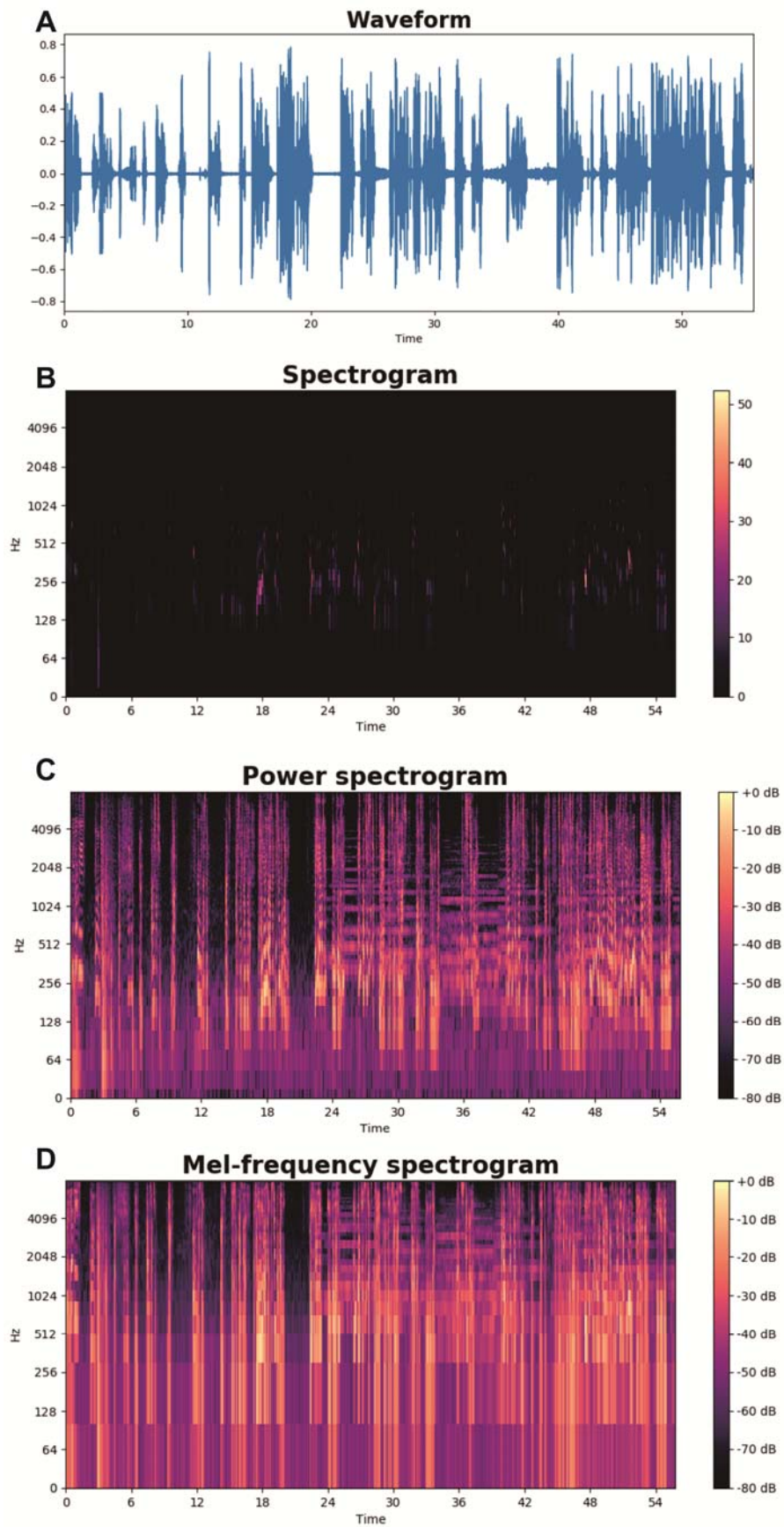
In this dissertation, we adopt the webrtcvad python module (GitHub link: <https://github.com/wiseman/py-webrtcvad>) and implement this module in our speaker recognizer. There are several parameters and implementation tricks. We will introduce them in detail in Section 5.

### 3.2. Audio preprocessing

Although the audio signal is continuous, the audio file is discrete to record the audio signal intensity (aka amplitude) as a function of a series of time points. The number of time points per second stored in the audio file is called sampling rate (also called sample rate). Normally, the sampling rate is 8k Hz, 16k Hz, 32k Hz and 44.1k Hz. For example, if the sampling rate is 16k Hz, it

means that the audio file records 16k time points per second and the corresponding audio signal intensity.

These are several representations of audio signal and different representations focus on different aspects and properties of audio signal. We will introduce them one by one in the next following parts.



**Figure 3 Different representations of audio signal**

### **3.2.1. Waveform**

The waveform of an audio signal is the shape of its audio intensity as a function of time. In other words, the x-axis of waveform is time and the y-axis is amplitude. However, another important feature frequency is hidden in the waveform of audio signal.

### **3.2.2. Spectrum**

It is common to apply fast Fourier transform algorithm (FFT) to transform the waveform to spectrum which the x-axis is frequency and y-axis is amplitude. This operation will obtain more useful information from the audio signal. However, time information is missing in the spectrum.

### **3.2.3. Spectrogram**

In order to take advantage of time property, due to steady property of short time, it is common to transform the waveform of audio to spectrum frame by frame (frame is a very short time, such as 25ms) and combine them together to get spectrogram. In spectrogram, x-axis is time or frame, y-axis is frequency and the value of this point is amplitude. The technique to transform the waveform to spectrum is also called short-time fast Fourier transform (STFT).

### 3.2.4. Mel-spectrogram

The unit of frequency is usually Hz. However for human hearing, the difference between 500 and 1000 Hz is obvious, whereas the difference between 7500 and 8000 Hz is barely noticeable. As a result, Mel scale is constructed such that sounds of equal distance from each other on the Mel Scale, also “sound” to humans as they are equal in distance from one another. It is common to use Mel scale filterbanks to transform spectrogram to Mel-spectrogram.

Mel scale  $f_{mel}$  vs. Hz scale  $f$ :

$$f_{mel} = 2595 \times \log_{10} \left( 1 + \frac{f}{700Hz} \right) \quad (2)$$

Or

$$f_{mel} = 1125 \times \ln \left( 1 + \frac{f}{700Hz} \right) \quad (3)$$

### 3.3. Net “*Vector of Locally Aggregated Descriptors*” embedding extraction

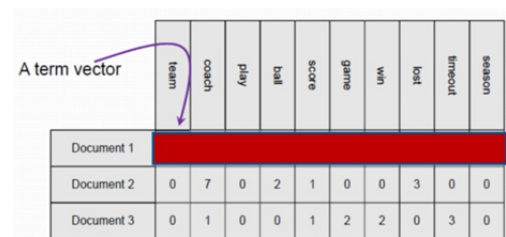
There are lots of methods to generate compact representation for each segment such as MFCCs, speaker factors and i-vectors. With the rapid development of deep neural network, d-vectors are also proposed[1]. In this dissertation, we choose NetVLAD embedding extraction technique which is proposed in 2019 [14]. The idea of aggregated local descriptors is proposed in 2010[17]. After that, NetVLAD[18] and GhostVLAD[19] are developed for image recognition. Finally in 2019, NetVLAD is applied to the audio

recognition field[14].

It is the state of the art performance by a significant margin on the VoxCeleb1 test set. More importantly, fewer parameters than previous methods give NetVLAD a huge advantage for efficient training.

### 3.3.1. Bag of figure

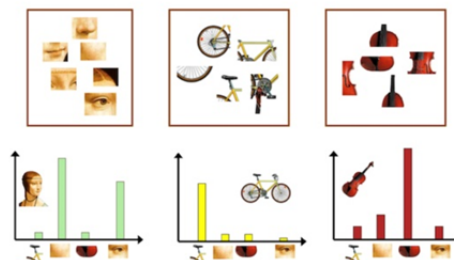
#### Bag of words



A term vector

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1										
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

#### Bag of figure



**Figure 4: Key idea of NetVLAD, this figure is adopted from**

**<http://jermmy.xyz/2017/04/28/2017-4-28-understand-bag-of-feature/> and**

**COMP7103A Data mining course powerpoint.**

When people analyze document, it is common to use a term vector which is also called word embedding concept to represent documents. Firstly put all the documents together to get the several commonly occurring words and



then use these words to construct a list called codebook. Finally, represent each document with these words. The most well-known model is called TF-IDF model:

$$TF(t) \times IDF(t) \quad (4)$$

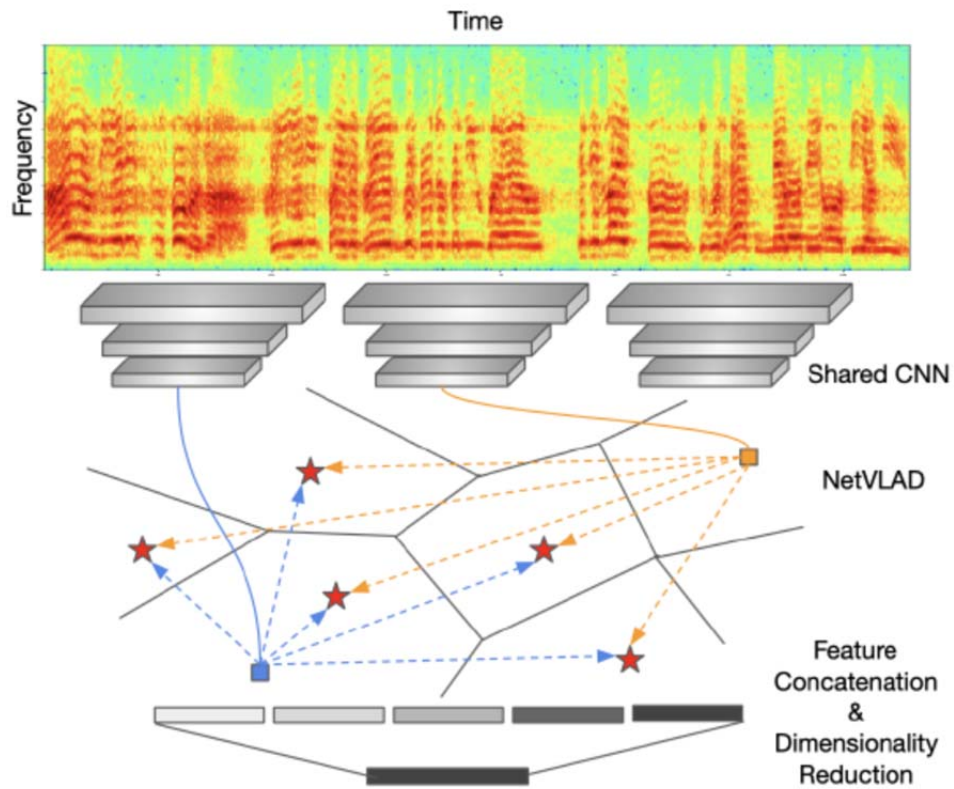
Where  $TF(t)$  is the term frequency of word  $t$ ,  $IDF(t)$  is inverse document frequency of  $t$ , the formula of  $IDF(t)$ :

$$IDF(t) = \log \frac{|D|}{|D_t|} \quad (5)$$

Where  $D_t$  is the number of documents containing  $t$

In similar to the concept of bag of words, the bag of feature is to decompose each image into several components (similar to decompose a document into words). Then find several key components to construct a codebook. Finally represent each figure using these key components to get embeddings.

### 3.3.2. NetVLAD network architecture



**Figure 5: NetVLAD Network architecture. This figure is adopted from [14]**

The NetVLAD embedding extraction system (Figure 5) contains two parts: feature extraction, which a shared CNN (34 layers ResNet) is used to encode the spectrogram and extract frame-level features, and NetVLAD layer which represents each feature with the center of each cluster.

Module	Input Spectrogram ( $257 \times T \times 1$ )	Output Size
<i>Thin ResNet</i>	conv2d, $7 \times 7, 64$	$257 \times T \times 64$
	max pool, $2 \times 2$ , stride (2, 2)	$128 \times T/2 \times 64$
	conv, $1 \times 1, 48$ conv, $3 \times 3, 48$ conv, $1 \times 1, 96$ $\times 2$	$128 \times T/2 \times 96$
	conv, $1 \times 1, 96$ conv, $3 \times 3, 96$ conv, $1 \times 1, 128$ $\times 3$	$64 \times T/4 \times 128$
	conv, $1 \times 1, 128$ conv, $3 \times 3, 128$ conv, $1 \times 1, 256$ $\times 3$	$32 \times T/8 \times 256$
	conv, $1 \times 1, 256$ conv, $3 \times 3, 256$ conv, $1 \times 1, 512$ $\times 3$	$16 \times T/16 \times 512$
	max pool, $3 \times 1$ , stride (2, 2)	$7 \times T/32 \times 512$
	conv2d, $7 \times 1, 512$	$1 \times T/32 \times 512$

**Table 1: The thin-ResNet used for frame level feature extraction. This table is adopted from [14]**

The detailed architecture of thin-ResNet is illustrated in Table 1. The thin ResNet layer maps the input spectrogram ( $R^{257 \times T \times 1}$ ) to frame-level matrix ( $R^{1 \times T/32 \times 512}$ ).

Given the frame-level matrix  $\frac{T}{32}$  512-dimensional local image descriptors  $\{x_t\}$  as input, and  $K$  cluster centers  $\{c_k\}$  as VLAD parameters, the VLAD layers output  $K \times D$  matrix  $V$  ( $K$  refers to the number of chosen cluster,  $D$  refers to the dimensionality of each cluster) using following equation:

$$V(k, j) = \sum_{t=1}^{T/32} \frac{e^{w_k x_t + b_k}}{\sum_{k'=1}^K e^{w_{k'} x_t + b_{k'}}} (x_t(j) - c_k(j)) \quad (6)$$

Where  $\{w_k\}$ ,  $\{b_k\}$  and  $\{c_k\}$  are trainable parameters,  $k \in \{1, 2, \dots, K\}$

After feature concatenation and dimensionality reduction, the network maps the input spectrogram into final audio embedding  $1 \times 512$  vector.

### **3.3.3. Datasets**

Weidi Xie et al. group trained their model end-to-end on the VoxCeleb2 dataset containing over 1 million utterances for 6,112 identities and test on the VoxCeleb1 test sets which is completely disjoint from the VoxCeleb2 dataset (no speakers in common).

Due to time limitation, in this dissertation, we use pre-trained model to build our speaker recognizer without further training.

GitHub link: <https://github.com/WeidiXie/VGG-Speaker-Recognition>

### **3.3.4. Equal error rate**

In audio and image recognition, it is common to use the equal error rate (EER) to evaluate the recognition result. Before introduce equal error rate (EER), we will explain other two related concepts: false acceptance rate (FAR) and false rejection rate (FRR).

	Predicted Class		
Actual Class		0	1
	0	TN	FP
	1	FN	TP

**Table 2: Confusion matrix, TN: true negative, FP: false positive, FN: false negative, TP: true positive**

The definitions of FAR and FRR are based on the confusion matrix (Table 2) as follow:

$$FAR = \frac{FP}{FP + TN} \quad (7)$$

$$FRR = \frac{FN}{FN + TP} \quad (8)$$

Then we use an arithmetic sequence between 0 and 1 as the threshold of the confusion matrix. With the increase of threshold, FRR will increase while FAR will decrease. The crossover point of FRR and FAR is EER.

From this definition, we can deduce that if threshold is 1, then all the prediction results are 1, FP and TP are 0, hence  $FAR = FRR = 0$ . If threshold is 0, all the prediction results are 0, TN and FN are 0, hence  $FAR = FRR = 1$ .

The EER of NetVLAD embedding for verification on the VoxCeleb1 test set is 3.22% which is state of the art performance compared to previous methods.

### 3.4. Links clustering algorithm

Links clustering algorithm is an online clustering algorithm which means the speaker label is immediately emitted once the segment (or segment embedding) is available without seeing future segments.

In this dissertation, we choose the Links: a high dimensional online clustering algorithm[15] as our online clustering algorithm. After we understand the algorithm, we reproduce the whole clustering algorithm with python code and tune the parameter to best represent our own data source.

The Links algorithm is to estimate the probability distribution of each cluster based on its current constituent vectors and take advantage of those estimates to assign new vectors to clusters and update the estimated distributions with each added vector.

#### 3.4.1. High-dimensional approximation

Given the  $N$ -dimensional embedding vectors ( $N \geq 128$ ), the following lemmas are true:

**Lemma 1:** Two randomly chosen vectors  $x, x'$  are almost always almost perpendicular, i.e.,

$$P(x \cdot x' > \delta) < \epsilon \quad (9)$$

for some positive numbers  $\delta \ll 1$  and  $\epsilon \ll 1$ .

**Lemma 2:** The angle  $\theta$  between a cluster center and a random vector from that cluster is almost always almost equal to a global constant  $\theta_c$ , i.e.,

$$P(|\theta - \theta_c| > \delta) < \epsilon$$

for some positive numbers  $\delta \ll \pi$  and  $\epsilon \ll 1$ .

**Lemma 3:** Given two randomly vectors chosen from a cluster with centroid  $\mu$ , their components perpendicular to  $\mu$  will almost always be almost perpendicular to each other, i.e.,

$$P(((x - (x \cdot \mu)\mu) \cdot (x' - (x' \cdot \mu)\mu)) > \delta) < \epsilon$$

for some positive numbers  $\delta \ll 1$  and  $\epsilon \ll 1$ .

### 3.4.2. Estimated distribution

Given a new vector  $x$  and an existing cluster known to include the  $k$  vectors  $\{x_i\}_{i=1}^k$  with centroid  $\mu$ , the author determined a threshold  $x \cdot \mu \geq s(k)$  on the cosine similarity between the new vector and the centroid  $\mu$ . Using the estimation of the cluster's probability distribution and the approximation in lemmas 2 and 3, and assuming  $N \gg k$ , This yields

$$\mu = \frac{1}{\sqrt{k^2 \cos^2 \theta_c + k \sin^2 \theta_c}} \sum_{i=1}^k x_i = \frac{1}{\sqrt{k^2 T_c^2 + k(1 - T_c^2)}} \sum_{i=1}^k x_i$$

and a threshold of

$$s(k) = \frac{T_c^2}{\sqrt{\frac{1}{k} + (1 - \frac{1}{k})T_c^2}}$$

where  $T_c = \cos \theta_c$ , which is called *cluster similarity threshold*

Similarly, to access whether two clusters are the same, the author determine a

threshold on the cosine similarity between their centroids  $\mu_c \cdot \mu'_c \geq s(k, k')$

where  $N \gg k$  and  $N \gg k'$

$$s(k, k') = \frac{1}{\sqrt{(1 + \frac{1}{k} (\frac{1}{T_c^2} - 1))(1 + \frac{1}{k'} (\frac{1}{T_c^2} - 1))}}$$

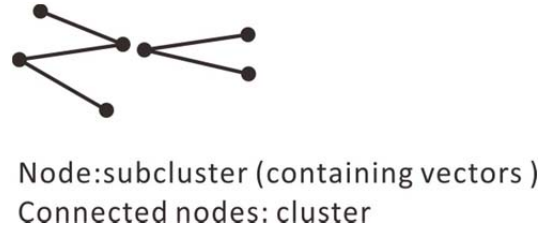
### 3.4.3. Internal representation

The Links algorithm's internal representation is a two-level hierarchy:

subclusters are collections of input vectors and clusters are collections of

subclusters.





**Figure 6: Internal representation of Links clustering**

Links algorithm considers vectors (*i.e.* audio embeddings) which have high cosine similarity as a subcluster which is represented by a node in graph. The edges join “nearly nodes” which means subclusters likely belong to the same cluster given the data so far. Each cluster corresponds to each speaker in the speaker recognition (Figure 6).

Whereas subclusters are indivisible, clusters can become split along graph edges responding to changes in subcluster estimated probability distributions as new data is added.

The advantages of the two-level hierarchy architecture are efficient and practical. It is efficient because the algorithm scales with the number of subclusters and edges rather than the number of vectors. It is practical because the cluster structure which is directly related to speaker labels is the set of potential split points.

#### **3.4.4. Links algorithm**

**Input:** N-dimensional vectors ( $N \geq 128$ )

**Output:** online clustering these vectors

**Key parameters:**

$T_s$ : the subcluster similarity threshold

$T_p$ : the pair similarity maximum

$T_c$ : the cluster similarity threshold

**Two thresholds:**

$$s(k) = \frac{T_c^2}{\sqrt{\frac{1}{k} + (1 - \frac{1}{k})T_c^2}}$$
$$s(k, k') = \frac{1}{\sqrt{(1 + \frac{1}{k} (\frac{1}{T_c^2} - 1))(1 + \frac{1}{k'} (\frac{1}{T_c^2} - 1))}}$$

where  $k$  is the number of vectors in the subcluster

**Refined thresholds:**

$$s'(k, k') = T_c^2 + \frac{T_p - T_c^2}{1 - T_c^2} (s(k, k') - T_c^2)$$

$$s'(k) = s'(k, 1)$$

---

**Add new vector:**

1: A new vector  $x$ , subclusters  $\mu_1, \mu_2, \mu_3 \dots$

2:  $J = \operatorname{argmax}_j \{x \cdot \mu_j\}$

3: **if**  $x \cdot \mu_J \geq T_s$

- 4: add  $x$  to subcluster  $J$
- 5: **else**
- 6: start a new subcluster containing just  $x$
- 7: **if**  $x \cdot \mu_J \geq s'(k_J)$
- 8:     add an edge joining the new subcluster to subcluster  $J$

---



---

#### **Merge clusters:**

- 1: **when**  $x$  is added to an existing subcluster  $J$
- 2:     update  $\mu_J$
- 3: **for** each edge  $(\mu_i, \mu_J)$
- 4:     **if**  $\mu_i \cdot \mu_J \geq T_s$
- 5:         merge subcluster  $\mu_i, \mu_J$
- 6: recursively merge

---



---

#### **Check edges:**

- 1: **if**  $\mu_i \cdot \mu_J \geq s'(k_i, k_J)$
- 2:     edge between subcluster  $i, J$  holds
- 3: **else**
- 4:     edge between subcluster  $i, J$  is removed
- 5:     **for** each edge  $(\mu_i, \mu_m)$

```

6:         if  $\mu_m \cdot \mu_J \geq s'(k_m, k_J)$ 
7:             add an edge joining the subcluster  $m$  to subcluster  $J$ 
8:         else
9:             the cluster remains permanently split

```

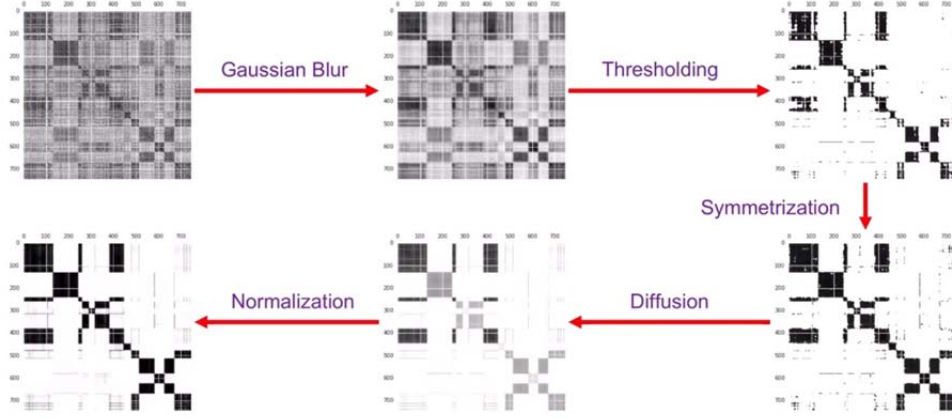
---

### 3.5. Spectral clustering algorithm

Spectral clustering algorithm is an offline clustering algorithm which means the speaker labels are produced after all the segments are available.

Key idea of spectral clustering is to use of the spectrum of the affinity matrix to perform dimensionality reduction and then run K-means on projected embeddings. The spectral clustering consists of 4 parts:

- (1) Construct the affinity matrix  $A$  based on the audio embedding vectors. In the affinity matrix when  $i \neq j$ ,  $A_{ij}$  is the cosine similarity between embedding segment  $i$  and  $j$  and the diagonal elements are set to the maximal value in each row. i.e.  $A_{ii} = \max_{j \neq i} A_{ij}$
- (2) Apply a sequence of refinement operations on the affinity matrix (Figure 7)



**Figure 7: Refinement operations on the affinity matrix. This figure is adopted from Youtube video “Google's Diarization System: Speaker Diarization with LSTM” and modified by ourselves.**

The function of each refinement operation is:

Gaussian blur  $\sigma$ : Smooth the data and reduce the effect of outliers

Row-wise thresholding  $p$ -percentile: Zero-out affinities between different speakers

Symmetrization: Restore matrix symmetry

Diffusion: Sharpen affinity section boundaries of distinct speakers

Row-wise max normalization: Avoid undesirable scale effects.

- (3) After applying all the refinement operations to the affinity matrix, perform eigen-decomposition on the affinity matrix. Let the  $n$  eigen-values be:  $\lambda_1 > \lambda_2 > \dots > \lambda_n$ . The authors use the maximal eigen-gap to determine the number of clusters  $k$

$$k = \arg \max_{1 \leq i \leq n} \frac{\lambda_i}{\lambda_{i+1}} \quad (10)$$

(4) Let the eigen vectors corresponding to the largest  $k$  eigen values be

$v_1, v_2, \dots, v_k$ . The authors replace the  $i$ th embedding by the corresponding eigen-vectors, i.e.

$$x_i = [v_{1i}, v_{2i}, \dots, v_{ki}] \quad (11)$$

and then use the K-means algorithm to cluster these new embeddings and output the speaker labels.

In this dissertation, we apply the spectral clustering to our system from the following Github link.

GitHub link: <https://github.com/wq2012/SpectralCluster>

## **4. Model of investigation**

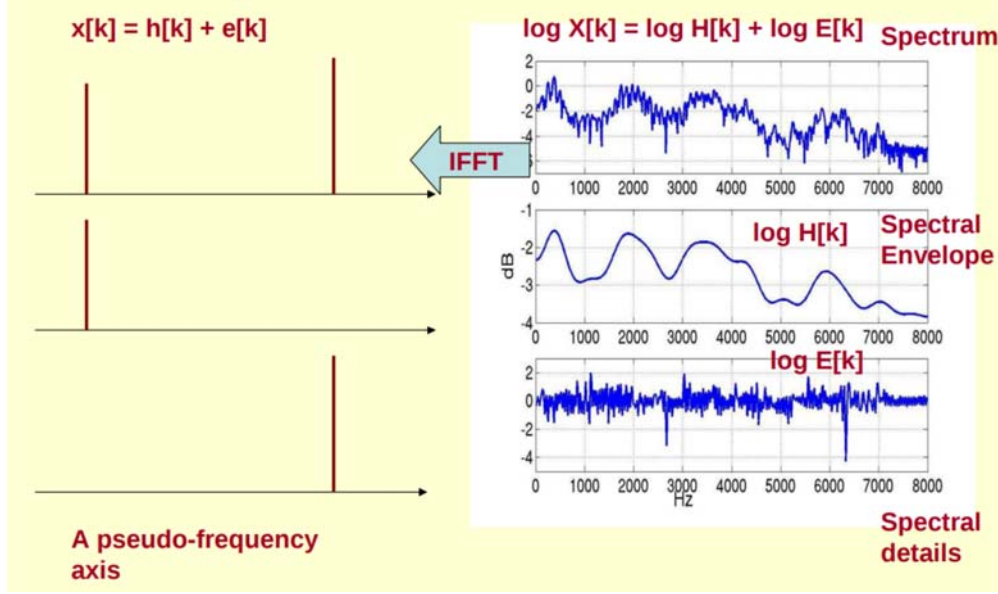
In this section, we review several popular algorithms related to speaker recognizer by introducing another two embedding extraction techniques and one typical offline clustering algorithm. In section 7, we will compare them with our algorithm and explain why we choose NetVLAD embedding extraction and spectral clustering algorithm.

### **4.1. Mel-frequency cepstral coefficients**

In section 3.2.4, we have introduced the Mel-spectrogram. After performing cepstral analysis, we could get the Mel-frequency cepstrum (MFC).

Mel-frequency cepstral coefficients (MFCC) are the coefficients which are part of MFC. MFCC is a traditional embedding extraction technique which is widely used in speaker recognition. In the following section, we will first introduce what the cepstral analysis is and then we will explain the procedure to obtain MFCC.

#### **4.1.1. Cepstral analysis**



**Figure 8: Cepstral analysis on spectrum to obtain spectral envelope and spectral details. This figure is adopted from the following link:**

**<https://www.cnblogs.com/liaohuiqiang/p/10159429.html>**

The spectrum of an audio signal is shown in the first panel of Figure 8. Every peak is the main component of the frequency. These peaks are also called the formants which contain the features of each person. The spectral envelope is the curve which connects these formants smoothly (Figure 8, Panel 2).

Therefore, the spectral envelope is commonly used to represent the features of each person in tradition way. In summary, the spectrum consists of spectral envelope and spectral details (Figure 8, Panel 3) as follows:

$$\log X[k] = \log H[k] + \log E[k] \quad (12)$$

where  $\log X[k]$  represents spectrum,  $\log H[k]$  represents spectral envelope and  $\log E[k]$  represents spectral details.



In conclusion, we need to extract spectral envelop from spectrum to represent the features of audio signal. The way to extract spectral envelope is called the cepstral analysis (Here we just use spectrum for an example to illustrate the process of cepstral analysis, actually it is usually to use mel-spectrogram to do the cepstral analysis).

The key procedure of cepstral analysis is inverse fast Fourier transform (IFFT). In the pseudo-frequency axis, the result of IFFT on spectral envelope  $\log H[k]$  is denoted as  $h[k]$  corresponding to the peak in the low frequency region. On the contrary, the result of IFFT on spectral details  $\log E[k]$  is denoted as  $e[k]$  corresponding to the peak in the high frequency region. Finally, the result of IFFT on spectrum  $\log X[k]$  is denoted as  $x[k]$  which is also called cepstrum. The relationship of  $x[k]$ ,  $h[k]$  and  $e[k]$  is as follows:

$$x[k] = h[k] + e[k] \quad (13)$$

Therefore, in order to obtain the spectral envelope  $h[k]$ , we just need to extract the low frequency region from cepstrum  $x[k]$ .

#### 4.1.2. MFCC

As we mentioned in section 4.1, MFCCs are the coefficients corresponding to

the low frequency region in MFC. The whole procedure to obtain MFCC is as follows:

- (1) First we obtain the spectrogram via STFT and transform into Mel-spectrogram via Mel-filterbanks.
- (2) Next we change Mel-spectrogram to the logarithm to base 10 of the Mel-spectrogram and denote it as  $\log X[k]$
- (3) Now we perform cepstral analysis on  $\log X[k]$
- (4) The low frequency region coefficients  $h[k]$  obtained for Mel-spectrogram are referred to as MFCC.

In section 7, we will compare MFCC with the deep learning embedding extraction technique by literature review and illustrate why we choose NetVLAD embedding extraction algorithm.

## **4.2. LSTM**

In addition to the NetVLAD embedding extraction via deep neural networks, Google speech team has proposed another embedding extraction algorithm based on LSTM. In 2017, they proposed a new loss function called Generalized End-to-End loss (GE2E loss) [11] and in the following year, they implemented this loss function in LSTM neural networks and built their speaker diarization system.

In this section, we will introduce GE2E loss function and LSTM model architecture. In the section 7, we will simply implement these two embedding extraction algorithm (i.e. GE2E and NetVLAD) and compare the diarization result.

#### 4.2.1. Generalized End-to-End loss

The training process of generalized end-to-end (GE2E) is based on a large number of utterances. These utterances are captured from  $N$  speakers and each speaker has  $M$  utterances. The feature extracted from speaker  $j$  utterance  $i$  is denoted as the feature vector  $x_{ji}$  ( $1 \leq j \leq N, 1 \leq i \leq M$ ). Then they feed the utterance feature vector  $x_{ji}$  into a LSTM neural network and use the last frame last layer as the output of the LSTM which denoted as  $(x_{ji}, w)$ .  $w$  represents all the parameters of the LSTM neural network.

The embedding vector  $e_{ji}$  is obtained by applying L2 normalization to the output of the LSTM  $f(x_{ji}, w)$  as follows:

$$e_{ji} = \frac{f(x_{ji}, w)}{\|f(x_{ji}, w)\|} \quad (14)$$

The centroid of speaker  $j$  represents the voiceprint obtained from  $M$  utterances  $(e_{j1}, \dots, e_{jM})$  and is defined as follows:

$$c_j = \frac{1}{M} \sum_{m=1}^M e_{jm} \quad (15)$$

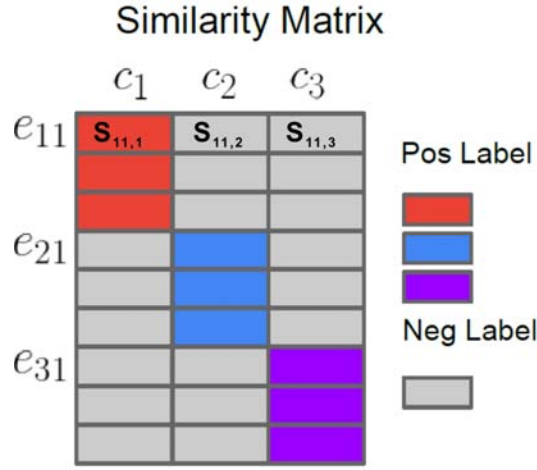
Based on the  $N$  speakers centroids and  $N \times M$  utterances embeddings, the

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

$$S_{ji,k} = \omega \cdot \cos(e_{ji}, c_k) + b \quad (16)$$

where  $\omega$  and  $b$  are training parameters.

The details are illustrated in the Figure 9.



**Figure 9 the details of similarity matrix. This figure is adopted from [11]**

In order to make the  $S_{ji,j}$  equal to 1 if and only if  $k = j$  otherwise make the  $S_{ji,j}$  equal to 0, loss function  $L(e_{ji})$  is defined as follows:

$$L(e_{ji}) = -S_{ji,j} + \log \sum_{k=1}^N \exp(S_{ji,k}) \quad (17)$$

This loss function makes each embedding close to its centroid and away from other centroid.

In addition, removing  $e_{ji}$  when calculating the  $S_{ji,j}$  ( $1 \leq j \leq N, 1 \leq i \leq M$ )

will make the training stable and avoid trivial solutions. Therefore the

Equation 15 and 16 will be updated as follows:

$$c_j^{(-i)} = \frac{1}{M-1} \sum_{\substack{m=1 \\ m \neq i}}^M e_{jm} \quad (18)$$

$$S_{ji,k} = \begin{cases} \omega \cdot \cos(e_{ji}, c_j^{(-i)}) + b & \text{if } k = j \\ \omega \cdot \cos(e_{ji}, c_k) + b & \text{otherwise} \end{cases} \quad (19)$$

Finally, the overall GE2E loss  $L_G$  is the sum of all the losses over the embeddings  $(1 \leq j \leq N, 1 \leq i \leq M)$ .

$$L_G = \sum_{j,i} L(e_{ji}) \quad (20)$$

#### 4.2.2. Model architecture

Based on the GE2E loss, a LSTM neural network model is proposed in order to obtain the embedding of each audio signal [1]. The audio signal is transformed into log mel-spectrogram of dimension 40. Each frame is defined as 25ms and the hop length is 10ms. The log mel-filterbank energies of dimension 40 from each frame as the network input are feed into the LSTM network. The architecture of LSTM neural network is 3-layer LSTM with 768 hidden nodes followed by a projection layer of 256 units. Finally, the embedding is defined based on the last frame last layer of LSTM.

In the section 7, we will implement these two embedding extraction

algorithm (i.e. GE2E and NetVLAD) on our labeled audio data, compare and analyze the diarization result.

The Github link of LSTM is <https://github.com/resemble-ai/Resemblyzer>.

Although it is not the original code published by Google, the voice encoder was implemented based on [11].

### **4.3. K-means**

K-means is the most famous and common offline clustering algorithm. As we mentioned in section 3.5, the last procedure of spectral clustering is also to apply K-means to the dimensional reduced feature vectors. However, it also has several limitations. In this section, we will introduce the algorithm of K-means and limitations. In section 7, we will compare K-means with spectral clustering.

#### **4.3.1. K-means algorithm**

K-means is center based clustering algorithm in order to achieve that a set of objects in a cluster are closer to the “center” of the cluster than to the center of any other cluster. The center of a cluster is often referred to a centroid which means the average of all the points in the cluster or a medoid, the most “representative” point of a cluster. The algorithm of K-means is very simple as follows:

---

### K-means algorithm

- 1: Specify the number of clusters  $K$
  - 2: Randomly select  $K$  points as the initial centroids
  - 3: **Repeat**
  - 4:     Form  $K$  clusters by assigning all points to the closest centroid
  - 5:     Re-compute the centroid of each cluster
  - 6: **Until** the centroids don't change
- 

Because the initial  $K$  randomly chosen centroids will determine the final clustering result, therefore there will be some measures to evaluate the clustering result. The most common measure is *Sum of Squared Error* (SSE).

For each point, the error is the distance to its cluster center (centroid or medoid). The equation of SSE is as follows:

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x) \quad (21)$$

where  $x$  is the point in cluster  $C_i$  and  $m_i$  is the center of the cluster  $C_i$ .

Based on the SSE, we choose the cluster with the smallest SSE from all the clustering results determined by different initial centroids.

#### 4.3.2. Limitations of K-means

The effects of K-means algorithms largely depend on the property of the clusters. The limitations of K-means are from the following three aspects:

- (1) The scales of different clusters are different. More specifically, clusters are of different size or of different densities or non-globular shapes.
- (2) K-means are susceptible to noise and outliers. Because noise and outliers will largely affect the computation of the center of the cluster.
- (3) The definition of distance between points is critical for clustering. When the dimensionality of the each point increases, the data becomes increasingly sparse in the occupied space. In our speaker recognizer, the dimensionality of the feature vector is 512. Therefore we need to do dimensionality reduction before apply K-means to our data. It is exactly what spectral clustering do which we introduced in section 3.5.

In section 7, we will compare two offline clustering algorithms K-means and spectral clustering and illustrate why we choose spectral clustering in our offline speaker recognizer.



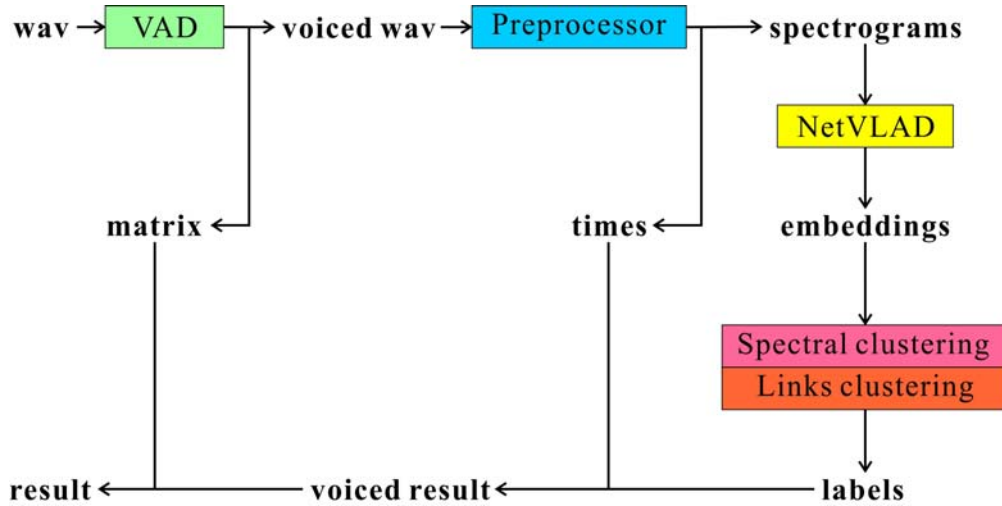
## 5. Design and construction of software system

In this section, we will first give a brief overview about our real-time and offline speaker recognizer including the overall workflow of our system.

Next we will explain each component one by one in very detailed manner including the parameters setting. Finally, we will integrate all the intermediate results into the final speaker diarization result.

### 5.1. Overview

Our system mainly consists of four components: voice activity detector (VAD), preprocessor, NetVLAD embedding extractor and spectral/Links clustering. The overall workflow of our system is shown in Figure 10 and the details are as follow:



**Figure 10: Overall workflow of our online and offline speaker recognizer.**

All the audio data are transformed to the wav format with the sampling rate

16000 Hz, mono channel as the system input. Then the VAD module will classify each frame of the audio data as speech or non-speech and concatenate all the speech frames into **voiced wav data**. In addition, it will produce a **matrix** to record all the non-speech frames timestamp. The details will be illustrated in the section 5.2.

Next, the voiced wav data will be processed by the preprocessor. The main functions of the preprocessor are (1) speech feature analysis which extracts the spectrogram from the audio data via short-time Fourier transform (STFT); (2) speech segmentation which splits the audio file into equal time-length non-overlapping segments. The preprocessor will produce the **spectrograms** and **corresponding timestamps** (times in Figure 10) of each segment one by one. The detailed parameters will be explained in section 5.3.

Based on the spectrogram of each segment, the embedding extractor will output the corresponding embedding via NetVLAD neural networks. The principle of NetVLAD is illustrated in section 3.3 and the detailed parameter setup will be shown in section 5.4.

As we mention in section 3.4 and 3.5, we utilize two different clustering algorithms to cluster the audio segments based on their embeddings: (1) Links online clustering algorithm which means the speaker label is

immediately emitted once the segment (or segment embedding) is available without seeing future segments; (2) Spectral offline clustering algorithm which means the speaker labels are produced after all the segments are available.

After we obtain the segment labels, we integrate these labels with the corresponding timestamp previously produced by preprocessor and produce the diarization result of the voiced audio data. Finally, combining the non-speech matrix produced by VAD, we can acquire the final diarization result.

In addition, our online speaker recognizer just uses the online clustering algorithm instead of the offline one. Although the online speaker recognizer can be easily transformed to be real-time one, due to the performance of the Links online clustering algorithm, it is not real-time in this dissertation.

## **5.2. VAD**

As we mentioned in Section 3.1, the VAD module classifies each frame of the audio data as speech or non-speech via the Gaussian mixture model (GMM).

The aggressiveness mode which is an integer between 0 and 3 defines the different level to filter out non-speech data. 0 is the least aggressive while 3 is the most aggressive. After we compare different aggressiveness mode, mode

3 has the best behavior to classify speech or non-speech data in our system.

In order to increase the accuracy of classification, we choose the percentage of speech frames in a fixed number of consecutive frames instead of single frames classification result as our evaluation criteria. In more detail, we define each frame time length as 20ms and classify each frame as speech or non-speech one by one. In addition, we define two states of VAD:

TRIGGERED and NOTTRIGGERED and start in the NOTTRIGGERED state. Then we calculate the percentage of speech frames in 10 consecutive frames. If this percentage exceeds 90%, we will enter TRIGGERED state. Until the percentage is below 10%, we will enter NOTTRIGGERED state. The frames in the TRIGGERED state are classified as speech data.

Take one clip of audio data DHS01E04\_15 for example: the intermediate result is as follows:

000001111111111+(9.24)1111111111111111110000000000-(10.06)000

0 means non-speech frame and 1 is speech frame. We use a deque for our sliding window which contains 10 consecutive frames. If the percentage of speech frames in sliding window exceeds 90%, we enter TRIGGERED state. The following frames including sliding window frames will be classified as speech data. Next, we continue to slide the sliding window and examine the percentage. Until the percentage is below 10%, we enter NOTTRIGGERED

state and the following frames will be classified as non-speech data. In this example, the timestamp of the first frame in TRIGGERED state is 9.24s and the last frame is 10.06s. Therefore, we classify the audio data between 9.24s and 10.06 as speech data.

After we identify the speech data out of the whole audio data, we concatenate all the speech data into the voiced wav data and simultaneously produce a matrix to record all the non-speech frames timestamp.

### 5.3. Preprocessor

Next, the voiced wav data will be further processed by the preprocessor. The main functions of the preprocessor are (1) speech feature analysis (2) speech segmentation.

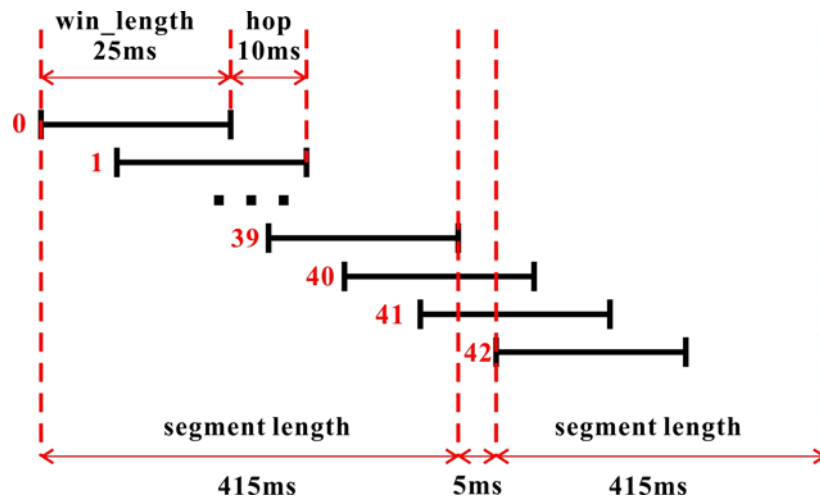


Figure 11: Detailed parameters of preprocessor

### 5.3.1. Speech feature analysis

Speech feature analysis aims to extract the spectrogram from the audio data via short-time Fourier transform (STFT). The sampling rate of all the input audio data is 16000Hz. Window length (win\_length) is 25ms which corresponds to 400 points and the hop length is 10ms which corresponds to 160 points (Figure 11). In each frame, we do STFT on 512 points. If the number of points in one frame is lower than 512, we will do zero-padding. Therefore, we obtain  $\frac{N}{2} + 1 = 257$  values in each frame via STFT.

### 5.3.2. Speech segmentation

The purpose of speech segmentation is to split the audio file into equal time-length non-overlapping segments. As illustrated in Figure 11, the first segment contains 40 frames (frame 0-39) corresponding to 415ms which is also the resolution of speaker recognizer. In order to make sure the segments are non-overlapping, we skip the following two frames (frame 40, 41) and start the next segment from frame 42. The duration between two segments is 5ms which makes very subtle effect on our speaker recognizer system.

In summary, the preprocessor will produce the spectrograms of each segment into a  $257 \times 40$  matrix and we integrate all the segments spectrograms into one 3D array. In addition, preprocessor will also produce the corresponding timestamps of each segment into a matrix called times in Figure 10.

## 5.4. NetVLAD

As we mentioned in Section 5.1, the NetVLAD will produce the corresponding embeddings based on the spectrogram of each segment. Weidi Xie’s group[14] experimented with different cluster number and a recently proposed layer called GhostVLAD. In this dissertation, we choose the optimal parameters setting from their research.

### 5.4.1. GhostVLAD

In the GhostVLAD layer[19], some of the clusters are excluded in the final concatenation and dimensionality reduction. In other words, it means they do not contribute to the final representation. Hence, these clusters are referred to as ‘ghost clusters’.

### 5.4.2. Detailed parameters setting

In this dissertation, we choose the optimal parameters setting from Weidi Xie’s group research[14].

Parameter	Value
VLAD cluster number	8
Ghost cluster number	2
Loss function	Softmax

**Table 3: NetVLAD parameters setting**

### 5.5. Spectral clustering

In this dissertation, we choose spectral clustering as our offline clustering algorithm. As we mentioned in Section 3.5, there is a sequence of refinement operations on the affinity matrix. Two main parameters could have an effect on the refinement result and the details are as follow:

Gaussian blur  $\sigma$ : Smooth the data and reduce the effect of outliers

Row-wise thresholding p-percentile: Zero-out affinities between different speakers.

In this dissertation, we choose the default values of these two parameters:

Parameter	Value
Gaussian blur $\sigma$	1
Row-wise thresholding p-percentile	0.95

**Table 4: Spectral clustering parameters setting**

### 5.6. Links clustering

As we mentioned in Section 3.4, there are three important thresholds to determine the accuracy of the Links online clustering algorithm. More specifically, the similarity thresholds are listed as follow:

$T_s$ : the subcluster similarity threshold

$T_p$ : the pair similarity maximum



$T_c$ : the cluster similarity threshold

So far we have 16 labeled video clips. In order to find the optimal thresholds which can represent our audio data, we use 70% of labeled video clips (11 clips) as training data to find out the optimal thresholds and use the rest 30% (5 clips) as test data.

The strategy of finding the optimal thresholds is:

(1) We search all the combinations of  $T_s$ ,  $T_p$  and  $T_c$  between 0.5 and 1.

The step is 0.1.

(2) We calculate the DER of each video clip in the training set under these thresholds.

(3) We calculate the average DER of the training set.

(4) The thresholds which result in the minimal DER of the training set are the optimal thresholds we should find.

Based on the strategy, we find the optimal thresholds which can result in the minimal DER of the training set as follow:

Parameter	Value
Subcluster similarity threshold $T_s$	0.7
Pair similarity maximum $T_p$	0.9
Cluster similarity threshold $T_c$	0.6

**Table 5: Links clustering parameters setting**

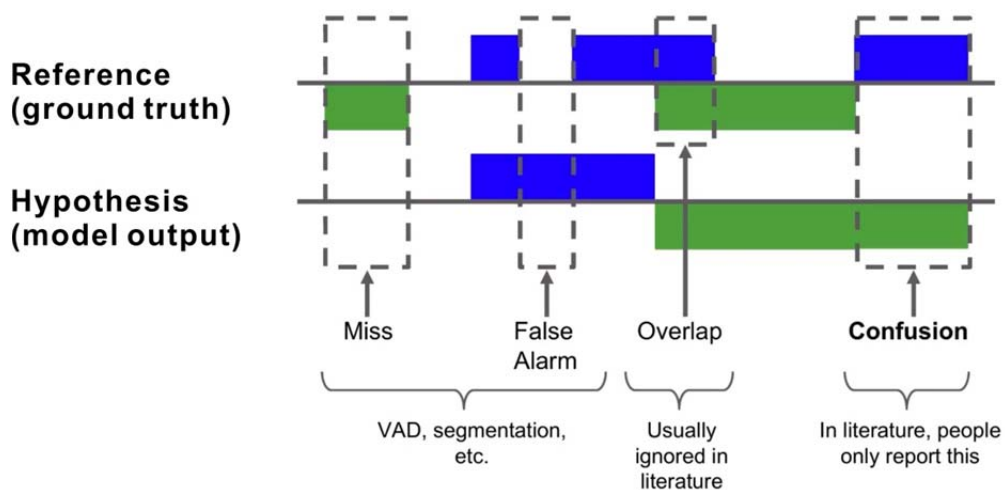
The minimal DER of the training set is 0.414.

## 6. Experimental result

In this section, we first illustrate the evaluation metrics which aims to evaluate our speaker recognizer system. Then we analyze the effect of VAD on our recognizer system. Finally, we summarize the evaluation results of our offline and online system.

### 6.1. Evaluation metrics

Diarization error rate (DER) is the most common and standard metric to evaluate the speaker recognizer system. DER consists of four components: missed detection, false alarm, overlap and confusion (Figure 12). Missed detection and false alarm are related with voice activity detector, while overlap is usually ignored in literature. Confusion part is the most important in DER which represents whether the label is right or not.



**Figure 12: The components of Diarization Error Rate (DER). This figure**

**is adopted from Youtube video “Google’s Diarization System: Speaker**

### **Diarization with LSTM” and modified by ourselves.**

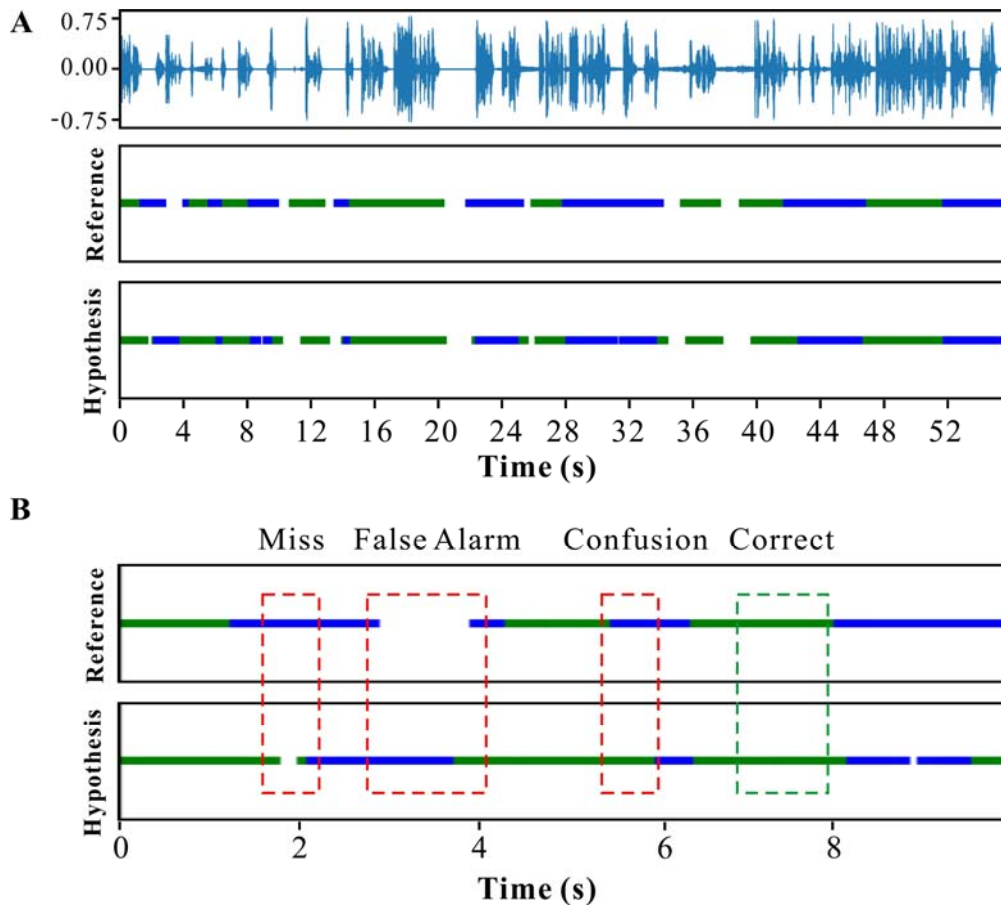
In this dissertation, we use pyannote.metrics python module to evaluate our system [20]. pyannote.metrics is an open-source Python library which contains a large set of evaluation metrics which are available for diagnostic purposes of all modules of typical speaker diarization including speech activity detection, speaker change detection and clustering.

In pyannote.metrics python module, DER is defined as follows:

$$\text{DER} = \frac{\text{missed detection} + \text{false alarm} + \text{confusion}}{\text{total}} \quad (22)$$

where missed detection is the time length of speech incorrectly classified as non-speech, false alarm is opposite which means the time length of non-speech incorrectly classified as speech, confusion is the time length of incorrect labeled segment and total is the total time length in the reference.

So far we manually labeled 16 video clips as our references. Then we use our speaker recognizer system to recognize the speaker in each clip and summarize all the information into a single figure as follows (Panel A in Figure 13). We take DHS01E04\_15 clip for example:



**Figure 13: DHS01E04\_15 clip diarization result**

The Panel A of Figure 13 consists of three parts: the upper part is the waveform of the video clip, the middle part is diarization result of the video clip which is labeled by ourselves manually as reference and the lower part is the predicted diarization result produced by our speaker recognizer system as hypothesis. In both reference and hypothesis, the lines with the same color correspond to the same speaker in each clip.

In the panel B, we take a small clip from the video between 0-10 second to illustrate the three main components of the DER (Miss, False Alarm and

Confusion) and summarize the final result into the following table.

Audio ID	Miss (second)	False alarm (second)	Confusion (second)	Total (second)	DER
DHS01E04_15	8.15	2.84	2.4	46.4	0.29

**Table 6: DHS01E04\_15 clip diarization result**

The total time of the video clip is 46.4 seconds. Missed detection, False alarm and Confusion time are 8.15 seconds, 2.84 seconds and 2.4 seconds respectively. Therefore the DER is  $(8.15+2.84+2.4)/46.4 = 29\%$ .

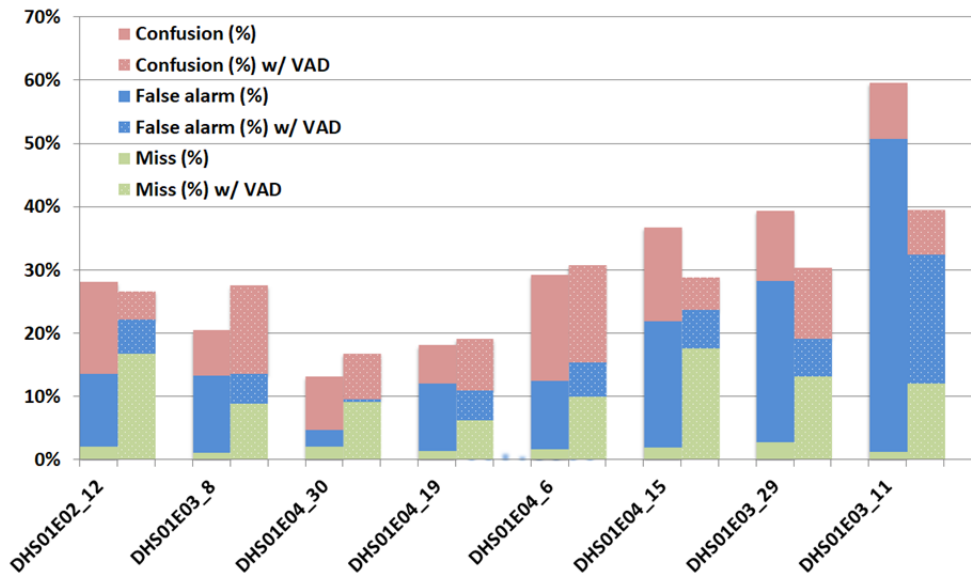
## **6.2. VAD impact on DER**

In this section, we analyze the effect of VAD on DER and illustrate the importance of VAD.

As we mentioned in Section 5.2, the VAD module classifies each frame of the audio data as speech or non-speech and in Section 6.1, we illustrate the DER consists of three components: Missed detection, False Alarm and Confusion. Based on the definition, the Missed detection and False Alarm are closely related to the VAD components.

In order to further illustrate the effect of VAD on DER, we use 8 video clips diarization results to analyze this problem. As shown in Figure 14, each video

clip contains two columns corresponding to the diarization result without (Left column) or with VAD (Right column). We observe that if DER without VAD is less than 30% (first five clips), VAD has limited impact on DER, however, if DER without VAD is more than 30% (last three clips), the VAD can significantly decrease the DER and achieve the better result.



**Figure 14: VAD impact on DER**

Therefore, we conclude the VAD is essential for the speaker recognize system and integrate the VAD in our system.

### 6.3. Offline speaker diarization result

As we mention in Section 6.1, we use pyannote.metrics python module to evaluate our system. We manually labeled 16 video clips as reference and compare the predicted output by our speaker recognizer system as hypothesis.

<b>Audio ID</b>	<b>Miss</b>	<b>False alarm</b>	<b>Confusion</b>	<b>Total</b>	<b>DER</b>
	<b>(second)</b>	<b>(second)</b>	<b>(second)</b>	<b>(second)</b>	
<b>DHS01E02_12</b>	7.33	2.33	1.98	43.60	27%
<b>DHS01E03_8</b>	3.19	1.66	5.02	35.80	28%
<b>DHS01E03_11</b>	3.39	5.68	2.01	28.00	40%
<b>DHS01E03_29</b>	3.12	1.42	2.65	23.70	30%
<b>DHS01E04_6</b>	2.10	1.12	3.21	20.90	31%
<b>DHS01E04_15</b>	8.15	2.84	2.40	46.40	29%
<b>DHS01E04_19</b>	1.95	1.44	2.49	30.80	19%
<b>DHS01E04_30</b>	2.43	0.10	1.91	26.40	17%
<b>DHS01E05_4</b>	1.37	1.63	2.76	28.30	20%
<b>DHS01E05_18</b>	3.29	3.03	5.88	38.20	32%
<b>DHS01E05_31</b>	2.83	5.07	3.06	46.50	24%
<b>DHS01E06_4</b>	3.23	5.94	3.88	37.90	34%
<b>DHS01E06_10</b>	2.03	3.00	2.78	39.70	20%
<b>DHS01E06_25</b>	2.53	2.89	4.30	42.80	23%
<b>DHS01E07_14</b>	1.56	4.19	0.95	26.00	26%
<b>DHS01E07_18</b>	2.90	3.00	1.50	33.10	22%

**Table 7: Offline speaker recognizer results**

Table 6 is the detailed result of our offline speaker recognizer and we summarize the results and get the percentage of each component of DER in Table 7.



	Miss/total	False alarm/total	Confusion/total	DER
Average	9.2%	8.3%	8.7%	26%

**Table 8: Percentage of each component of DER with offline speaker recognizer**

In Table 7, we observe that the DER of offline speaker diarization system is 26%. The percentage of Missed detection and False alarm are 9.2% and 8.3%. These two parts contribute to 67% of the final DER. As we mentioned in Section 6.2, these two parts are very closely related to VAD and segmentation. Also most teams [3, 5, 6] exclude the Missed detection and False alarm parts and only report the Confusion part.

Therefore, the DER of our offline speaker recognizer system is 26% and the percentage of the most important part Confusion is only 8.7%. We believe our speaker recognizer is satisfactory and we will discuss the improvements of Missed detection and False Alarm parts in Section 7.

#### **6.4. Online speaker diarization result**

As we mentioned in Section 5.6, we scan all the combinations of the similarity thresholds  $T_c$ ,  $T_s$  and  $T_p$  and find the optimal ones based on the 11 video clips in the training set. The final total DER in the training set is 41.4%.

Next step, we use this Links online clustering algorithm to analyze the next 5 video clips in the test set.

<b>Audio ID</b>	<b>Miss</b>	<b>False alarm</b>	<b>Confusion</b>	<b>Total</b>	<b>DER</b>
	<b>(second)</b>	<b>(second)</b>	<b>(second)</b>	<b>(second)</b>	
<b>DHS01E06_4</b>	3.23	5.94	6.17	37.90	40%
<b>DHS01E06_10</b>	2.03	3.00	4.89	39.70	25%
<b>DHS01E06_25</b>	2.53	2.89	11.05	42.80	38%
<b>DHS01E07_14</b>	1.56	4.19	4.27	26.00	39%
<b>DHS01E07_18</b>	2.90	3.00	10.53	33.10	50%

**Table 9: Online speaker recognizer results**

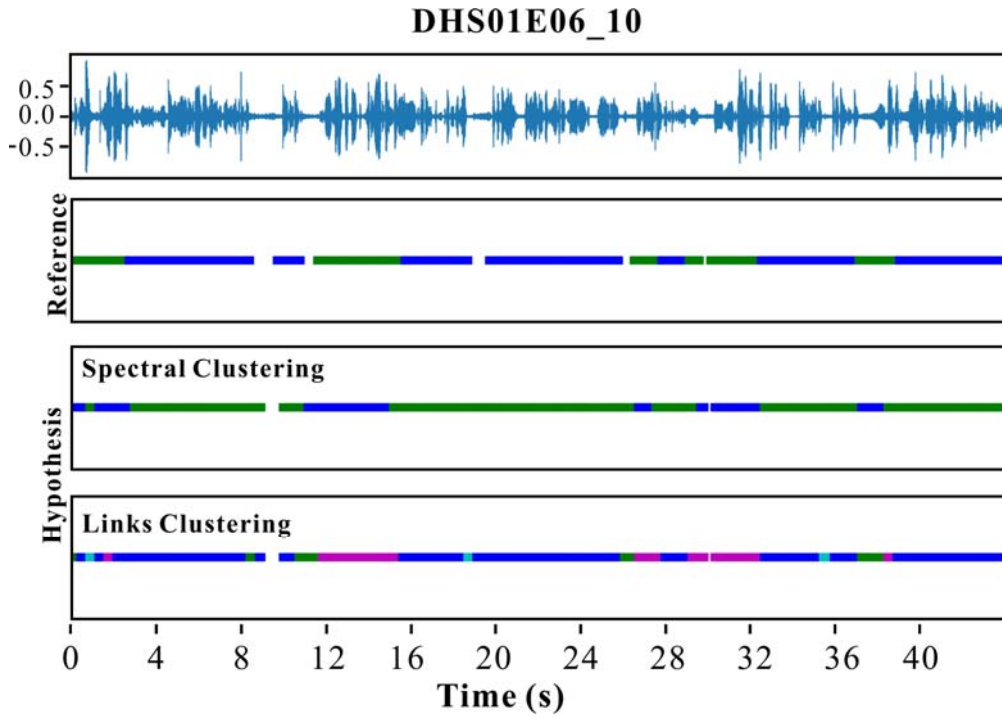
Table 9 is the detailed results of our online speaker recognizer and we summarize the results and get the percentage of each component of DER in Table 10.

	<b>Miss/total</b>	<b>False alarm/total</b>	<b>Confusion/total</b>	<b>DER</b>
<b>Average</b>	6.9%	11.0%	20.5%	38%

**Table 10: Percentage of each component of DER with online speaker recognizer**

Because the VAD of online or offline speaker recognizer is same, the percentage of Miss and False alarm remains the similar value: offline 9.2% +

8.3% = 17.5%, online 6.9% + 11% = 17.9%. We calculate 16 video clips results in Table 8 and 5 video clips in Table 10 which can explains the slightly different in Missed detection part and False Alarm part. The DER of test set is 38% and the percentage of the confusion part is 20.5%. Both of them are higher than the ones of the offline speaker recognizer.



**Figure 15: Example of online speaker recognizer diarization result**

In order to illustrate the difference between online clustering and offline clustering, we take one example DHS01E06\_10 in Figure 15. We observe that Links clustering algorithm could differentiate the speaker roughly but it is more sensitive than the spectral clustering. Hence we discover that Links clustering algorithm produces more clusters than spectral clustering. Based

on the Table 10 and Table 8, we find that the percentage of Confusion part in DER increases from 33% of offline to 54% of online speaker recognizer.

Therefore, we believe that the accuracy of the offline speaker recognizer is better than online speaker recognizer while online speaker recognizer can be transformed into real-time speaker recognizer easily which is a big advantage.

## **7. Discussion and future works**

In this section, we will discuss several questions as follow:

- (1) Why we choose NetVLAD embedding extraction algorithm.
- (2) The performance of offline speaker recognizer for multi-persons speech.
- (3) The difference between K-means and spectral clustering algorithm.
- (4) How to decrease the Missed detection and False Alarm.

Also we proposed the future work which we can continue to improve our speaker recognizer.

### **7.1. Embedding extraction algorithm selection**

As we mentioned in Section 1, in addition to the NetVLAD embedding extraction algorithm, it is reported that specific features such as Mel-frequency cepstral coefficients (MFCCs), speaker factors, or i-vectors are extracted from the audio data. In recent years, more and more neural network based audio embedding extraction techniques are proposed. In 2018, Google speech team proposed a new technique based on LSTM which outperforms the tradition technique i-vectors [1].

The basic principle of LSTM based technique is introduced in Section 4.2. In order to compare LSTM to NetVLAD which we used in this dissertation, we used these two techniques to analyze two video clips. The Github link of

LSTM is <https://github.com/resemble-ai/Resemblyzer>.

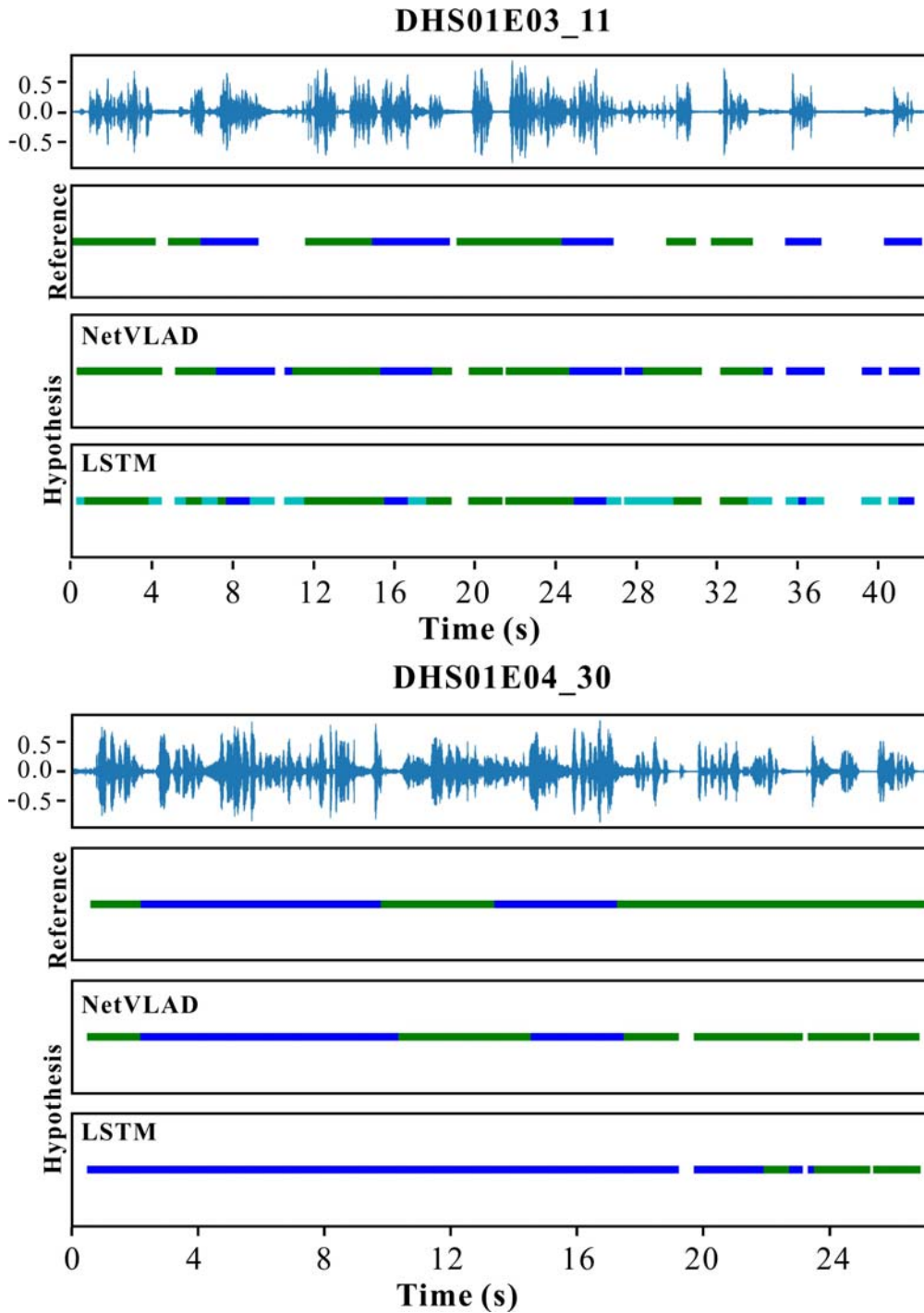


Figure 16: LSTM and NetVLAD comparison

The result is shown in Figure 16, we find that under our system setting,

NetVLAD based embedding extraction technique is better than LSTM based

one. We believe that LSTM based one may be sensitive to the audio file length which means the audio file length in the training set must be comparable with the one in the test set. The audio file length in the training set is 1.6s while the segment length is 415ms which is mentioned in Section 5.3. In contrast, NetVLAD may tolerate this difference which can perform better for speaker recognizer.

## 7.2. Multi-persons speech

After we investigate the speech contains only two persons, we explore what if multiple persons are in speech. Therefore we labeled several video clips and take one example DHS01E02\_17.

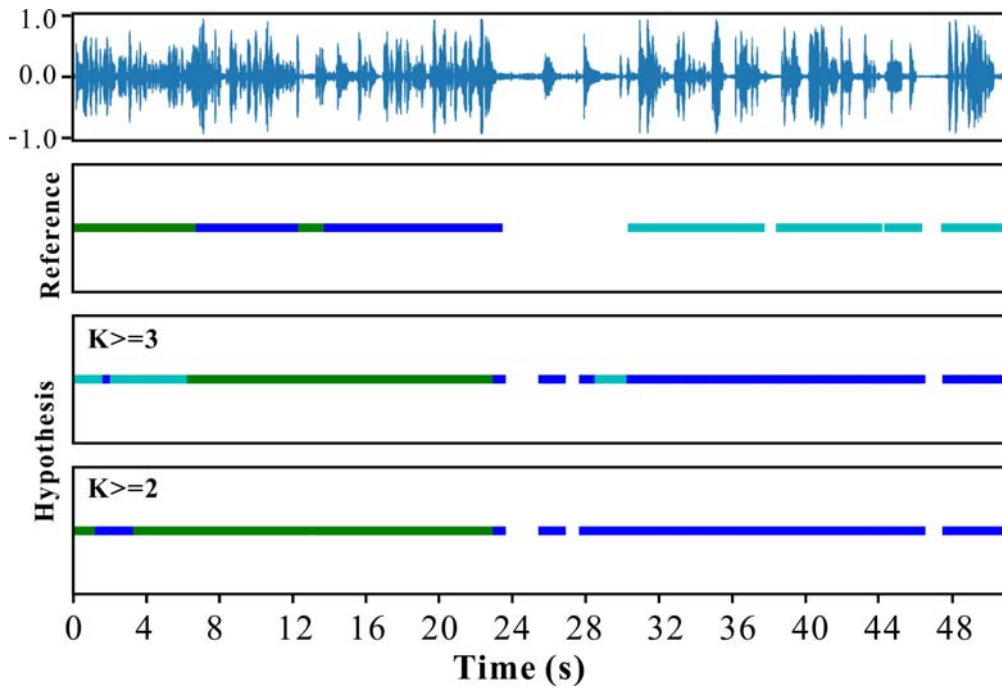


Figure 17: multi-persons speech DHS01E02\_17

If we set the minimal value of  $K$  is 3, the spectral clustering algorithm can differentiate these 3 persons clearly. However if we set the minimal value is 2, the spectral clustering cannot achieve the good result. This belongs to the wrong cluster number determination which is mentioned in Section 3.5. We will further improve this part in the future works.

### **7.3. K-means vs. Spectral clustering.**

There are two famous offline clustering algorithms: K-means and spectral clustering. Due to the following reasons:

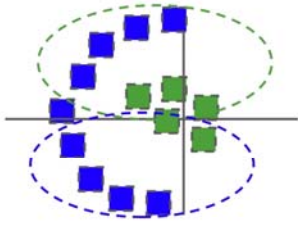
- (1) Speech data are often non-Gaussian.
- (2) One person may speak more often.
- (3) Inter-gender differences large, intra-gender differences small. In Figure 18 Gender/age/race effects, it is Challenging to find  $K$  is 3, K-means prefers 2 clusters.
- (4) Overlapping speech creates connects between clusters.

K-means may not be good at clustering the speech data. However, these problems can be mitigated by spectral clustering (Figure 18). Therefore we choose spectral clustering for offline clustering [1].

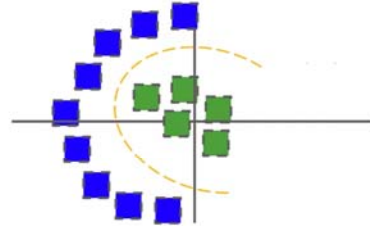


## K-means

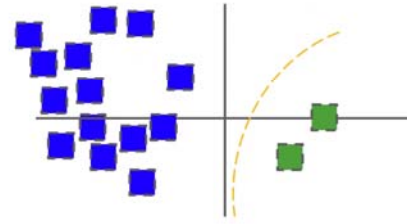
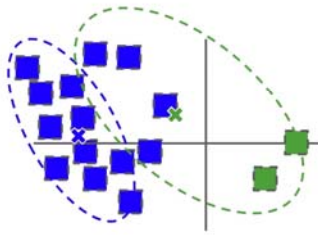
Non-Gaussian data



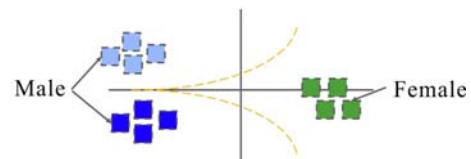
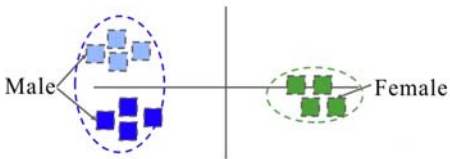
## Spectral clustering



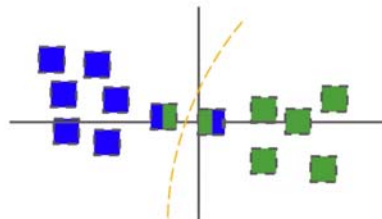
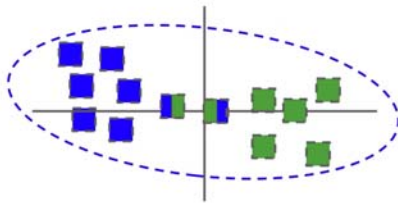
Imbalanced clusters



Gender/Age/Race effects



Overlapping speech



**Figure 18: K-means and spectral clustering compare, this figure is adopted from Youtube video “Google's Diarization System: Speaker Diarization with LSTM” and modified by ourselves.**

### 7.4. Decrease Missed Detection and False Alarm

As we mentioned in Section 6.1, the Missed Detection and False Alarm is

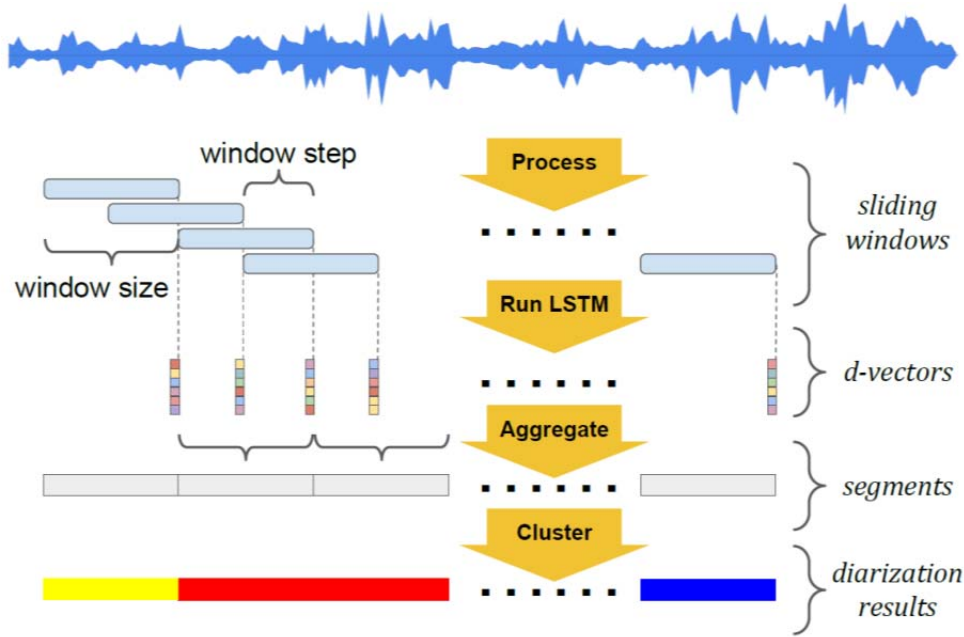
very closely related to the VAD and segmentation. In our offline system, these two parts contribute to 67% of the final DER. We believe we can decrease these two parts from the following aspects:

#### (1) VAD

In our video clips, there are some noise and background music which may affect our VAD. On the other hand, in order to increase the accuracy of VAD, we choose the percentage of speech frames in a fixed number of consecutive frames instead of single frames classification result as our evaluation criteria (Section 5.2). We will continue to tune the parameter such as the number of consecutive frames and optimize the VAD.

#### (2) Segmentation

In Section 5.3, we explain our segmentation strategy. We just choose 40 frames as one segment and split the audio data into non-overlap segment.



**Figure 19: A flowchart of d-vector based diarization system. This figure is adopted from [1]**

In the Figure 19, this is the flowchart of LSTM based speaker diarization system [1]. We notice that they first run the LSTM on sliding windows and get the corresponding embeddings d-vectors. Then they aggregate the d-vectors to form the final embeddings of the segment. We think that it will improve the accuracy of the embeddings which can represent the segment better. However due to lack of enough information, we cannot obtain the detailed method of the aggregation.

### 7.5. Training NetVLAD embedding algorithm by ourselves

So far, we just use pre-trained parameters of NetVLAD to get the embedding of the segments. Their model parameters work very well for speaker

verification. In contrast, the percentage of the confusion part of our offline speaker recognizer is 8.7%. Therefore, we may train the model by ourselves to tune the parameters for the speaker recognizer in future works.

#### **7.6. Increase the labeled audio file number**

Labeling the video clips by ourselves is very time consuming. So far, we only have 16 labeled video clips. With the increase number of the video clips, we can find more accurate similarity thresholds  $T_c, T_s$  and  $T_p$  which can improve the accuracy of our online speaker recognizer. We will label more video clips in future works.

## 8. Conclusions

In this dissertation, we successfully develop our own offline and online speaker recognizer and we summarize the key point of system and corresponding code we adopt as follow:

### (1) WebrtcVAD

We use the VAD developed by Google for the WebRTC project, one of the best VADs which is available, fast, and free to exclude the non-human speech (Section 3.1). In order to increase the accuracy, we choose the percentage of speech frames in a fixed number of consecutive frames instead of single frames classification result as our evaluation criteria (Section 5.2).

Github link: <https://github.com/wiseman/py-webrtcvad>

### (2) Preprocessor

After we get the voiced audio file, we extract the spectrogram from the audio data via short-time Fourier transform (STFT) (Section 5.3.1). And then we split the audio file into non-overlapping segment which contains 40 frames corresponding to 415ms which is also the resolution of speaker recognizer (Section 5.3.2).

### (3) NetVLAD

Based on the spectrogram of each segment, we utilize the NetVLAD

embedding extraction technique to obtain the embeddings (Section 3.3 and Section 5.4).

Github link: <https://github.com/WeidiXie/VGG-Speaker-Recognition>

#### (4) Spectral offline clustering

We use the spectral clustering algorithm as our offline clustering algorithm to cluster the segments according to their embeddings and get the speaker label of each segment (Section 3.5 and Section 5.5)

Github link: <https://github.com/wq2012/SpectralCluster>

#### (5) Links online clustering

Links online clustering is the key component for our online speaker recognizer. We scan all the combinations of the similarity thresholds  $T_c$ ,  $T_s$  and  $T_p$  and find the optimal ones to present our audio data (Section 3.4 and Section 5.6).

#### (6) DER

Finally we use the pyannote.metrics python module to evaluate our system and get the DER of our offline and online speaker recognizer system.

In conclusion, the DER of our offline speaker recognizer system is 26% and

the percentage of the most important part Confusion is only 8.7%. The DER of our online speaker recognizer is 38% and the percentage of the Confusion part is 20.5%

## 9. Reference

- 1 Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Moreno, I.L.: ‘SPEAKER DIARIZATION WITH LSTM’, *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4133-4136
- 2 Patrick Kenny, D.R., and Fabio Castaldo: ‘Diarization of telephone conversations using factor analysis’, *IEEE Journal of Selected Topics in Signal Processing*, 2010, vol. 4, pp. 1059–1070
- 3 Fabio Castaldo, D.C., Emanuele Dalmasso, Pietro Laface, and Claudio Vair: ‘Stream-based speaker segmentation using speaker factors and eigenvoices’, *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 4133–4136
- 4 Gregory Sell, D.G.-R.: ‘Speaker diarization with plda i-vector scoring and unsupervised calibration’, *Spoken Language Technology Workshop (SLT)*, 2014, pp. 413–417
- 5 Stephen H Shum, N.D., R’eda Dehak, and James R Glass: ‘Unsupervised methods for speaker diarization: An integrated and iterative approach’, *IEEE Transactions on Audio, Speech, and Language Processing*, 2013, vol. 21, no. 10, pp. 2015–2028
- 6 Mohammed Senoussaoui, P.K., Themis Stafylakis, and Pierre Dumouchel: ‘A study of the cosine distance based mean shift for telephone speech diarization’, *IEEE/ACM Transactions on Audio, Speech and*



*Language Processing (TASLP)*, 2014, vol. 22, no. 1, pp. 217–227

- 7 Ehsan Variani, X.L., Erik McDermott, Ignacio Lopez Moreno, Javier Gonzalez-Dominguez: ‘Deep neural networks for small footprint text-dependent speaker verification’, *Acoustics, Speech and Signal Processing (ICASSP)*, 2014 IEEE International Conference on, 2014, pp. 4052–4056
- 8 Yu-hsin Chen, I.L.-M., Tara N Sainath, Mirk’o Visontai, Raziell Alvarez, Carolina Parada: ‘Locallyconnected and convolutional neural networks for small footprint speaker recognition’, *Sixteenth Annual Conference of the International Speech Communication Association*, 2015
- 9 Georg Heigold, I.M., Samy Bengio, Noam Shazeer: ‘End-to-end text-dependent speaker verification’, *Acoustics, Speech and Signal Processing (ICASSP)*, 2016 IEEE International Conference on. IEEE, 2016, pp. 5115–5119
- 10 F A Rezaur Rahman Chowdhury, Q.W., Li Wan, Ignacio Lopez Moreno: ‘Attention-based models for text-dependent speaker verification’, *arXiv preprint arXiv:1710.10470*, 2017
- 11 Li Wan, Q.W., Alan Papir, Ignacio Lopez Moreno: ‘Generalized end-to-end loss for speaker verification’, *arXiv preprint arXiv:1710.10467*, 2017
- 12 Guoguo Chen, C.P., Georg Heigold: ‘Smallfootprint keyword spotting using deep neural networks’, *Acoustics, Speech and Signal Processing*

(ICASSP), 2014 IEEE International Conference on. IEEE, 2014, pp. 4087–4091

13 Rohit Prabhavalkar, R.A., Carolina Parada, Preetum Nakkiran, Tara N Sainath: ‘Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks’, Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE, 2015, pp. 4704–4708

14 Weidi Xie, Arsha Nagrani, Joon Son Chung, and Zisserman, A.: ‘UTTERANCE-LEVEL AGGREGATION FOR SPEAKER RECOGNITION IN THE WILD’, ICASSP, 2019

15 Philip Andrew Mansfield, Q.W., Carlton Downey, Li Wan, and Ignacio Lopez Moreno: ‘Links: A high dimensional online clustering method’, arXiv preprint arXiv:1801.10123, 2018

16 Ramírez, J.J.M.G.J.C.S.: ‘Voice Activity Detection. Fundamentals and Speech Recognition System Robustness’, *Robust Speech Recognition and Understanding*, 2007, pp. 1–22

17 H. Jégou, M.D., C. Schmid, P. Pérez: ‘Aggregating local descriptors into a compact image representation’, Proc. CVPR, 2010

18 Relja Arandjelović, P.G., Akihiko Torii, Tomas Pajdla, Josef Sivic: ‘NetVLAD: CNN architecture for weakly supervised place recognition’, CVPR, 2016

19 Yujie Zhong, R.A., Andrew Zisserman: ‘GhostVLAD for set-based face

recognition', *Asian Conference on Computer Vision, ACCV*, 2018

20 Bredin, H.e.: 'pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems', <http://pyannote.github.io/pyannote-metrics>, 2017