université PARIS-SACLAY

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

NNT : 2019SACLS261

Thèse de doctorat

Steps towards end-to-end neural speaker diarization

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Orsay, le 26/09/2019, par

RUIQING YIN

Composition du Jury :

Anne VILNAT Professeur, Université Paris Sud	Présidente
Sylvain MEIGNIER Professeur, Le Mans Université	Rapporteur
Najim DEHAK Assistant Professor, Johns Hopkins University,	Rapporteur
Jean-François BONASTRE Professeur, Université d'Avignon	Examinateur
Ricard MARXER Maître de conférences, Université de Toulon	Examinateur
Claude BARRAS Maître de conférences, Université Paris Sud	Directeur de thèse
Hervé BREDIN Chargé de Recherche CNRS, LIMSI	Co-encadrant de thèse

Acknowledgements

Firstly, I would like to thank my thesis advisors Hervé Bredin and Claude Barras, for giving me the opportunity to do the internship and the thesis in LIMSI. Working with Hervé during my thesis is an invaluable experience. He can always give some handy advice when I met a problem or had a question about my research, writing, or coding. I am also grateful to Claude for his insights and encouragement on my work. I appreciate all their contributions of time and ideas to make my Ph.D. experience productive.

I would like to thank Jose Patino, Héctor Delgado, Nicholas Evans in EURECOM and Pavel Korshunov, Sébastien Marcel, Alain Komaty in Idiap. We worked together on low-latency speaker spotting and Albayzin Challenge. I am grateful for the collaboration and their good advice.

I would like to thank my thesis committee members. I am grateful to Sylvain Meignier, Najim Dehak, Ricard Marxer, for their time, interest, and helpful comments. I would also like to thank Anne Vilnat, Jean-François Bonastre, for their time and insightful questions.

My time at LIMSI was made enjoyable, mostly due to the many friends and colleagues there. Thanks to Zheng Zhang, Sibo Cheng Ye Hong, Benjamin Maurice, François Buet, Léo Galmant, Aman Zaid Berhe, Yuming Zhai. Thanks to Laurence Rostaing, Sophie Pageau-Maurice for their assistance in administrative procedure and thank all other members in LIMSI,

I gratefully acknowledge the ANR ODESSA (ANR-15-CE39-0010) project for funding my Ph.D. work.

Finally, I would also like to express my gratitude to my family for their moral support and warm encouragement.

Abstract

Speaker diarization is the task of determining *"who speaks when"* in an audio stream that usually contains an unknown amount of speech from an unknown number of speakers. Speaker diarization systems are usually built as the combination of four main stages. First, non-speech regions such as silence, music, and noise are removed by Voice Activity Detection (VAD). Next, speech regions are split into speaker-homogeneous segments by Speaker Change Detection (SCD), later grouped according to the identity of the speaker thanks to unsupervised clustering approaches. Finally, speech turn boundaries and labels are (optionally) refined with a re-segmentation stage. In this thesis, we propose to address these four stages with neural network approaches.

We first formulate both the initial segmentation (voice activity detection and speaker change detection) and the final re-segmentation as a set of sequence labeling problems and then address them with Bidirectional Long Short-Term Memory (Bi-LSTM) networks.

In the speech turn clustering stage, we propose to use affinity propagation on top of neural speaker embeddings. Experiments on a broadcast TV dataset show that affinity propagation clustering is more suitable than hierarchical agglomerative clustering when applied to neural speaker embeddings. The LSTM-based segmentation and affinity propagation clustering are also combined and jointly optimized to form a speaker diarization pipeline. Compared to the pipeline with

independently optimized modules, the new pipeline brings a significant improvement. In addition, we propose to improve the similarity matrix by bidirectional LSTM and then apply spectral clustering on top of the improved similarity matrix. The proposed system achieves state-of-the-art performance in the CALLHOME telephone conversation dataset.

Finally, we formulate sequential clustering as a supervised sequence labeling task and address it with stacked RNNs. To better understand its behavior, the analysis is based on a proposed encoder-decoder architecture. Our proposed systems bring a significant improvement compared with traditional clustering methods on toy examples.

Résumé

La tâche de segmentation et de regroupement en locuteurs (*speaker diarization*) consiste à identifier *" qui parle quand "* dans un flux audio. Plus précisément, il s'agit d'un processus non supervisé qui a pour objectif d'identifier les différents locuteurs d'un flux audio et de déterminer quel locuteur est actif à chaque instant. Le plus souvent, le nombre de locuteurs ou leurs identités ne sont pas connus à l'avance ; l'objectif est donc d'attribuer à chaque locuteur un identifiant anonyme unique. C'est une technologie clef dans des domaines comme la recherche d'information par le contenu, la biométrie vocale ou l'analyse des comportements sociaux. Les systèmes de segmentation et de regroupement en locuteurs sont généralement construits en combinant quatre étapes principales. Premièrement, les régions ne contenant pas de parole telles que les silences, la musique et le bruit sont supprimées par la détection d'activité vocale (voice activity detection). Ensuite, les régions de parole sont divisées en segments homogènes en locuteur par détection des changements de locuteurs (speaker change detection), puis regroupées en fonction de l'identité du locuteur (clustering). Enfin, les frontières des tours de parole et leurs étiquettes sont affinées avec une étape de re-segmentation. Dans cette thèse, nous proposons d'aborder ces quatre étapes avec des approches fondées sur les réseaux de neurones.

Nous formulons d'abord le problème de la segmentation initiale (détection de l'activité vocale et des changements entre locuteurs) et de la re-segmentation finale sous la forme d'un ensemble de problèmes d'étiquetage

de séquence basés sur les Mel-Frequency Cepstral Coefficients (MFCC), puis nous les résolvons avec des réseaux neuronaux récurrents de type LSTM bidirectionnels (*Bidirectional Long Short-Term Memory*). Pour la détection de parole ou la segmentation en tours de parole, l'ensemble de nos expériences sur la base de données télévisées ETAPE montrent que les réseaux neuronaux récurrents fonctionnent mieux que les modèles classiques par mélanges de Gaussiennes, en particulier sur la qualité des frontières.

Au stade du regroupement des régions de parole, nous proposons d'utiliser l'algorithme de propagation d'affinité (affinity propagation) à partir de plongements neuronaux de ces tours de parole dans l'espace vectoriel des locuteurs.

Des expériences sur la base de données télévisées ETAPE montrent que le regroupement par propagation d'affinité est plus approprié que le regroupement hiérarchique agglomératif (hierarchical agglomerative clustering) lorsqu'il est appliquée à des plongements neuronaux de locuteurs qui permettent une projection discriminante des segments de parole. La segmentation basée sur les réseaux récurrents et la propagation d'affinité sont également combinées et optimisées conjointement pour former une chaîne de regroupement en locuteurs. Comparé à un système dont les modules sont optimisés indépendamment, la nouvelle chaîne de traitements apporte une amélioration significative.

De plus, nous proposons d'améliorer l'estimation de la matrice de similarité par des réseaux neuronaux récurrents, puis d'appliquer un partitionnement spectral à partir de cette matrice de similarité améliorée. Le système proposé atteint des performances à l'état de l'art sur la base de données de conversation téléphonique CALLHOME issue de la campagne NIST 2000 Speaker Recognition Evaluation (SRE 2000).

Enfin, nous formulons le regroupement des tours de parole en mode séquentiel sous la forme d'une tâche supervisée d'étiquetage de séquence et abordons ce problème avec des réseaux récurrents empilés semblable à la détection d'activité vocale et détection des changements de locuteurs. Pour mieux comprendre le comportement du système, une analyse basée sur une architecture de codeur-décodeur est proposée. Sur des exemples synthétiques, nos systèmes apportent une amélioration significative par rapport aux méthodes de regroupement traditionnelles telles que le regroupement hiérarchique agglomératif et la propagation d'affinité.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the decreasing cost of storage and the development of Internet and social media, every day, millions of audio and video recordings are being produced and distributed, including broadcast news, telephone, meeting, lecture, TV series, etc. As the amount of available data grows, finding useful information becomes more difficult.

Imagine a meeting or an interview where the discussions are only recorded. If you want to find the desired information, you should spend several hours listening to the recordings. However, if the recordings are split and annotated with speaker names, background noise, music, together with a transcript obtained by an Automatic Speech Recognition (ASR) system, it will be more efficient to search and index the useful information.

As described in [7], audio diarization is defined as *the process of annotating an input audio channel with information that attributes (possibly overlapping) temporal regions of signal energy to their specific sources. These sources can include particular speakers, music, background noise sources and other signal source/channel characteristics.* The types and details of the audio sources are application specific. When audio sources are speakers, this task is called speaker diarization. Generally, speaker diarization is the task of determining "who speaks when" in an audio file that usually contains an unknown number of speakers. A

speaker diarization system involves splitting the audio into speaker-homogeneous segments (segmentation) and then grouping them by speaker identities (clustering). Since it is an unsupervised process, the output of the system is a set of segments with unique identifiers for different speakers.

Speaker diarization is often used as a preprocessing step in some other applications. In ASR, speaker diarization output is used to adapt the acoustic models to each speaker in order to improve the accuracy of the transcription. For speaker recognition and verification, speaker diarization can remove the non-speech part by Voice Activity Detection (VAD) and accumulate more information for a speaker. In addition, speaker diarization enables other high-level applications such as summarization.

1.1 Motivations

Speaker diarization has been applied in many audio domains. Current speaker diarization systems perform well for some domains such as phone calls which usually contain two dominant speakers in each recording. However, speaker diarization is still a hard task in other domains such as meeting recordings, child language recordings, clinical interviews, etc [8]. In most of the conversations, there are more than two speakers, and they will interrupt each other. In addition, conversations usually contain different types of noise, spontaneous speech, and short speaker turns. Traditional statistical methods cannot achieve good performance in these challenging scenarios.

In recent years, the performance of the state-of-the-art speaker verification system has improved enormously thanks to the neural network (especially deep learning) approaches. The neural-based approaches show much better performance than i-vector and other statistical methods, especially for short duration utterances [9]. In addition, Recurrent Neural networks (RNN) have been used successfully for sequence-to-sequence tasks such as sequence labeling [10], lan-

guage modeling [11] and machine translation [12]. That may be because the RNN is able to learn the context required to make predictions. Those successful applications of neural network approaches motivate us to apply neural networks to the speaker diarization task.

1.2 Objectives

The main objective of this thesis is to apply neural network approaches to the speaker diarization task. In details, the objectives are summarized as follows:

1. Propose a neural network model for the segmentation task. In speaker diarization system, the segmentation includes voice activity detection, speaker change detection, and re-segmentation. All of them can be formulated as a set of sequence labeling problems, addressed using recurrent neural networks.

2. Extract the high-level features from audio segments by existing neural speaker embedding system [13; 14]. Then assess the adequacy of the standard Hierarchical Agglomerative Clustering (HAC) with these features and compare it to alternative approaches like affinity propagation [15] and spectral clustering [16].

3. Propose a new neural network architecture for end-to-end sequential clustering. Conversations between several speakers are usually highly structured and turn-taking behaviors are not randomly distributed over time. The proposed architecture should be able to take the sequential information into consideration.

1.3 Overview of the Thesis

- Chapter 2 (State of the Art): This chapter reviews each step of common speaker diarization pipelines. For each step, the different methods are also introduced and compared. It also introduces the various input features for speaker diarization task and the most used probabilistic models and neural network models. Finally, it reviews the databases used for this thesis and the evaluation metrics to evaluate the sub-modules and diarization outputs.

- Chapter 3 (Neural Segmentation): This chapter explains how to model the segmentation (voice activity detection, speaker change detection, and re-segmentation) as sequence labeling tasks and addressed with Recurrent Neural Networks (RNN). The experiments are done on broadcast news corpora.

- Chapter 4 (Clustering Speaker Embeddings): This chapter splits the clustering into three steps: speech turn embedding, similarity matrix measurement, and actual clustering. The first two steps are addressed with neural network approaches in this chapter. It first reviews the neural-based speaker embedding systems and shows how to extract the embedding vectors from speech segments with variable lengths. Then it compares the Affinity Propagation (AP) and Hierarchical Agglomerative Clustering (HAC) on top of the embedding vectors of segments. Finally, it introduces how to use RNN to improve the similarity matrix and apply spectral clustering with the improved similarity matrix.

- Chapter 5 (End-to-End Sequential Clustering): This chapter introduces a Proof of Concept (PoC) of a fully end-to-end neural speaker diarization system. It first proposes to jointly optimize hyper-parameters of the whole diarization pipeline. Then the clustering step is also formulated as a sequence labeling task and addressed with RNN like VAD and SCD.

- Chapter 6 (Conclusions and Perspectives): This chapter summarizes the conclusions and contributions of this thesis. It also proposes some possible perspectives.

-

Chapter 2

State of the Art

Introduction

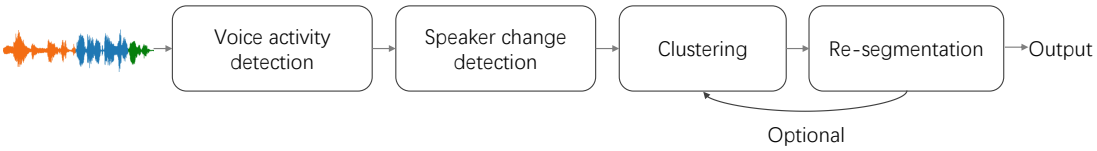

Figure 2.1: Diarization pipeline.

Speaker diarization is the task of determining *"who speaks when"* in an audio stream that usually contains an unknown amount of speech from an unknown number of speakers [7; 17].

Most speaker diarization systems are usually built as the combination of four main stages as shown in Figure 2.1. First, non-speech regions such as silence, music, and noise are removed by Voice Activity Detection (VAD). Next, speech regions are split into speaker-homogeneous segments by Speaker Change Detection (SCD). Then, segments are grouped according to the identity of the speaker thanks to unsupervised clustering approaches. Finally, speech turn boundaries and labels are (optionally) refined with a re-segmentation stage. In some research papers, several alternations of clustering and re-segmentation are performed until

convergence.

This chapter reviews the literature related to the speaker diarization task. The overview starts with an introduction of feature extraction, where we review the most used features for speech processing. Next, the modeling methods and the main stages of speaker diarization systems are reviewed. Finally, the datasets for experiments and the evaluation metrics are introduced.

2.1 Feature extraction

Feature extraction is a dimensionality reduction process that converts the raw speech signal into a sequence of acoustic feature vectors. Speaker diarization aims at grouping audio signal into speaker-homogeneous segments, the extracted feature should therefore carry the speaker-specific characteristics to enable a system to distinguish and separate different speakers in conversations recordings. An ideal feature extractor should maintain both high inter-speaker and low intra-speaker discrimination at the same time. In this section, features are divided into three categories: short-term features, dynamic features, and prosodic features. Other high-level features will be discussed later when needed.

2.1.1 Short-term features

Short-term features are based on the analysis of short frames of speech. The lengths of frames range between 20ms to 40ms, where speech could be regarded as pseudo-stationary signal. Adjacent frames usually have from 50% to 75% overlap to prevent lacking information. The most widely used short-term features for speaker diarization systems are Mel Frequency Cepstral Coefficients (MFCC) [18]. Other short-term features include Linear Frequency Cepstral Coefficients (LFCC) [19], Perceptual Linear Predictive (PLP), Linear Predictive Coding (LPC). Even though those short-term features were first introduced for Automatic Speech Recognition (ASR) to capture the phonetic information and

not for distinguishing speakers, they are widely used and yield good performance in speaker recognition and verification tasks. The reason may be that those features rely on the human hearing perception (MFCC, PLP) or the human speech production (LPC) and they should carry enough information to identify the speakers, through a compact representation of the short-term vocal track configuration.

2.1.2 Dynamic features

Dynamic features describe the time varying information of audio signal such as the change of formant and energy. Dynamic information is very important for speech recognition and speaker recognition, but simple models may hardly catch this information from the presented short-term features. The most used dynamic feature include the delta (first derivative) and double-delta (second derivative) of short-term features (MFCC, LPCC). It was observed that our diarization system improved significantly when using MFCC dynamics. Some other dynamic features are introduced in [20].

2.1.3 Prosodic features

Prosodic speech features are often used to extract information about the speaking style of a person. Different from short-term features extracted from acoustic frames, prosodic speech features are based on speech segments such as syllable, word, or sentences. The fundamental frequency [21], formants, duration, and frame energy are the most used prosodic features. Prosodic features and their dynamics have been successfully applied in speaker recognition task [22; 23]. [24] shows that prosodic features and other long-term features can be combined with short-term features to improve the speaker diarization result.

2.2 Modeling

In speech processing, different models have been applied to model speech/non-speech, phoneme, and speakers. Probabilistic models such as Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM) have been widely used in the literature. In recent years, with the increase of available annotated data, neural network models achieve state-of-the-art performance on numerous tasks.

2.2.1 Gaussian Mixture Models (GMM)

A Gaussian Mixture Model (GMM) is a generative model that assumes all data points are generated from a mixture of some Gaussian distributions. The probability density function is a weighted sum of Gaussian component densities:

$$p(\boldsymbol{x}|\Theta) = \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}, \theta_k) \tag{2.1}$$

where Θ is the set of parameters in GMM, the sum of weights $\sum_{k=1}^{K} \pi_k = 1$, and $N(\boldsymbol{x}, \theta_k)$ is a multivariate Gaussian:

$$N(\boldsymbol{x}, \theta_k) = \frac{1}{(2\pi)^{D/2} \left|\boldsymbol{\Sigma}_k\right|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top {\boldsymbol{\Sigma}_k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)} \tag{2.2}$$

The data point dimension is D. $\boldsymbol{\mu}_k$ is the mean vector and $\boldsymbol{\Sigma}_k$ is the covariance matrix.

In speaker diarization, GMM modeling is widely used to model speech/non-speech and speakers. An utterance u can be represented by a sequence of feature vectors extracted from acoustic frames. Each feature vector represents a data point generated by the GMM and all the data points are treated independently from each other. The generative probability of u is the product of all the generative probability of data points. The parameters Θ of a GMM can be estimated via the Expectation Maximization (EM) algorithm based on a collection of training

data.

2.2.2 Hidden Markov Models (HMM)

A Hidden Markov Model (HMM) [25] is a probabilistic model that includes two types of random variables: hidden states x_t and observations $\boldsymbol{y}_t$. The hidden (unobservable) state sequence $X = x_1, x_2, \ldots, x_T$ is assumed to be a Markov chain where the conditional probability distribution of the hidden state at time t depends only on the value of the hidden state x_{t-1}. The observation at time t is generated by the hidden state x_t and it can be either discrete or continuous. A HMM can be specified by the following parameters:

- $\boldsymbol{\pi}$, the initial probability vector of the first hidden state.
- $\boldsymbol{A}$, the transition probability matrix where $\boldsymbol{A}_{ij} = P(x_t = s_j \mid x_{t-1} = s_i)$ represents the probability of transition from state s_i to state s_j.
- Emission probability of observation given a hidden state. When the observation is discrete, it is a matrix $\boldsymbol{B}$ where $\boldsymbol{B}_{ik} = P(y_t = o_k \mid x_t = s_i)$. When the observation is a continuous $\boldsymbol{y} \in \mathbb{R}^D$, $P(\boldsymbol{y}_t \mid x_t = s_i)$ is usually modeled by a GMM.

HMM can solve three basic problems:

1. Evaluation problem: given the model parameters, compute the likelihood of an observation sequence.
2. Decoding problem: given the model parameters, choose an optimal hidden state sequence of an observation sequence. This is solved by the Viterbi algorithm.
3. Learning problem: estimate the optimal model parameters from the observation data. This is solved by the Baum-Welch algorithm.

2.2.3 Neural networks

Neural networks is a representation learning method inspired by the mechanism in the human brain, which aims to automatically learn the representations needed for detection or classification tasks from raw data [26]. Different from probabilistic models required to design a complex model, neural network models are composed of hierarchical architectures with multiple simple but non-linear layers. Each layer is composed of a number of nodes, which make decisions (activation) based on their inputs. This architecture is similar to a real nervous system, with each node acting as a neuron within a large network.

Neural networks learn representations of data with multiple levels of abstraction progressively as it goes through the network layers. Lower layers learn low-level representation and feed into higher layers, which can learn representation at a more abstract level. For classification tasks, representation outputs in higher layers amplify aspects of the input that are important for discrimination and suppress irrelevant variations [26]. For example, in image recognition [27], the input data is an array of pixel values, and the learned feature in the first layer may be oriented edges. The second layer may learn the combinations of edges such as corners, angles, and surface boundaries in the images. The subsequent layers may learn an object by combining the features learned in previous layers. The key advance of neural networks is that the hierarchical representations are not designed by human engineers: they are learned from data by using the backpropagation algorithm [26]. Thanks to the increasing amount of available datasets (ImageNet [28], Voxceleb [29] *etc.*) and the wide use of Graphics Processing Unit (GPU), neural network models have dramatically improved the state-of-the-art in different tasks.

Neural network models include three important parts: architecture, loss function, and optimizer. In this section, four most used neural network architectures are presented from Section 2.2.3.1 to Section 2.2.3.4. The loss function and optimizer are introduced in Section 2.2.3.5.

2.2.3.1 Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP), also called feedforward neural network, is one of the most used neural networks architecture. As shown in Figure 2.2, MLP is composed by three parts: an input layer to receive the input data, an output layer to make predictions about the input and in between are several hidden layers. Each layer contains a number of nodes with connections feeding forward to the neurons in the next layer. The value o of each node in a hidden layer is defined as:

$$o = f(\sum_{i=0}^{m} \boldsymbol{w}_i \boldsymbol{x}_i + b) \tag{2.3}$$

where $\boldsymbol{x}$ is the values of nodes from the previous layer, $\boldsymbol{w}$ is the vector of weights and b is the bias. The linear part $\sum_{i=0}^{m} \boldsymbol{w}_i \boldsymbol{x}_i + b$ can be rewritten by matrix multiplication. f is the activation function. The most used activation functions are listed in Table 2.1.

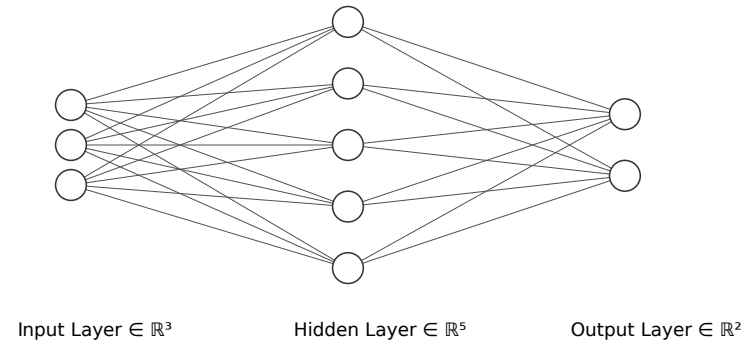

Figure 2.2: A 2-layer Neural Network (one hidden layer of 4 neurons and one output layer with 2 neurons), and three inputs.

2.2.3.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a specialized kind of neural network for processing data with a grid-like topology such as time-series data and image data [1]. Similar to MLP, CNN consists of an input layer, an output layer, and

Name	Formula
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
tanh	$\tanh(x)$
ReLU [30]	$\max(0, x)$
Leaky ReLU [31]	$\max(0.1x, x)$
ELU [32]	$\begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$
Maxout [33]	$\max(W_1 x + b_1, W_2 x + b_2)$

Table 2.1: Examples of activation functions.

several hidden layers. The matrix multiplication in MLP hidden layers is replaced by convolution to simplify the computation. Beside convolutional layers, hidden layers in CNN include pooling layers, fully connected layers, and normalization layers. The following is a brief description of the convolution layer and pooling layer.

Convolution layer Convolutional layers apply a convolution operation to the input which involves two arguments: **input** and **kernel**. If we use a two-dimensional image I as input, a two-dimensional kernel K should be applied. The convolution is defined like:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \tag{2.4}$$

where S is the output, sometimes also called feature map. m, n are the indices of kernel K. As shown in Figure 2.3, the kernel slides through the input, and all computations share the same parameters. Compared to the matrix multiplication in MLP, convolution needs less free parameters, and the parameter sharing strategy allows the CNN network to be deeper with fewer parameters. Each CNN layer usually contains multiple kernels. To keep the output dimension the same as input, padding operations should be applied before convolution.

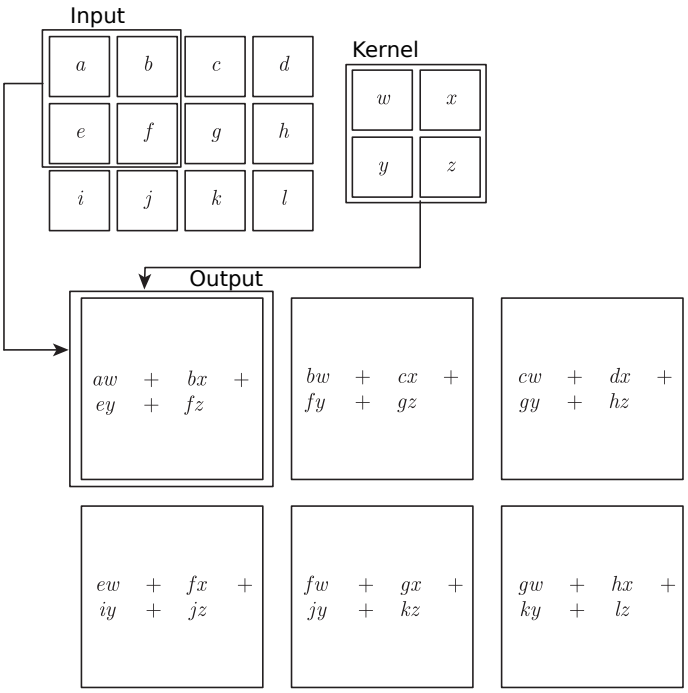

Figure 2.3: An example of 2-D convolution. Figure taken from [1].

Pooling layer A pooling layer is typically applied right after the convolution layer. It replaces the output of previous layers at a certain location with a summary statistic of the nearby outputs. It helps to reduce the spatial size of the input and extracts representations approximately invariant to small translations. The most used pooling function is the max pooling and the average pooling. Other pooling functions include stochastic pooling, L2 norm pooling *etc.*

2.2.3.3 Recurrent Neural Network (RNN)

Recurrent neural network (RNN) is a type of neural networks used to process sequential data. The parameter sharing strategy is also applied in RNN architecture and makes it possible to process sequences of variable length. In the convolution operation, the same convolution kernel is applied at each time step, and the corresponding output is a function output over a small number of neighboring members of the input. In RNN, it works differently. RNN has a recurrent

connection from the current hidden unit to the next hidden unit. Each member of the output sequence is produced using the same update rule applied to the current hidden state and current input. Figure 2.4 shows a traditional RNN for

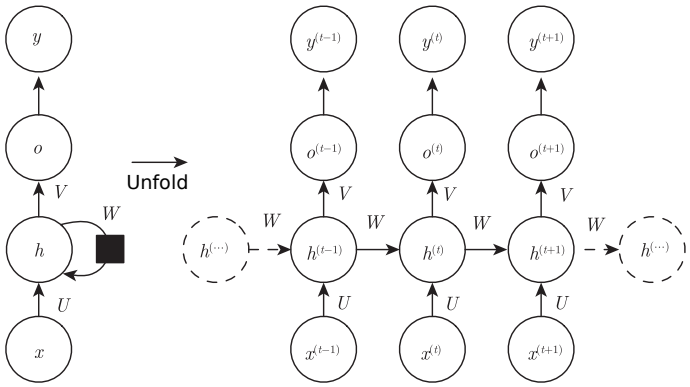

Figure 2.4: The computational graph of RNN. Figure taken from [1] with a few modifications.

the sequence classification task. The update rule is defined as follows:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \tag{2.5}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}), \tag{2.6}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \tag{2.7}$$

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(\boldsymbol{o}^{(t)}), \tag{2.8}$$

where $\boldsymbol{x}^{(t)}$ is the input vector at timestep t, $\boldsymbol{h}^{(t)}$ is the hidden state, $\hat{\boldsymbol{y}}^{(t)}$ is the corresponding output, $\boldsymbol{b}$, $\boldsymbol{W}$ and $\boldsymbol{U}$ are shared parameters. softmax is the most used activation function at output layer for classification task. It is defined as follow:

$$\text{softmax}(\boldsymbol{o})_i = \frac{e^{\boldsymbol{o}_i}}{\sum_{j=1}^{K} e^{\boldsymbol{o}_j}} \tag{2.9}$$

where the K is the number of categories.

Long Short-Term Memory networks (LSTM) For standard RNNs, it is difficult to learn long-term dependencies because gradients propagated over many stages tend to either vanish or explode. Long Short-Term Memory network (LSTM) is designed to overcome this problem. Its structure resembles a standard RNN with a hidden layer, but each ordinary node in the hidden layer is replaced by a memory cell which is shown in Figure 2.5. Each memory cell contains a node with a self-connected recurrent edge with minor linear interactions, ensuring that the gradient can pass across many time steps without vanishing or exploding. Other variants of LSTM such as GRU [34] are also widely used.

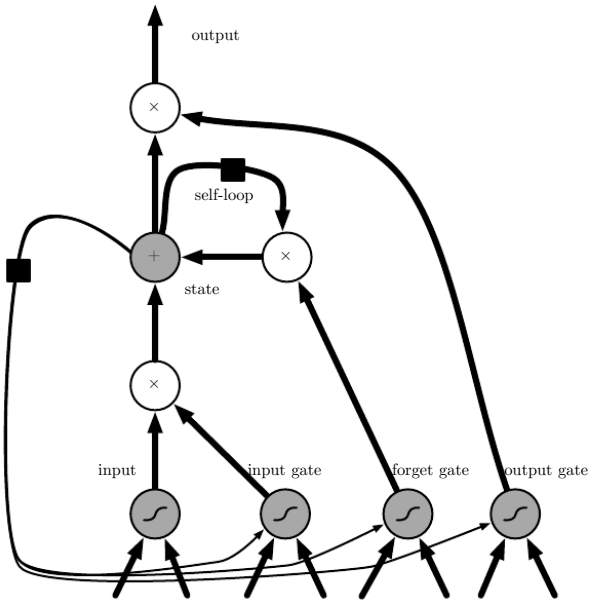

Figure 2.5: A LSTM memory block with one cell. Cells are connected recurrently to each other and have gates to control whether the cell can be overwritten by an input, forgotten, or allowed to be fed to the output gates. Figure taken from [1].

Bidirectional RNN In a bidirectional RNN, there are two layers of hidden nodes. Both hidden layers are connected to input and output. The first layer is the same as a standard RNN, which has recurrent connections from the past time steps while in the second layer, the direction of recurrent of connections is

flipped in order to pass information backward along the sequence. In other words, bidirectional RNN can be realized by a forward RNN layer and a backward one. Then the concatenation of outputs is passed to the next layer. By using bidirectional RNN, the state at the current time step can use the context information from the past and the future, which is helpful for speaker modeling and other sub-tasks in speaker diarization.

2.2.3.4 Encoder-decoder

Standard RNN can map an input sequence to an output sequence of the same length or to a fixed-size vector (the hidden state at the last timestep). In some other sequence-to-sequence tasks such as speech recognition and machine translation, the input sequence and the output sequence may have different sizes. Encoder-decoder is designed to solve these tasks. The traditional encoder-decoder architecture [34] is shown in Figure 2.6. An encoder RNN processes the input sequence and outputs the context vector $\boldsymbol{c}$, which represents a summary of the input sequence. Usually, $\boldsymbol{c}$ is the final hidden state in RNN. Another decoder RNN is used to generate the output sequence with the context $\boldsymbol{c}$. In [2], attention mechanism is introduced to encoder-decoder in order to use different context vectors at each time step.

2.2.3.5 Loss function and optimization

In machine learning tasks, the loss function or objective function represents the inaccuracy of predictions. These tasks can be considered as optimization problems seeking to minimize a loss function. The most used loss functions include mean squared error, binary cross-entropy, and category cross-entropy.

Gradient descent is a common method to solve optimization problems, especially when the objective function is convex. However, in neural network models, we do not use the gradient descent directly. The main reason is that the train set becomes so big that it is expensive to compute the gradient. In addition, the

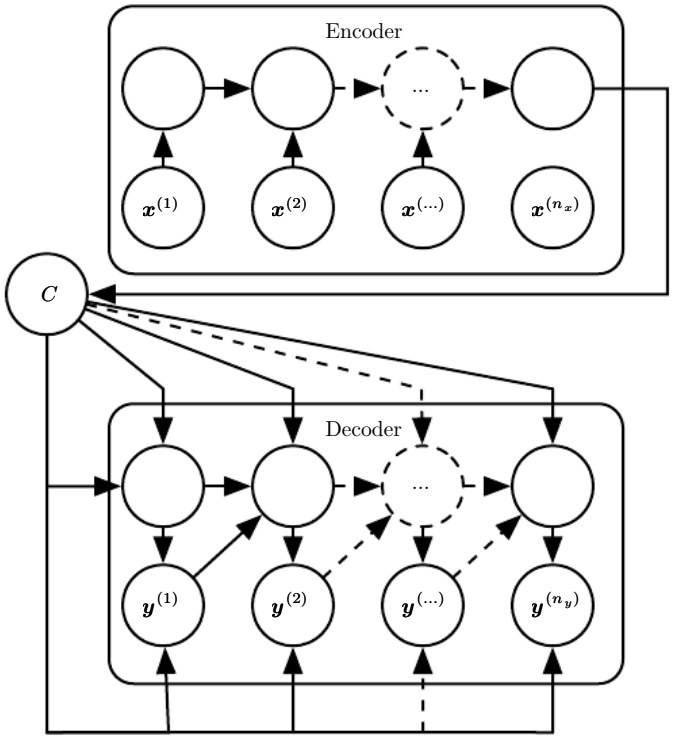

Figure 2.6: Encoder-decoder architecture introduced in [2]. Figure taken from [1].

objective functions are typically non-convex, and the result may converge to a local optimum. Stochastic Gradient Descent (SGD), also known as incremental gradient descent is widely used in neural network models, which is a stochastic approximation of the gradient descent. It seeks minima by iteration with a learning rate. The train set is divided into several batches. Each iteration just uses one batch and does the gradient descent with it. SGD method can diverge or converge slowly if the learning rate is set inappropriately. There are also many alternative advanced methods. For example, Momentum, Nesterov accelerated gradient, Adagrad, Adadelta, RMSprop, Adam. A brief introduction to these methods can be found in [35].

2.2.4 Speaker Modeling

The features introduced in 2.1 are not only representing the individual characteristics of speakers but also some interfering sources such as background noise, music, and channel. To find an invariant speaker representation and build more robust speaker verification and diarization systems, researchers design speaker models with the original acoustic features. In recent years, probabilistic speaker models and neural network based speaker models are mostly used.

2.2.4.1 Probabilistic speaker model

Probabilistic speaker model aims at factorizing the speech signal features into factors related to speakers and other variations. A classical probabilistic speaker model is the Gaussian Mixture Model-Universal Background Model (GMM-UBM). The UBM is a model that represents general, person-independent feature characteristics. UBM is usually represented by a GMM and trained with a lot of data [36]. The speaker model is derived from the UBM by Maximum a Posteriori (MAP) Adaptation [37]. GMM-UBM is extended to a low-rank formulation, leading to the Joint Factor Analysis (JFA) model that decomposes speech signal into speaker independent, speaker dependent, channel dependent, and residual components [38]. I-vector model [39] is a simplified version of JFA and it became the state-of-the-art in early 2010. The speaker dependent and channel dependent factors are replaced by a total variability factor:

$$\boldsymbol{s} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{w} \tag{2.10}$$

where $\boldsymbol{s}$ is the utterance supervector, $\boldsymbol{m}$ is a speaker and channel independent supervector from UBM, $\boldsymbol{T}$ is the total variability matrix, and $\boldsymbol{w}$ is the i-vector. If $\boldsymbol{T}$ is given, i-vector can be extracted from speech utterances. Therefore, i-vector system can be used as a feature extractor to extract a low-dimensional fixed-size representation vector from a speech utterance.

2.2.4.2 Neural network based speaker model

Although the probabilistic speaker models yield good performance in speaker recognition and diarization tasks, the systems still have an inevitable limitation on robustness against the complex environments (noise, channel, speaking style). The main reason is that the probabilistic model relies on strong prior assumption, and it is difficult to model all the variations from original acoustic features with a GMM. Motivated by the powerful feature extraction capability of deep neural networks (DNNs) applied to speech recognition and face recognition, neural networks are also used to directly model the speaker space. Similar to i-vector introduced in 2.2.4.1, neural network models are often used as a feature extractor and the extracted representation vector are called d-vector [13] or x-vector [14]. In early works, a supervised DNN was trained to classify speakers in a fixed list over the frame level input features. The high-level features are extracted from bottleneck or the last DNN layer and then used to train the speaker model. Probabilistic speaker model introduced in 2.2.4.1 can also be applied over bottleneck features [40; 41]. In [13], the average output of the last hidden layer in DNN is taken as the speaker representation, and it achieves better performance than the i-vector system on a small footprint text-dependent speaker verification task. Instead of stacking frames as input, [42] proposes to use time-delay DNN [43] and a statistics pooling layer to capture long-term speaker characteristics. The speaker representation is the outputs of two affine layers after statistics pooling. In [9], *Heigold et al.* propose an end-to-end text-dependent speaker verification system that learns speaker embeddings based on the cosine similarity. This system is developed to handle variable length input in a text-independent verification task through a temporal pooling layer [44] and data augmentation [14]. The above systems are based on the cross-entropy loss, and encourage the separability of speaker features. However, it is not sufficient to learn features with a large margin. To make features not only separable but also discriminative, researchers in face recognition domain explored discriminative loss functions for

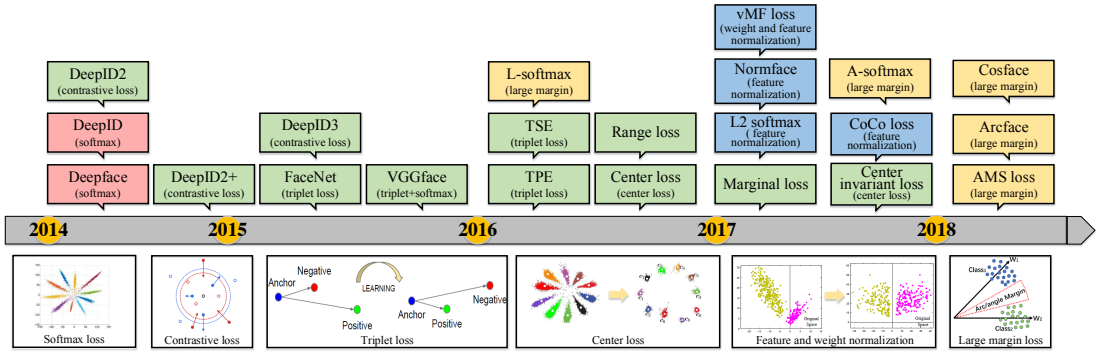

Figure 2.7: The development of loss functions. Figure taken from [3].

enhanced generalization ability [3]. Figure 2.7 shows the development of the loss functions in face recognition domain. The contrastive loss and the triplet loss became the commonly used loss functions in face recognition task [45; 46] and then applied to speaker verification task [47; 48; 49]. They project inputs into Euclidean feature space and compress intra-variance and enlarges inter-variance. During the training process, the contrastive loss and triplet loss occasionally encounter instability and slow convergence due to the selection of training pairs or triplets, [50] proposed center loss to enhance the discriminative power of the deeply learned features. After that, angular/cosine-margin-based loss as well as feature and weight normalization became popular. The neural network systems introduced in this section are also called embedding systems which extract the speaker embedding vectors from audio segments. The similarity between audio segments can be directly computed by cosine metric or Euclidean metric with their embedding vectors. The speaker verification and identification can be done by thresholding the similarities.

2.3 Voice Activity Detection (VAD)

Voice Activity Detection (VAD) is the task of labeling speech and non-speech segments in an audio stream. Non-speech segments may include silence, music,

laughing, and other background noises. VAD is a fundamental task in almost all fields of speech processing tasks such as speech enhancement, speaker recognition, and speech recognition [51]. In speaker diarization task, VAD has a significant impact in two ways. First, the missed and false alarm speech segments contribute directly to the diarization evaluation metrics such as diarization error rate (DER). Poor VAD performance will therefore increase DER. Second, in the clustering step, the missed speech segments reduce the available data for speakers and the false alarm speech segments bring impurities into speaker clusters. So a poor VAD system also leads to an increase of clustering error. Initial speaker diarization system attempted to do VAD in clustering step where non-speech segments were treated as an extra cluster. However, it was observed that using VAD as a pre-processing step can lead to a better result [17].

[17; 51] reviewed different traditional approaches for the VAD task. These approaches can be separated into two categories: rule-based and model-based approaches. In recent years, neural network approaches are also successfully applied to VAD.

2.3.1 Rule-based approaches

Rule-based approaches make the decision of speech/non-speech directly based on the feature of the current observation or frame. The most used feature is energy [52]. A threshold is used on short-term spectral energy to decide whether a region contains speech/non-speech. Other rule-based approaches include spectrum divergence measures between speech and background noise [53], pitch estimation [54], zero crossing rate, and higher-order statistics in the LPC residual domain [55]. These approaches were generally used on telephone speech data and do not require any labeled training data. However, in broadcast news and meeting data, rule-based approaches have proven to be relatively ineffective.

2.3.2 Model-based approaches

Model-based approaches rely on a classifier with two classes: speech and non-speech. Each class is trained on external data. Traditionally, similar to speaker model, speech and non-speech models are estimated with GMMs and the detection is based on Viterbi decoding. In addition, Discriminant classifiers such as Linear Discriminant Analysis (LDA) [56] and Support Vector Machines (SVM) [57] have also been used in VAD task.

More recently, MLP, CNN, and LSTM were also applied to VAD tasks. In [58; 59], a MLP was trained to map long-temporal spectral features to speech/non-speech posterior probabilities. Then two strategies are used in the detection step. The first makes frame-wise speech/non-speech decisions by thresholding on the posterior probability. The second is based on a Viterbi decoder with a 2-state (speech/nonspeech) HMM, which finds a smoother path through the posteriors.

Although model-based approaches show a better performance than rule-based approaches, VAD is still a challenging task in meeting and broadcast TV data.

2.4 Speaker change detection (SCD)

Speaker change detection is an important part of speaker diarization systems. It aims at finding the boundaries between speech turns of one more different speakers in a given audio and then split audio stream into speaker homogeneous segments which will be used for clustering step. Some diarization systems [16; 60] use uniform segmentation directly. However, conversations may have fast speaker interactions, and impure segments will cause confusion in the diarization error rate. In addition, longer segments can get more information to represent the contained speaker.

In conventional speaker change detection methods, one will use two adjacent sliding windows on the audio data, compute a distance between them, then decide (usually by thresholding the distance) whether the two windows originate

from the same speaker. Gaussian divergence [61] and Bayesian Information Criterion (BIC) [62] have been used extensively in the literature to compute such a distance: they have both advantages of leading to good segmentation results and not requiring any training step (other than for tuning the threshold).

Recently, there are some attempts at improving over these strong baselines with supervised approaches. *Desplanques et al.* [63] investigate factor analysis and i-vector for speaker segmentation. *Bredin* [48] proposes to replace BIC or Gaussian divergence by the Euclidean distance between *TristouNet* embeddings, and it brings significant speaker change detection improvement. However, because they rely on relatively long adjacent sliding windows (2 seconds or more), all these methods tend to miss boundaries in fast speaker interactions.

Recently, neural networks were also applied in speaker change detection. In [64], the speaker change detection is formulated as a classification problem and addressed with DNN. The DNN output states correspond to the location of the speaker change points in the speech segment. Results show that the proposed system can reduce the number of missed change points, compared with traditional methods. In [65] the proposed system is based on CNN and fuzzy labeling, and it outperforms the GLR-based system.

Automatic Speech Recognition (ASR) is also used to find candidate speech turn points [66]. Any two segments centered by word boundary positions of the transcript are compared to detect the possible speaker-turn points.

2.5 Clustering

In some scenarios, clustering is the most important step in speaker diarization system. It is an unsupervised problem, and no information about the number of speakers and their identities is provided. In this step, the speaker homogeneous segments obtained from the speaker change detection step will be grouped according to the hypothesized identity of the speaker. The similarity metrics described

in Section 2.4 can also be used to measure the distance between clusters, such as BIC, KL divergence, and GLR. In recent years, motivated by the success of i-vector and d-vector in speaker verification tasks, the input audio segments are first embedded into a fixed-length vectors, and the clustering is done on top of these embedding vectors. Clustering algorithms can be split into offline clustering and online clustering according to the run-time latency.

2.5.1 Offline clustering

Offline systems have access to all the recording data before processing. Therefore, offline clustering systems typically outperform online clustering systems and are mostly used in the literature.

2.5.1.1 Hierarchical clustering

Hierarchical clustering can be categorized into two groups: divisive and agglomerative. In divisive clustering, one cluster is initialized and then divided until the stopping criterion is met. Usually, a single GMM model is first trained on all the speech segments. Then new speaker clusters are added one-by-one iteratively using some selection procedures to identify suitable training data from the single GMM model [17]. Divisive approaches are extremely computationally efficient [67]. However, they are prone to poor speaker model initialization, and they are generally out-performed by the best agglomerative systems [17].

Hierarchical Agglomerative Clustering (HAC) is the most used clustering algorithm in speaker diarization systems. All segments are initialized as single clusters. At each iteration, two clusters with the highest similarity are merged until the similarity score between any two segments is below a given threshold. In traditional methods, similar to divisive clustering, clusters are modeled by GMMs. When two clusters are merged, a new GMM is re-estimated. Distance metrics introduced in Section 2.4 are used to evaluate the similarity, such as:

Bayesian Information Criterion (BIC) [62], KullbackLeibler divergence [61], the Generalized Likelihood Ratio (GLR) [68]

In recent years, i-vector and d-vector introduced in Section 2.2.4, are widely used in speaker verification tasks, both of them can be used as speaker embedding systems to extract representation vectors from audio segments. Then the distance between two representation vectors can be computed by cosine distance, angular distance, or Probabilistic Linear Discriminant Analysis (PLDA) [69]. For clusters, two strategies can be used to compute their distances. The first one is pooling. The segments representation vectors in a cluster are first pooled into a single vector. Then the distances between clusters are computed the same way as segments. The second one is linkage. The distance of two clusters is the distance of the minimum distance (single linkage), maximum distance (completed linkage), and average distance (average linkage) between their members.

2.5.1.2 K-means

K-means is one of the most used clustering algorithms in speaker diarization task. Given cluster number k, it aims at choosing k cluster centers and minimize the average squared distance between each point and its closest cluster center. It works as follows:

1. Choose k initial cluster centers $C = c_1, ..., c_k$. Different methods can be applied in this step, a commonly used method is called Forgy, which randomly chooses k data points from the whole set as the initial cluster centers.

2. Assign each segment to a cluster with the least squared distance between segment and cluster centers. Segment i is noted by x_i and the set of segments in cluster j is noted by C_j.

3. Update the new cluster center by averaging the segment embeddings in the corresponding cluster: $c_i^{(t+1)} = \frac{1}{\left|C_i^{(t)}\right|} \sum_{x_j \in C_i^{(t)}} x_j$

4. Repeat Steps 2 and 3 until convergence

K-means results can be arbitrarily bad compared to the optimal clustering due to the arbitrary initialization step, and initialization with K-means++ [70] can improve both the speed and the accuracy. [71] proposes to initialize the speaker diarization system using the K-means, and it brings an improvement on the CALLHOME dataset. If the number of speakers is unknown in advance, elbow method [16] or average silhouette method can be applied to determine the number of speakers.

2.5.1.3 Spectral clustering

Spectral clustering is also a widely used clustering method in speaker diarization tasks. It outperforms the other clustering methods such as k-means if the cluster shapes are very complex. Given data points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$, and the similarity matrix $\boldsymbol{S} = (s_{ij})$, where s_{ij} is the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. We set $s_{ij} = 0$, when $i = j$. Spectral clustering consists of the following steps:

1. Construct the normalized Laplacian matrix $\boldsymbol{L}_{\text{norm}}$:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S} \tag{2.11}$$

$$\boldsymbol{L}_{\text{norm}} = \boldsymbol{D}^{-1}\boldsymbol{L} \tag{2.12}$$

   where $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} S_{i,j}$.

2. Manually or automatically select the number of clusters k.

3. Compute eigenvalues and eigenvectors of $\boldsymbol{L}_{\text{norm}}$.

4. Take the k largest eigenvalues $\lambda_1, \lambda_2, ...\lambda_k$ and corresponding eigenvectors $\boldsymbol{p}_1, \boldsymbol{p}_2, ...\boldsymbol{p}_k$ of $\boldsymbol{L}_{\text{norm}}$ and form matrix $\boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, ...\boldsymbol{p}_k] \in \mathbb{R}^{n \times k}$.

5. Cluster row vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, ...\boldsymbol{y}_n$ of $\boldsymbol{P}$ using the k-means algorithm. Let's denote the k-means clustering results as $C_1, C_2, ...C_k$, then the final output

clusters $A_1, A_2, ...A_n$ satisfy $A_i = \{j|\boldsymbol{y}_j \in C_i\}$.

The distance metrics introduced in Section 2.4 cannot be used directly as the similarity metric. A normalization should be applied as follows:

$$\boldsymbol{S}_{ij} = exp(\frac{-d^2(x_i, x_j)}{\sigma^2}) \tag{2.13}$$

where $d(x_i, x_j)$ is the distance between segment i and j and σ^2 is a scaling parameter. In [72], $d(x_i, x_j) = 1 - \cos\_score(w_i, w_j)$, w_i, w_j are i-vectors for segment i and j. In [16], with an embedding system proposed in [73], the similarity is the cosine similarity between segment embeddings, and it also proposes some refinements to smooth and denoise the data in the similarity matrix.

2.5.1.4 Affinity Propagation (AP)

Affinity Propagation (AP) [15] is a clustering method based on the concept of "message passing" between data points. It has been used to cluster images of faces, detect genes, and identify representative sentences in an article. Affinity propagation does not require the number of clusters to be determined or estimated before running the algorithm. All data points are potential cluster centers (exemplars). The algorithm should find the exemplars and decide which other data points belong to which exemplar.

The clustering algorithm starts with a similarity matrix S, where $s(i, k)$ $i \neq k$ indicates how well x_k is suited to be the exemplar for x_i. On the diagonal of the similarity matrix, $s(k, k)$ is set to be the preference value, a hyperparameter which influences the choice of exemplars and thus the final number of clusters. The "message passing" process has two kinds of message: responsibility and availability. Responsibility $r(i, k)$ is a message sent from data point i to k that quantifies how well-suited x_k is to serve as the exemplar for x_i. Availability $a(i, k)$ is a message sent from data point k to i that represents how appropriate it would be for x_i to pick x_k as its exemplar. Responsibility and availability are

first initialized to 0 and then iteratively updated by the following formulas:

$$r_t(i,j) = (1-\lambda) r_t^{new}(i,j) + \lambda r_{t-1}(i,j) \tag{2.14}$$

$$a_t(i,j) = (1-\lambda) a_t^{new}(i,j) + \lambda a_{t-1}(i,j) \tag{2.15}$$

where λ is a damping factor introduced to avoid numerical oscillations. $r_t^{new}(i,j)$ is defined as follows:

$$r_t^{new}(i,k) = s(i,k) - \max_{k':\ k' \neq k} [a_{t-1}(i,k') + s(i,k')] \tag{2.16}$$

and

$$a_t^{new}(i,k) = \begin{cases} \min[0, r_{t-1}(k,k) + \sum_{i':i' \notin \{i,k\}} \max[0, r_{t-1}(i',k)]], & \text{if } k = i \\ \sum_{i':i' \neq k} r(i',k), & \text{otherwise} \end{cases} \tag{2.17}$$

At each iteration, affinity propagation combines the responsibilities and availabilities to control the selection of exemplars. For segment i, the segment k which maximizes $r(i,k) + a(i,k)$ is the corresponding exemplar. The whole affinity propagation procedure terminates after a fixed number of iterations or after the exemplar stay unchanged for a chosen number of iterations.

2.5.2 Online clustering

Online clustering should process the data at real time. In other words, the system can only access the data recorded up to the current time. To make sure the data contains enough information for processing, it might allow a latency in output.

A brute-force strategy for online clustering is to re-run the clustering from scratch when a new audio segment comes. But that would be expensive, and bring an issue of temporal discontinuity: the labels obtained from current clustering and previous results may be conflict. To overcome this problem, *Zhu et*

al. propose to use a greedy algorithm [74], where the clustering is run only once after a warm-up period, and then only the existing clusters will be updated. However, the greedy algorithm is significantly less accurate than re-clustering. It is also sensitive to the initial conditions and does not converge to the off-line solution [75]. Another solution proposed in [66; 75] is reconciliation algorithm. It compares the sequences of labels obtained in previous and current cluster sets on the same portion of the audio, and examines all possible permutations of the current labels, then selects the permutation with the lowest Hamming distance between both sequences of labels. In other words, it permutes the current labels to make it similar to the previous ones. To reduce the computational complexity, [66] proposes to use "active window" to limit the history to the N latest segments.

Another naive online clustering method is introduced [16]. When a new segment comes, it is compared with all existing clusters. If the minimum similarity is smaller than a given threshold, then create a new cluster containing only this segment. Otherwise, add this segment to the most similar cluster. [76] proposes unbounded interleaved-state recurrent neural network (UIS-RNN) for clustering. The clustering step is treated as an online generative process of an entire utterance (X, Y), where X is the sequence of segment embeddings and Y is the sequence of speaker labels. Each speaker is modeled by a parameter-sharing RNN, while the RNN states for different speakers interleave in the time domain. The unbounded speaker number is modeled by distance-dependent Chinese Restaurant Process (ddCRP). It also uses an online decoding approach for prediction. This method outperforms the state-of-the-art spectral offline clustering algorithm on the CALLHOME dataset.

2.6 Re-segmentation

Re-segmentation is the final step in most diarization systems. The errors made in VAD and SCD will be accumulated and lead to an increase of clustering errors.

Re-segmentation aims at refining speech turn boundaries and labels. It is usually solved by the Viterbi decoding based on a frame-level, temporally-constrained process with MFCC features. Each state of the HMM represents a speaker or the non-speech and is modeled by a GMM. Transitions between states correspond to speaker turns. Usually, a minimum duration constraint is applied in the decoding process to avoid spurious short speaker turns. The re-segmentation and clustering can be repeated iteratively. In [77], after merging two clusters, the Viterbi re-segmentation and model re-estimation steps are performed. [78] proposes an algorithm for re-segmentation that operates in factor analysis subspace and achieves good performance on the CALLHOME dataset.

2.7 Datasets

In this section, the principal datasets used in our experiments are presented.

2.7.1 REPERE & ETAPE

Dataset	Hours (speech)	nb. of speakers	
		Total	Per file
REPERE	59 (96%)	1758	9.6 ± 6.1
ETAPE TV (train)	14 (94%)	184	9.7 ± 7.6
ETAPE TV (dev.)	4 (93%)	93	8.0 ± 4.4
ETAPE TV (test)	4 (92%)	92	9.2 ± 5.6

Table 2.2: Datasets statistics with mean and standard deviation of speaker counts per file.

Both REPERE [79] and ETAPE TV [80] datasets contain recording of French TV broadcast with news, debates, and entertainment. The annotations for the ETAPE TV dataset were obtained using the following two-steps process: automatic forced alignment of the manual speech transcription followed by manual boundaries adjustment by trained phoneticians. The statistics of REPERE and ETAPE are shwon in Table 2.2.

2.7.2 CALLHOME

CALLHOME is a subset of NIST SRE 2000 (the R65_8_1 folder), which is one of the most used benchmark datasets in diarization papers. It is a collection of telephone call recordings between familiar speakers. It contains 500 utterances distributed across six languages: Arabic, English, German, Japanese, Mandarin, and Spanish. In each conversation, there are between 2 and 7 speakers including 2 dominant speakers (average is 2.57 speaker) and other speakers are from the same channel as either of the 2 main speakers.

2.8 Evaluation metrics

Speaker diarization systems are usually evaluated using Diarization Error Rate (DER). In addition, each stage in the diarization system has its evaluation metric. This section first summarizes the most used evaluation metrics for VAD, SCD, and clustering. Then the DER is introduced.

2.8.1 VAD

The VAD is usually evaluated by False Alarm Error (E_{FA}) and Miss Detection Error (E_{MD}), which are two important parts in DER. E_{FA} is the percentage of time that the hypothesized speech part is labeled as non-speech in the reference:

$$E_{FA} = \frac{|S_{\text{Hyp}} - S_{\text{Ref}}|}{T_{\text{total}}} \tag{2.18}$$

E_{MD} is the percentage of time that the reference speech part is labeled as non-speech in the hypothesis:

$$E_{MD} = \frac{|S_{\text{Ref}} - S_{\text{Hyp}}|}{T_{\text{total}}} \tag{2.19}$$

where S_{Hyp} and S_{Ref} indicate the hypothesized and reference speech part, $|S_{\text{Hyp}} - S_{\text{Ref}}|$ indicates the duration of hypothesized speech not in reference speech and T_{total} is the total duration

The detection error (E_D) for VAD is the sum of E_{FA} and E_{MD}:

$$E_D = E_{FA} + E_{MD} \tag{2.20}$$

2.8.2 SCD

Speaker change detection system is usually evaluated by recall and precision. [48] introduces another evaluation metric: coverage and purity.

2.8.2.1 Recall and precision

Speaker change detection result can be viewed as sequences of 0 and 1. 1 represents the change point or a segment boundary. The comparison process is shown in Figure 2.8. A hypothesis change point is counted as correct if it is within a temporal distance (tolerance) of a reference change point. If more than one predicted change point occurs within the range of tolerance, only the closest one is correct. If a hypothesis change point is not in reference, it is a False Alarm (FA) change point. If a reference change point is not detected by a model, it is a Miss Detection (MD) point.

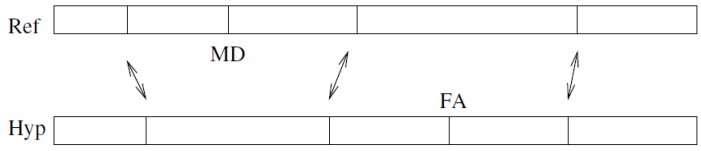

Figure 2.8: False alarm and miss detection. A hypothesis change point will be counted as correct if it is within a tolerance of a reference change point.

The False Alarm Error (E_{FA}) can be computed by the number of false alarm

n_{FA} and the total number of the predicted change points n_H.

$$E_{FA} = \frac{n_{\text{FA}}}{n_H} \tag{2.21}$$

The Missed Detection Error (E_{MD}) can be computed by the number of miss detection n_{MD} and the total number of change points in reference n_R.

$$E_{MD} = \frac{n_{\text{MD}}}{n_R} \tag{2.22}$$

The recall and precision is computed by the following formula:

$$\text{Recall} = 1 - E_{MD} \tag{2.23}$$

$$\text{Precision} = 1 - E_{FA} \tag{2.24}$$

2.8.2.2 Coverage and purity

In recall and precision evaluation metric, a hypothesized change point is counted as correct if it is within the temporal neighborhood of a reference change point. Both values are very sensitive to the actual size of this temporal neighborhood (*aka.* tolerance) – quickly reaching zero as the tolerance decreases. It also means that it is very sensitive to the actual temporal precision of human annotators. Purity and coverage evaluation metrics (as defined in *pyannote.metrics* [81]) do not depend on a tolerance parameter and are more relevant in the perspective of a speaker diarization application. Purity [82] and coverage [83] were introduced to measure cluster quality but can also be adapted to the speaker change points detection task. Given $\mathcal{R}$ the set of reference speech turns, and $\mathcal{H}$ the set of hypothesized segments, coverage is defined as follows:

$$\text{coverage}(\mathcal{R}, \mathcal{H}) = \frac{\sum_{r \in \mathcal{R}} \max_{h \in \mathcal{H}} |r \cap h|}{\sum_{r \in \mathcal{R}} |r|} \tag{2.25}$$

where $|s|$ is the duration of segment s and $r \cap h$ is the intersection of segments r and h. Purity is the dual metric where the role of $\mathcal{R}$ and $\mathcal{H}$ are interchanged. Over-segmentation (*i.e.* detecting too many speaker changes) would result in high purity but low coverage, while missing lots of speaker changes would decrease purity – which is critical for subsequent speech turn agglomerative clustering.

2.8.3 Clustering

Clustering is an unsupervised step and it does not need to identify the speakers by names. Since the speaker labels assigned to both the hypothesis and the reference segmentation are different, an optimal label mapping between the hypothesis and reference files is first done according to the overlap time between speaker-pairs in two sets. Two evaluation metrics are introduced for clustering stage.

2.8.3.1 Confusion

Confusion Error ($E_{\text{confusion}}$) is an important part of DER. Some research papers directly refer to confusion as their DER. Confusion is the percentage of time that the hypothesized speaker is assigned to the wrong speaker in reference:

$$E_{\text{confusion}} = \frac{\sum_{s \in \mathcal{S}} |s| \cdot (\min(N_{hyp}(s), N_{ref}(s)) - N_{\text{correct}}(s))}{\sum_{s \in \mathcal{S}} N_{ref}(s)|s|} \tag{2.26}$$

where $\mathcal{S}$ is the segment set which is obtained by collapsing together the hypothesis and reference speaker turns. $|s|$ is the duration of segment s, $N_{ref}(s)$ and $N_{hyp}(s)$ indicate number of speakers in reference and hypothesis on segment s. N_{correct} indicates the number of speakers in segment s that has been matched correctly between reference and hypothesis. Non-speech segments contain 0 speakers. When all speakers/non-speech are correctly matched in a segment s, the corresponding error is 0.

2.8.3.2 Coverage and purity

While the confusion error provides a convenient way to evaluate the clustering result, purity [82] and coverage [83] are also widely used to analyze the type of errors committed by the system [81]. Purity and coverage are two dual evaluation metrics and are defined as follows:

$$\text{purity} = \frac{\sum_{\text{cluster}} \max_{\text{speaker}} |\text{cluster} \cap \text{speaker}|}{\sum_{\text{cluster}} |\text{cluster}|} \tag{2.27}$$

$$\text{coverage} = \frac{\sum_{\text{speaker}} \max_{\text{cluster}} |\text{speaker} \cap \text{cluster}|}{\sum_{\text{speaker}} |\text{speaker}|} \tag{2.28}$$

where |speaker| (respectively |cluster|) is the speech duration of this particular reference speaker (resp. hypothesized cluster), and |cluster∩speaker| is the duration of their intersection. Over-segmented results (*e.g.* too many speaker clusters) tend to lead to high purity and low coverage, while under-segmented results (*e.g.* when two speakers are merged into one large cluster) lead to low purity and higher coverage.

2.8.4 Diarization error rate (DER)

Speaker diarization systems are usually evaluated and compared using Diarization Error Rate (DER), which is used by NIST in the RT evaluations. It is measured as the fraction of time that is not attributed correctly to a speaker or non-speech, and it is computed as:

$$DER = \frac{\sum_{s \in \mathbb{S}} |s|(\max(N_{ref}(s), N_{hyp}(s)) - N_{\text{correct}}(s))}{\sum_{s \in \mathbb{S}} N_{ref}(s)|s|} \tag{2.29}$$

In addition, DER can be decomposed into three components:

$$DER = E_{FA} + E_{MD} + E_{\text{Confusion}} \tag{2.30}$$

where $E_{\text{Confusion}}$ is the confusion error in clustering step, E_{FA} and E_{MD} are the false alarm error and miss detection error in VAD. The definitions of E_{FA} and E_{MD} in DER are a little different from the evaluation metrics for VAD, where the overlap parts are not taken into consideration. The E_{FA} and E_{MD} in DER are computed as:

$$E_{FA} = \frac{\sum_{s\in\mathcal{S}} \mathbb{1}_{(N_{hyp}(s)-N_{ref}(s)>0)}|s| \cdot (N_{hyp}(s) - N_{ref}(s))}{\sum_{s\in\mathcal{S}} N_{ref}(s)|s|} \tag{2.31}$$

$$E_{MD} = \frac{\sum_{s\in\mathcal{S}} \mathbb{1}_{(N_{ref}(s)-N_{hyp}(s)>0)}|s| \cdot (N_{ref}(s) - N_{hyp}(s))}{\sum_{s\in\mathcal{S}} N_{ref}(s)|s|} \tag{2.32}$$

In order to account for manual annotation imprecision, it is common practice not to evaluate short collars centered on each speech turn boundary (usually 250ms on both sides) and speech regions with more than one simultaneous speaker.

DER for a dataset with multiple audio files is the weighted average DER of individual files. Usually, the corresponding weight is computed according to the total (including overlap part) time that has been evaluated for each file.

Practically, for all experiments in the following chapters, we use the open-source implementation of diarization error rate available in *pyannote.metrics* [81].

Chapter 3

Neural Segmentation

3.1 Introduction

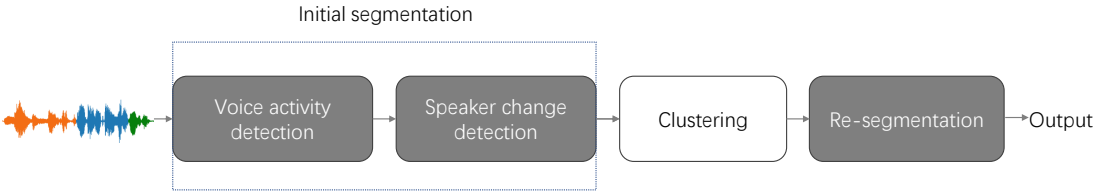

Figure 3.1: Diarization pipeline. In this chapter, we propose to rely on recurrent neural networks for gray modules.

Most diarization systems rely on probabilistic models to address four sub-tasks: Voice Activity Detection (VAD), Speaker Change Detection (SCD), speech turn clustering, and re-segmentation. Usually, VAD and SCD are referred as the initial segmentation which aims at removing non-speech regions in an audio stream and then splitting it into speaker homogeneous segments. The re-segmentation aims at refining speech turn boundaries and labels after clustering.

In recent years, the performance of the state-of-the-art speech and speaker recognition systems has been improved enormously thanks to the neural network (especially deep learning) approaches. In speech recognition and natural language processing, Long Short-Term Memory (LSTM) networks have been

used successfully for sequence labeling [10], language modeling [11] and machine translation [12]. However, existing speaker diarization systems do not take full advantages of these new techniques. As introduced in Chapter 2, conventional initial segmentation and re-segmentation methods still rely on probabilistic models. For example, in speaker change detection, traditional methods are based on two adjacent sliding windows and a distance metric. Gaussian divergence [61] and Bayesian Information Criterion (BIC) [62] have been used extensively in the literature to compute such a distance: they have both advantages of leading to good segmentation results and not requiring any training step (other than for tuning the threshold). There were some recent attempts at improving over these strong baselines, such as factor analysis, i-vector [63] and *TristouNet* [48]. However, because they rely on relatively long adjacent sliding windows (2 seconds or more), all these methods tend to miss boundaries in fast speaker interactions.

Gelly et al. propose to address Voice Activity Detection (VAD) as a frame-wise sequence labeling task on top of MFCC features [84]. Then they apply bidirectional LSTM on overlapping feature sequences to predict whether each frame corresponds to a speech region or a non-speech one.

In this chapter, we first define the generic sequence labeling task. Then the LSTM-based VAD proposed by *Gelly et al.* is reviewed in Section 3.3. Our first contribution is presented in Section 3.4. It is the direct translation of *Gelly*'s work: the SCD is also addressed as a supervised binary classification task (change vs. non-change) using bidirectional LSTM. Our second contribution is introduced in Section 3.5, where we show how to adapt this method to re-segmentation, which is traditionally done using GMM and Viterbi decoding [85]. As shown in Figure 3.1, at the end of this chapter, all modules except the clustering stage will be based on neural networks.

3.2 Definition

Let $\mathbf{x} \in \mathcal{X}$ be a sequence of feature vectors extracted from an audio recording: $\mathbf{x} = (x_1, \ldots, x_T)$ where T is the length of the sequence. Typically, $\mathbf{x}$ would be a sequence of MFCC features extracted on a short (a few milliseconds) overlapping sliding window (*aka.* frame). Let $\mathbf{y} \in \mathcal{Y}$ be the corresponding sequence of labels: $\mathbf{y} = (y_1, \ldots, y_T)$ and $y_i \in \{0, \ldots, K-1\}$. K is the number of classes and depends on the task.

The objective is to find a function $g : \mathcal{X} \to \mathcal{Y}$ that matches a feature sequence $\mathbf{x}$ to the corresponding label sequence $\mathbf{y}$.

3.3 Voice activity detection (VAD)

Voice activity detection (VAD) is an important preprocessing step in almost all speech processing tasks. It is the direct application of the above sequence labeling principle with $K = 2$ classes: $y_i = 1$ for speech, $y_i = 0$ for non-speech. The traditional approaches reviewed in Chapter 2 cannot take full advantage of the contextual information. For example, the energy-based approach predicts the speech/non-speech only based on the current frame. However, the sequence of speech and non-speech in meetings and broadcast news are usually highly structured. For example, in some broadcast news, the music (music is considered as non-speech) is always played after an interview. This type of information is difficult to be modeled by simple approaches. Recently, data-driven modeling methods like neural networks have been applied to VAD.

Gelly et al. propose to model the function g with a stacked LSTMs [84]. MLP is also tested by them, which shows worse performance than LSTM. That may be because MLP only focuses on the current frame like energy-based approach, and cannot make use of any contextual information. Since the VAD system proposed in [84] is used for the speech recognition task, which aims at minimizing the Word Error Rate (WER), the proposed loss functions are related to the WER. However,

in the speaker diarization system, it is not necessary. We simplify the system and propose to train the neural network directly with the binary cross-entropy:

$$L = -\frac{1}{T}\sum_{i=1}^{T} y_i \log(f(\mathbf{x})_i) + (1 - y_i)\log(1 - f(\mathbf{x})_i) \tag{3.1}$$

The actual architecture of the network is composed of Bi-LSTMs and multi-layer perceptrons (MLP) whose weights are shared across the sequence. Bi-LSTMs [86] allow to process sequences in forward and backward directions, making use of both past and future information.

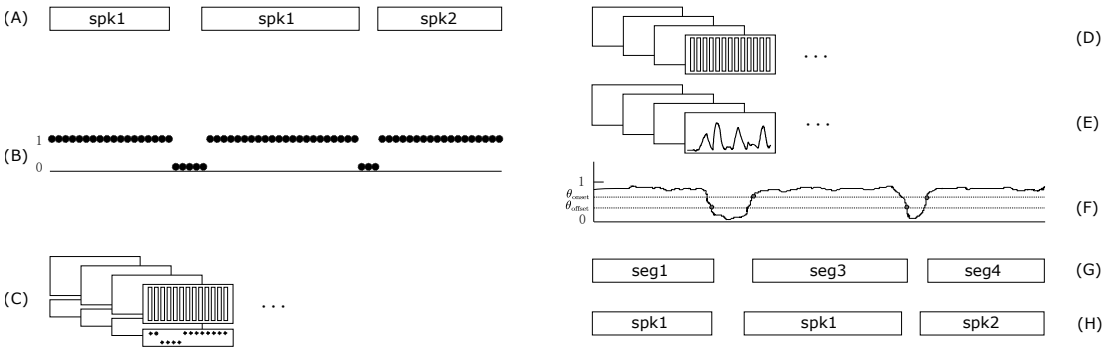

Figure 3.2: Training process (*left*) and prediction process (*right*) for voice activity detection.

3.3.1 Training on sub-sequence

One well-publicized property of LSTMs is that they are able to avoid the *vanishing gradients* problem encountered by traditional recurrent neural networks [10; 87]. Therefore, the initial idea was to train them on whole audio sequences at once, but we found out that this has several limitations, including the limited number of training sequences, and the computational cost and complexity of processing such long sequences with variable lengths. Consequently, as depicted in part C of

Figure 3.2, the long audio sequences are split into short fixed-length overlapping sequences. This has the additional benefit of increasing the variability and number of sequences seen during training, as is usually done with data augmentation for computer vision tasks.

3.3.2 Prediction

Once the network is trained, it can be used to perform voice activity detection as depicted in the right part of Figure 3.2. Similarly to what is done during training, test files are split into overlapping feature sequences (part D of Figure 3.2). The network processes each subsequence to give a sequence of scores between 0 and 1 at the frame level (part E of Figure 3.2). Because input sequences are overlapping, each frame can have multiple candidate scores; they are averaged to obtain the final frame-level score. Then the sequence of speech scores is post-processed using two (θ_{onset} and θ_{offset}) thresholds for the detection of the beginning and end of speech regions [84], as shown in part F of Figure 3.2. Parts G and H respectively represent the hypothesized and groundtruth speech/non-speech parts.

3.3.3 Implementation details

Feature extraction. VAD and the following tasks share the same set of input features extracted every 10ms on a 25ms window using Yaafe toolkit [88]: 19 mel-frequency cepstral coefficients (MFCC), their first and second derivatives, and the first and second derivatives of the energy (amounting to a total of 59 dimensions).

Network architecture. The model for VAD is composed of two bidirectional LSTM layers and two fully connected layers. *Bi-LSTM1* has 64 outputs (32 forward and 32 backward) and *Bi-LSTM2* has 32 outputs (16 forward and 16 backward). The two fully connected layers are 16-dimensional with tanh activation function. The output layer is 1-dimensional with a sigmoid function to

output a SAD score between 0 and 1.

Training. For all experiments in this chapter, subsequences for training are 3.2s long with a step of 800ms (*i.e.* two adjacent sequences overlap by 75%). The actual training is implemented in Python using the Pytorch toolkit, and we use the Stochastic Gradient Descent (SGD) optimizer.

Dataset. All experiments in this chapter are trained on REPERE dataset, tuned on ETAPE development subset and applied on ETAPE test subset.

Hyperparameter tuning. For all experiments in this chapter, the hyperparameters (θ_{onset} and θ_{offset} for VAD) are tuned by *scikit-optimize* [89].

3.3.4 Results and discussion

Methods	Detection error rate(%)	FA(%)	Miss(%)
LSTM	4.93	4.22	0.71
GMM-HMM	7.69	7.51	0.18

Table 3.1: Detection error rates on the ETAPE *Test* dataset for different systems.

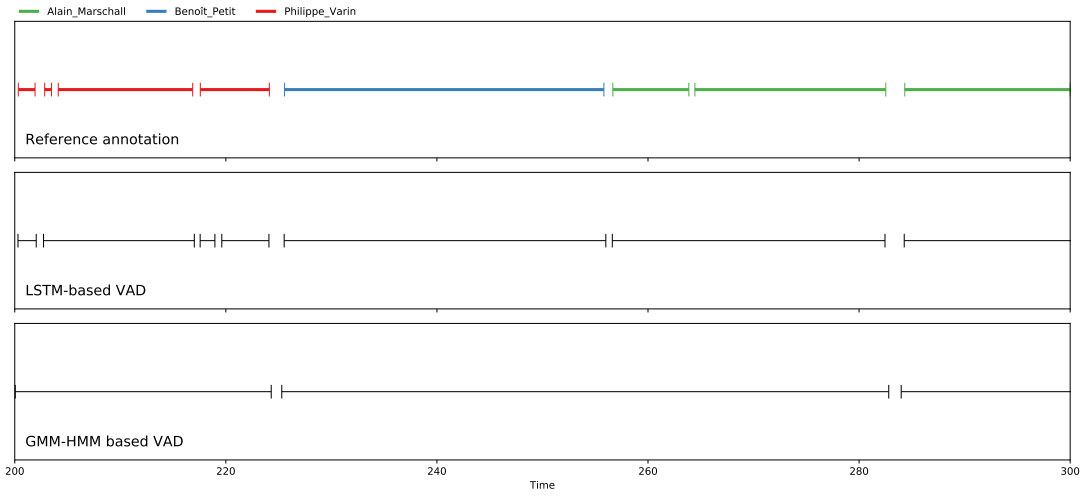

Figure 3.3: Predictions of two different VAD systems on an example from ETAPE dataset.

The detection error rates on the ETAPE dataset of two different VAD systems are shown in Table 3.1. The results of GMM-HMM based VAD system are provided by LIUM [90]. The LSTM-based system is 2.76% better than GMM-HMM based one, that corresponds to a 36% relative improvement. From Table 3.1, we can also find that the improvements of the LSTM-based system are mostly due to the low false alarm error rate. As shown in Figure 3.3, the GMM-HMM based system tends to ignore the short non-speech segments. In addition, the boundaries of segments generated by the LSTM-based system are more precise. This may be because the GMM-HMM based VAD system is a sub-module of a traditional diarization system where the different speakers are modeled by probabilistic model, and long segments are encouraged.

3.4 Speaker change detection (SCD)

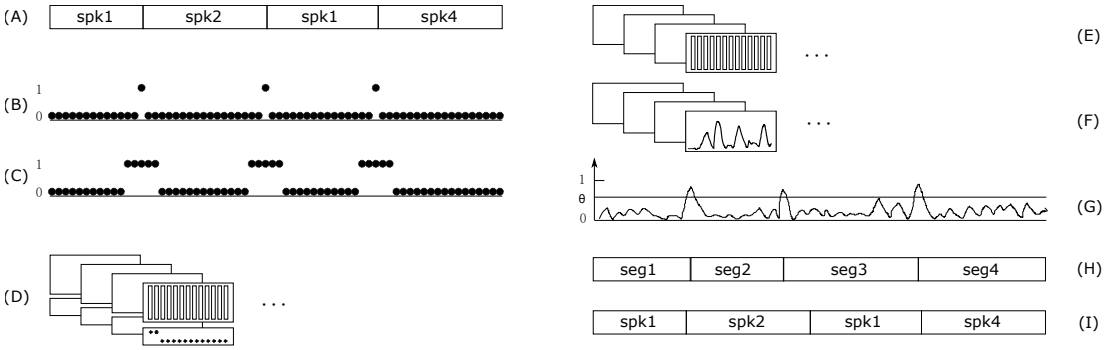

Figure 3.4: Training process (*left*) and prediction process (*right*) for speaker change detection.

Given an audio recording, speaker change detection aims at finding the boundaries between speech turns of different speakers. In Figure 3.4, the expected output of such a SCD system would be the list of timestamps between *spk1* & *spk2*, *spk2* & *spk1*, and *spk1* & *spk4*.

Similar to VAD, SCD can also be addressed using the same principle: $y_i = 1$

if there is a speaker change during the ith frame, $y_i = 0$ otherwise. Compared to VAD, the contextual information is more important for SCD task. It needs to capture change over time. It is virtually impossible to predict a change/not change based on a single frame. Traditional SCD approaches need two adjacent windows centered at the current frame. Then one should decide whether the two windows originate from the same speaker according to the statistic distance between them. Motivated by the success of Bi-LSTMs in VAD task, we adapt *Gelly*'s work to our SCD task and the process is depicted in Figure 3.4.

3.4.1 Class imbalance

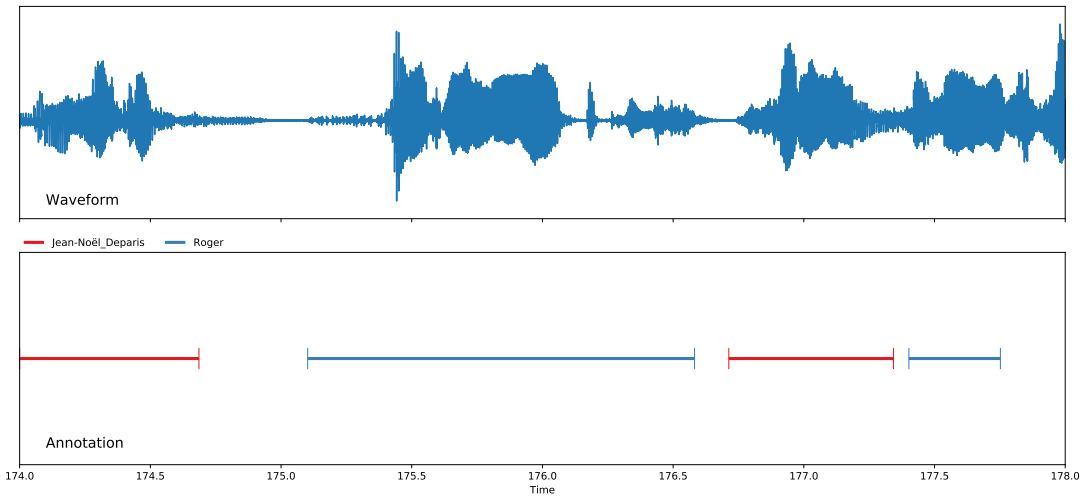

Figure 3.5: An example of annotation in ETAPE dataset.

Since there are relatively few change points in the audio files as shown in Figure 3.5, very little frames are in fact labeled as positive. For instance, in the ETAPE dataset which is used in the experimental section, this represents only 0.4% of all frames. This class imbalance issue could be problematic when training the neural network. Moreover, one cannot assume that human annotation is precise at the frame level. It is likely that the actual location of speech turn boundaries is a few frames away from the one selected by the human annotators.

This observation led most speaker diarization evaluation benchmarks [91; 92; 93] to remove from evaluation a short collar (up to half a second) around each manually annotated boundary. Therefore, as depicted in part C of Figure 3.4 and Figure 3.6, the number of positive labels is increased artificially by labeling as positive every frame in the direct neighborhood of the manually annotated change point. We will further evaluate the impact of the size of this neighborhood in Section 3.4.5.

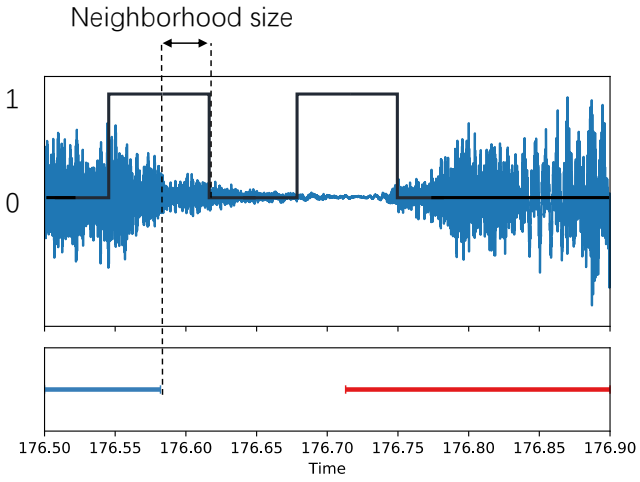

Figure 3.6: Zoom on the change point part. Frames in the direct neighborhood of the manually annotated change points are also labeled as positive.

3.4.2 Prediction

As shown in Figure 3.4, SCD shares its training and prediction processes with VAD. The long audio sequences are split into short fixed-length overlapping sequences, and the final sequence of scores is the average of several overlapping sequences of scores. However, the post-processing step proposed in VAD cannot be applied for SCD. While the speech or non-speech parts always consist of several consecutive frames, a change point is a single frame.

The segment duration distribution in ETAPE dataset is shown in Figure 3.7. From the distribution, we can find most segments are longer than 1s. In other

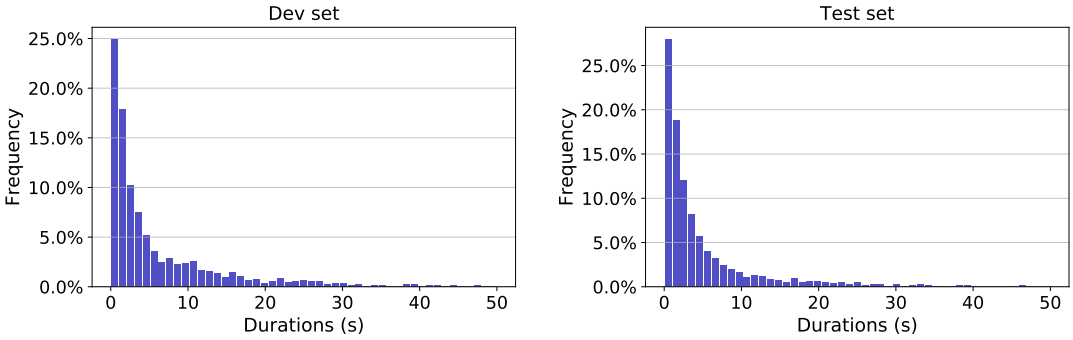

Figure 3.7: Segment duration distribution in ETAPE dataset.

words, the distance between adjacent change point frames is always longer than 1s. That leads us to use a similar post-processing step in conventional SCD approaches: all local maxima on a sliding window of duration δ_{peak} exceeding a threshold θ_{peak} are marked as speaker change points, as shown in part G of Figure 3.4, where δ_{peak} is used to prevent speech segments shorter than δ_{peak}. Parts H and I respectively represent the hypothesized and ground truth speaker change points.

3.4.3 Implementation details

Network architecture. The model for SCD is composed of two Bi-LSTM layers and two fully connected layers. *Bi-LSTM1* has 128 outputs (64 forward and 64 backward). *Bi-LSTM2* has 64 (32 each). Both fully connected layers are 32-dimensional. The output layer is 1-dimensional with a sigmoid function just like VAD task.

Class imbalance. A positive neighborhood of 100ms (50ms on both sides) is used around each change point, to partially solve the class imbalance problem.

Baseline. Both BIC [62] and Gaussian divergence [61] baselines rely on the same set of features (without derivatives, because it leads to better performance), using two 2s adjacent windows. We also report the result obtained by the *TristouNet* approach, that used the very same experimental protocol [48].

3.4.4 Experimental results

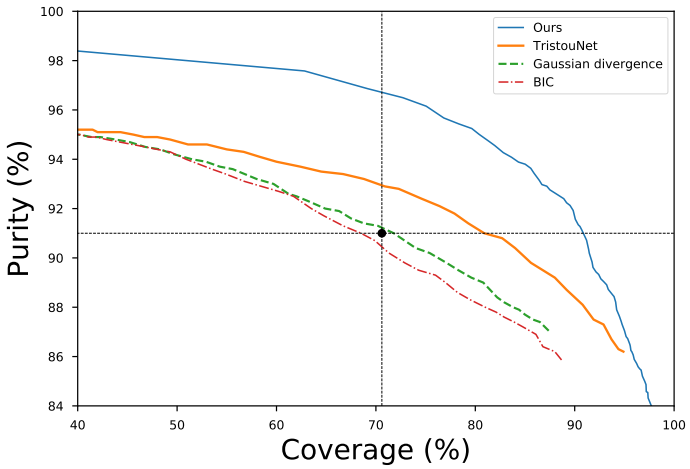

Figure 3.8: Speaker change detection on ETAPE development set.

All tested approaches (including the one we propose) rely on a peak detection step (keeping only those whose value is higher than a given threshold θ_{peak}). Curves in Figure 3.8 were obtained by varying the value of this threshold θ_{peak}. Our proposed solution outperforms BIC-, divergence-, and *TristouNet*-based approaches, whatever the operating point. Notice how it reaches a maximum purity of 98%, while all others are stuck at 95.1%. This is explained by the structural limitations of approaches based on the comparison of two adjacent windows: it is not possible for them to detect two changes if they belong to the same window. Our proposed approach is not affected by this issue.

Figure 3.9 summarizes the same set of experiments in a different way, showing purity at 70.6% coverage, and coverage at 91.0% purity. Those two values are marked by the horizontal and vertical lines in Figure 3.8 and were selected because they correspond to the operating point of the divergence-based segmentation module of our in-house multi-stage speaker diarization system [18]. Our approach improves both purity and coverage. For instance, in comparison to Gaussian divergence, it produces speech turns that are 28.8% longer on average, with the

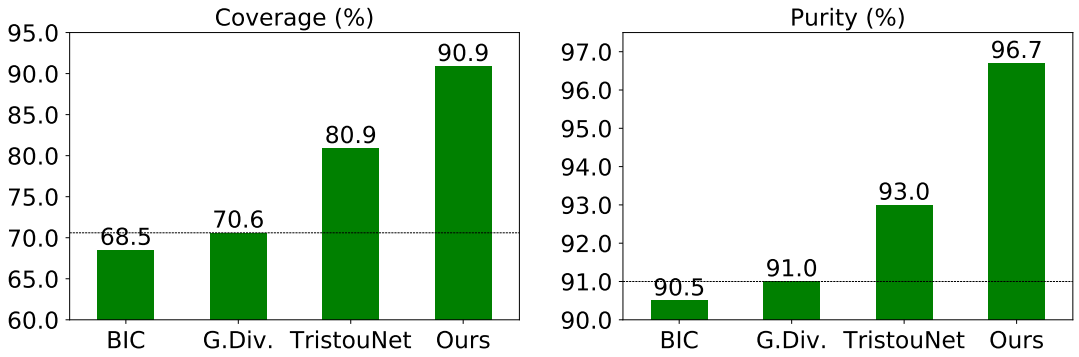

Figure 3.9: *Left*: coverage at 91.0% purity. *Right*: purity at 70.6% coverage.

same level of purity.

3.4.5 Discussion

3.4.5.1 Do we need to detect all speaker change points?

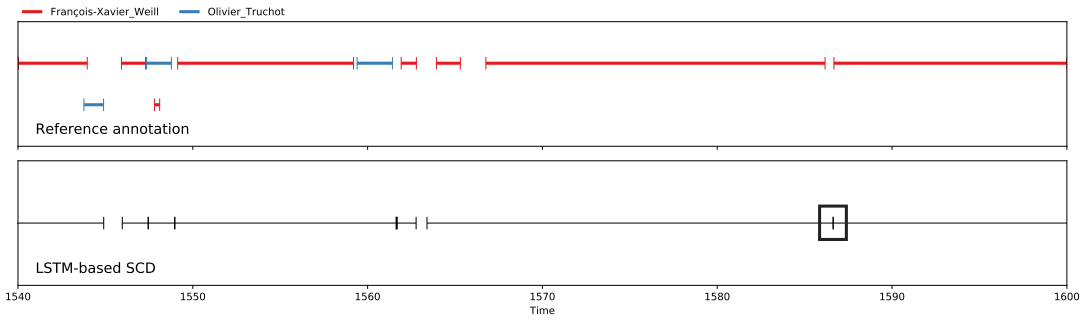

Figure 3.10: An example output of our SCD systems (bottom). The top is the reference annotation. The detected change point in the black rectangle corresponds to a short non-speech segment in the reference annotation.

In our training process, speech/non-speech changes are considered the same as speaker changes and our prediction relies on a peak detection step, where the short non-speech duration may converge to a single change point. However, as shown in Figure 3.10, the VAD system did not detect the non-speech around the rectangle, and the detected change point is not a real speaker change point, because the

speakers centered by this point are the same. Preliminary experiments tend to show that we should not consider those as change points. VAD will take care of that.

3.4.5.2 Fixing class imbalance

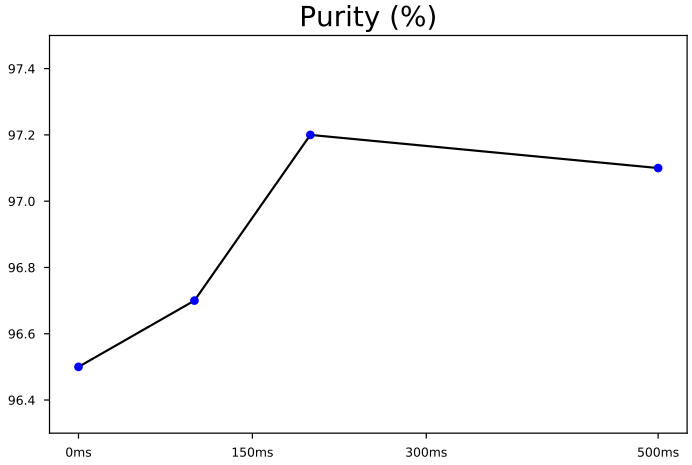

Figure 3.11: Purity at 70.6% coverage for different balancing neighborhood size.

As discussed in Section 3.4.1, to deal with the class imbalance problem, we artificially increased the number of positive labels during training by labeling as positive every frame in the direct neighborhood of each change point. Figure 3.11 illustrates the influence of the duration of this neighborhood on the segmentation purity, given that coverage is fixed at 70.6%. It shows a maximum value for a neighborhood of around 200ms. One should also notice that, even without any class balancing effort, the proposed approach is still able to reach 96.5% purity, outperforming the other three tested approaches: the class imbalance issue is not as problematic as we initially expected.

3.4.5.3 *"The Unreasonable Effectiveness of LSTMs"*

As *Karpathy* would put it[1], the proposed approach seems *unreasonably effective*. Even though LSTMs do rely on an internal memory, it is still surprising that they perform that well for speaker change detection, given that, at a particular time step i, all they see is the current feature vector $\boldsymbol{x}_i$. We first thought that concatenating features from adjacent frames would be beneficial, but this did not bring any significant improvement. The internal memory mechanism is powerful enough to collect and keep track of contextual information.

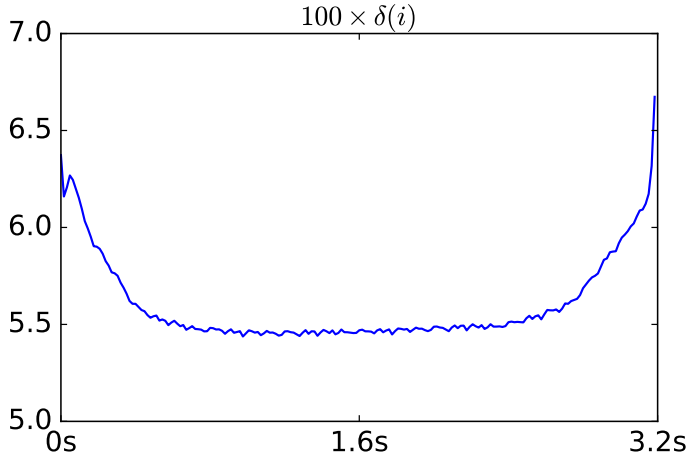

Figure 3.12: Expected absolute difference between prediction score and reference label, as a function of the position in the 3.2s subsequence.

This is further highlighted in Figure 3.12 that plots the expected absolute difference between predicted scores $f(\mathbf{x})_i \in [0, 1]$ and reference labels $\mathbf{y}_i \in \{0, 1\}$, as a function of the position i in the sequence: $\delta(i) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left(|f(\mathbf{x})_i - \mathbf{y}_i|\right)$. It clearly shows that the proposed approach performs better in the middle than at the beginning or the end of the sequence, quickly reaching a plateau as enough contextual information has been collected. This anticipated behavior justifies after the fact the use of strongly overlapping subsequences – making sure that each time step falls within the best performing region at least once.

[1] `karpathy.github.io/2015/05/21/rnn-effectiveness`

3.5 Re-segmentation

Given the output of the clustering step, re-segmentation aims at refining speech segments boundaries and labels. Similar to VAD and SCD, this task can also be addressed as a sequence labeling task. Assuming the output of the clustering step predicts k different speakers, we can use the same principle with $K = k + 1$ classes: $y_i = 0$ for non-speech and $y_i = k$ for speaker k.

Re-segmentation is usually achieved with a combination of GMMs cluster modeling and Viterbi decoding, as described in Chapter 2. We propose to use the same approach as VAD and SCD. The only difference is the loss function, which is changed to categorical cross entropy, and the activation function of the output layer is replaced by softmax.

Re-segmentation step is usually applied independently to each file. Similar to VAD and SCD, audio files are processed using overlapping sliding windows to generate subsequences. At training time, the (unsupervised) output of the clustering step is used as its reference label sequence, which is then used to train the neural network for several epochs E.

At test time, the model at E^{th} epoch is applied on the very same test file it has been trained on. For each time step i, this results in several overlapping sequences of K-dimensional (softmax-ed) scores, are averaged to obtain the final score of each class. Then, the resulting sequence of K-dimensional scores is post-processed by choosing the class with the maximum score for each frame.

Even though the training and testing are applied independently for each file, the hyper-parameter E is tuned globally. If E is small, the model may not be powerful enough to make a prediction (underfitting), and if the E is large, the prediction may converge to the clustering result.

Suitable E may vary in different files. To stabilize the choice of this hyper-parameter E and make the prediction scores smoother, scores from the $m = 3$ previous epochs are averaged when doing predictions at epoch E.

While this re-segmentation step does improve the labeling of speech regions, it

	DER	FA	Miss	Confusion	Purity(%)	Coverage
Before re-segm.	28.84	5.11	6.91	16.81	78.49	82.63
After re-segm.	27.50	4.81	7.22	15.46	80.01	83.89

Table 3.2: Effect of re-segmentation (%).

also has the side effect of increasing false alarms (*i.e.* non-speech regions classified as speech). Therefore, its output is further post-processed to revert speech/non-speech regions back to the original VAD output.

3.5.1 Implementation details

Network architecture. The model is composed of two Bi-LSTM layers and one fully connected layer. *Bi-LSTM1* has 128 outputs (64 forward and 64 backward). *Bi-LSTM2* has 64 (32 each). The fully connected layer is 32-dimensional. The output layer is K-dimensional with a softmax function.

Diarization system. The diarization system is based on neural VAD, SCD introduced in this chapter, affinity propagation clustering which will be introduced in Chapter 4, and hyper-parameters joint optimization which will be introduced in Chapter 5.

3.5.2 Results

Table 3.2 shows the effect of the proposed re-segmentation step on the output of affinity propagation clustering: it improves both cluster purity and coverage, leading to an absolute decrease of 1.34% in diarization error rate. A detailed file-wise analysis shows that this re-segmentation step consistently improves performance on every file.

Figure 3.13 is meant to analyze the behavior of the approach and to evaluate the robustness of its unique hyper-parameter E. The horizontal dashed line is the DER of the system before re-segmentation (*i.e.* the output of the clustering step). DER quickly decreases during the first few epochs, reaches an improved minimum

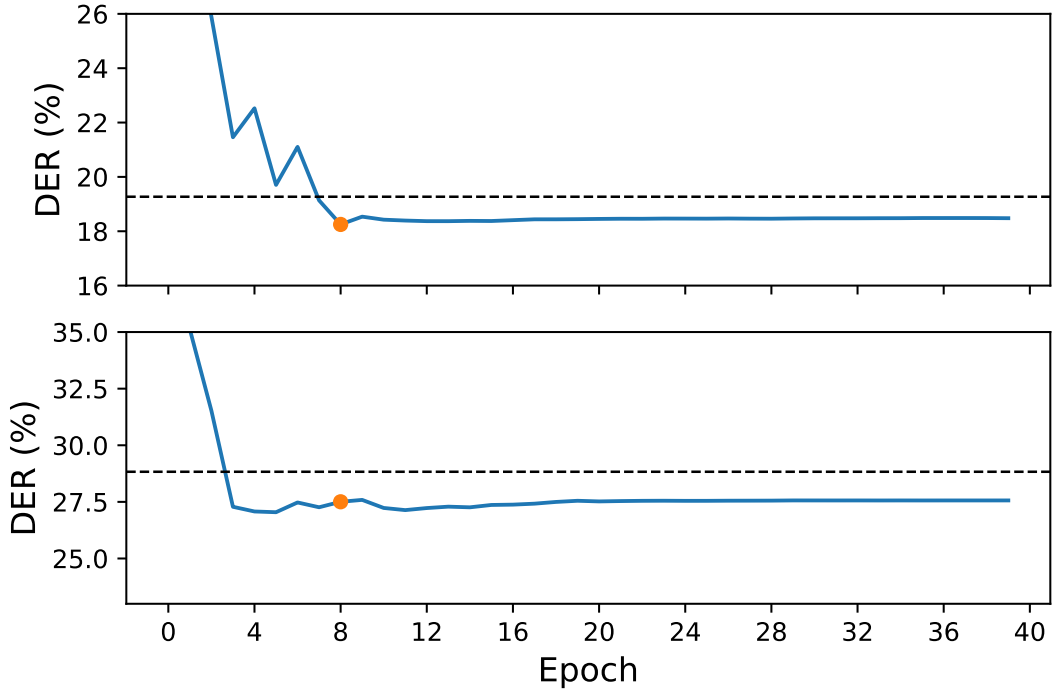

Figure 3.13: Re-segmentation on development (*top*) and test sets (*bottom*). The best epoch on the development set is marked with an orange dot.

value, then starts to over-fit and converges to a DER that is always better than original DER. This observation, combined with the fact that the optimal number of epochs on the test set is close to the one selected on the development set, leads us to the conclusion that the proposed LSTM-based re-segmentation is stable and very unlikely to degrade performance.

Figure 3.14 shows how the proposed re-segmentation system improves the diarization result. Usually, the errors made in the SCD step will be passed to the clustering step. In this example, the first speaker change point is not detected, and the first segment is grouped with a wrong cluster (the clustering algorithms for speaker diarization systems always make a lot of errors in overlap part). Our proposed re-segmentation can make some corrections around boundaries, as shown in the rectangle part in Figure 3.14.

In the framework of DIHARD II speaker diarization challenge [94], we also

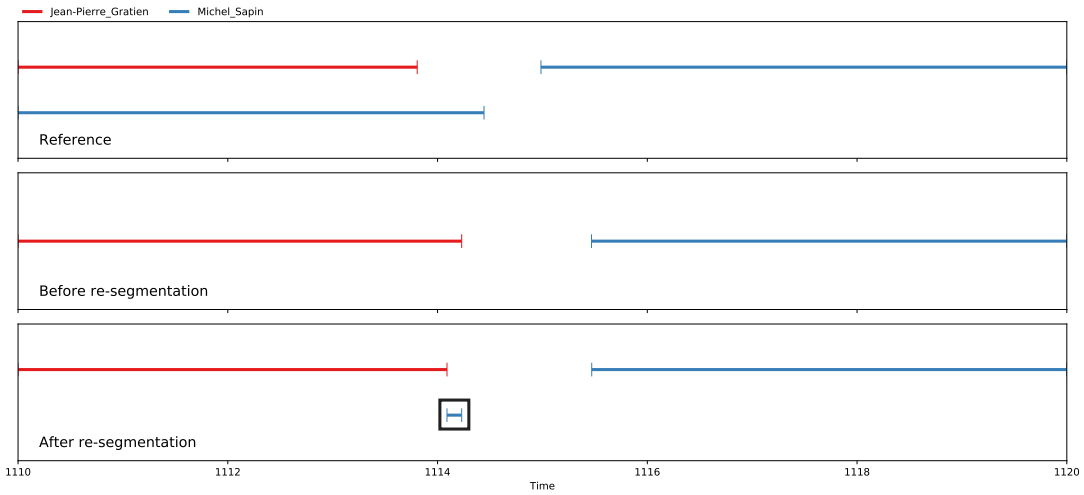

Figure 3.14: An example of re-segmentation result. *Top*: Reference annotation. *Middle*: Hypothesis annotation before the re-segmentation. *Bottom*: Hypothesis annotation after the re-segmentation. An optimal mapping has been applied to both hypothesis annotations. The correction made by the re-segmentation step is in the rectangle part.

successfully applied this re-segmentation technique, improving the provided baseline by 1.2% DER (32.6% vs 31.4%).

3.6 Conclusion

In this chapter, we show that both the initial segmentation (voice activity detection and speaker change detection) and the final re-segmentation can be formulated as a set of sequence labeling problems, addressed using bidirectional Long Short-Term Memory (Bi-LSTM) networks.

For speaker change detection, the experimental results on the ETAPE dataset led to significant improvements over conventional methods (*e.g.,* based on Gaussian divergence) and recent state-of-the-art results based on *TristouNet* embeddings [48] also using LSTMs). While neural networks are often considered as *"magic"* black boxes, we tried in Section 3.4.5.3 to better understand why these approaches are so powerful, despite their apparent simplicity.

For the re-segmentation step, it also shows an improvement in diarization results. However, finding the best epoch E relies on a development set. We plan to investigate a way to automatically select the best epoch for each file.

Preliminary experiments show that this family of approaches can also be used for overlapped speech detection ($y = 1$ for overlap, $y = 0$ otherwise)

We did try to integrate our improved speaker change detection into our in-house speaker diarization system. Unfortunately, the overall impact on the complete system in terms of diarization error rate is very limited. This may be because the subsequent clustering module was optimized jointly with the divergence-based segmentation step, expecting a normal distribution of features in each segment – which has no reason to be true for the ones obtained through the use of LSTMs. That leads us to Chapter 4, where we will integrate neural-based segmentation with neural speaker embedding.

Chapter 4

Clustering Speaker Embeddings

4.1 Introduction

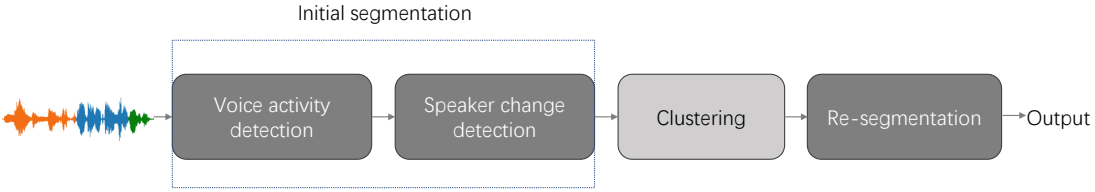

Figure 4.1: Diarization pipeline. In this chapter, we propose to rely on neural networks for some sub-steps of clustering.

As we proposed in Chapter 3, all modules in the diarization system are addressed with neural approaches except the clustering. However, even though VAD and SCD achieve excellent performance with LSTM, the integration with conventional HAC shows little impact on the final result. That may be because the clustering algorithm still relies on statistical similarity metrics such as BIC and CLR. Motivated by the successful application of i-vector and d-vector in speaker verification tasks, as shown in Figure 4.2, clustering in recent diarization systems is split into three steps: speech turn embedding, similarity matrix measurement, and actual clustering. Speech turn embedding aims at extracting high-level speaker representation vectors from audio segments by a speaker

embedding system. Then the similarity between two audio segments can be measured by the PLDA score or the other similarity metrics. In Section 4.2, we do a brief introduction of speaker embedding systems and show how to combine the segmentation results with the trained speaker embedding systems to compute the similarity matrix for clustering.

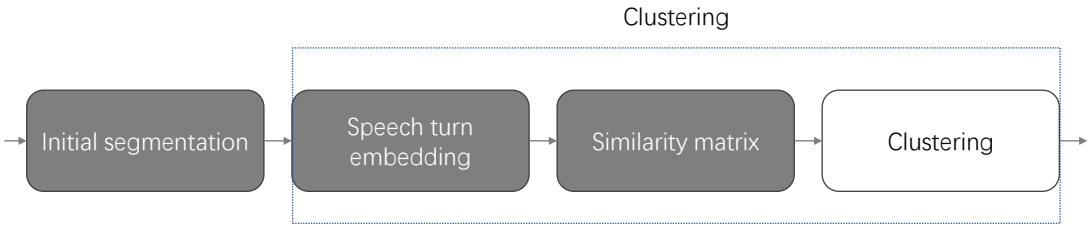

Figure 4.2: Clustering of the diarization pipeline. We propose to rely on neural networks for speech turn embedding and similarity matrix measurement.

Hierarchical agglomerative clustering is the most used clustering method in conventional diarization systems. In Section 4.3, we compare the hierarchical agglomerative clustering with another clustering algorithm: affinity propagation. Both of them are applied on top of a neural speaker embedding system introduced in [48; 95]. An affinity propagation variant has been introduced in [96] for speaker diarization, but it is supervised by the number of speakers and relies on standard statistical models to compute speaker similarities.

For similarity measurement, in most existing clustering algorithms, the similarity between any two segments is measured independently, and the sequential information is ignored. However, conversations between several speakers are usually highly structured, and turn-taking behaviors are not randomly distributed over time. In [95], structured prediction is applied for online speaker diarization, but only the structural information from the forward direction is considered. In Section 4.4, we propose to generate a more precise similarity matrix with a stacked bidirectional LSTMs and employ spectral clustering [16] to generate the final results. This work [5] was performed in collaboration with *Qingjian Lin* who did most of the experiments.

As shown in Figure 4.1 and Figure 4.2, at the end of this chapter, the clustering stage will be partly (speech turn embedding and similarity matrix measurement) based on neural networks.

4.2 Speaker embedding

As shown in Figure 4.2, the clustering module is split into three sub-steps. First, an embedding system $f : \mathcal{X} \to \mathbb{R}^D$ is trained to embed speech sequences $\mathbf{x}$ into a D-dimensional space where the segments from different speakers should be separable. Next, the pairwise similarity matrix is obtained by a similarity metric like cosine distance and PLDA. Finally, an actual clustering method is applied on top of the similarity matrix to generate the outputs.

4.2.1 Speaker embedding systems

There are three most used speaker embedding systems:

i-vector [39] is obtained by a dimensionality reduction process of the GMM supervector using joint factor analysis, where the GMM is speaker-specific and trained on MFCC features:

$$\boldsymbol{s} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{w} \tag{4.1}$$

where $\boldsymbol{s}$ is the speaker supervector, $\boldsymbol{m}$ is a speaker-independent supervector from UBM, $\boldsymbol{T}$ is the total variability matrix, and $\boldsymbol{w}$ is the i-vector. $\boldsymbol{m}$ and $\boldsymbol{T}$ should be trained with a large speaker dataset, if $\boldsymbol{T}$ is given, the i-vector can be computed from a speech segment.

d-vector is obtained by deep neural networks (DNNs). The input feature can be MFCC, Fbank and spectrogram *etc.* Here we categorize d-vector into two types. For the first type, a supervised DNN is trained to classify different speakers over the frame level features of speech segments, where the speakers are fixed in a given list. The d-vector is the output of bottleneck or the penultimate

layer. For this type, PLDA, together with a normalization method, is usually employed to measure the similarity between two d-vectors. For the second type, the DNN is used to embed a speech segment directly to a high-level embedding space. The first type of systems always uses the cross-entropy loss to encourage the separability of d-vector from different speakers. To make d-vector not only separable but also discriminative, the second type of d-vector usually involves some discriminative loss functions such as the contrastive loss and the triplet loss [45; 46]. The similarity between two d-vectors can be directly computed by cosine metric or Euclidean metric.

x-vector [14] is a specific case of the first type d-vector which is proposed by the Johns Hopkins University. The input feature is MFCC, and the neural network architecture is a time-delay neural network (TDNN) including a time-pooling layer to transform multiple frame-level features into a single vector which will be then passed to the fully connected layers. x-vector is the output of the penultimate layer.

4.2.2 Embeddings for fixed-length segments

Most speaker diarization systems rely on a uniform segmentation where speaker embeddings are extracted from a sliding window of fixed duration. This may lead to segments that contain more than one speaker. Since recent speaker embedding systems are trained with a large speaker dataset, and some data augmentation techniques are performed, systems are still able to extract the representation vector of the dominant speaker in the segments. In addition, when one evaluates the diarization results, it is common not to evaluate short collars centered on each speech turn boundary and exclude the overlap part. Therefore, uniform segmentation is widely used in recent speaker diarization systems. With fixed-length segments, the pretrained speaker embedding system can be applied directly to map them into a fixed-dimensional feature space.

4.2.3 Embedding system with speaker change detection

The initial segmentation system introduced in Chapter 3 aims at splitting the audio into speaker-homogeneous segments. Different from a uniform segmentation, the resulting segments have different lengths. The embedding systems trained with fixed-length speech segments cannot be applied directly.

An alternative solution is training a speaker embedding system with variable length utterances. The i-vector and most neural network architectures such as RNN and CNN support variable length inputs. In [49], *Zhang et al.* proposed to replace the final max/average pooling layer with a Spatial Pyramid Pooling layer in the Inception-Resnet-v1 architecture to train d-vector with the arbitrary size of the input. In [76], a d-vector model is trained by using variable-length windows to sample training examples. The window size is drawn from a uniform distribution within [240ms, 1600ms] during training. However, as shown in Figure 3.7, the duration of some segments is longer than 10 seconds, and long input sequences may cause a high computational cost and complexity.

For the second type of d-vector, our proposed solution is shown in Figure 4.3. It depicts how an embedding system – initially meant to process fixed-length (a few seconds, typically) speech segments – can be used to embed variable-length speech segments coming from the initial segmentation step (A). The idea is to slide a fixed-size window (B) over the duration of the file, embed each of these subsequences (C), and then average the embedding of all overlapping subsequences to obtain one embedding per initial segment (D).

4.2.4 Embedding system for experiments

The network architecture used for our experiments is introduced in [48] and further improved in [97]. Briefly, an LSTM-based neural network is trained to embed speech sequences $\mathbf{x}$ into a D-dimensional space, using the triplet loss paradigm. In the embedding space, two sequences $\mathbf{x}_i$ and $\mathbf{x}_j$ of the same speaker (resp.

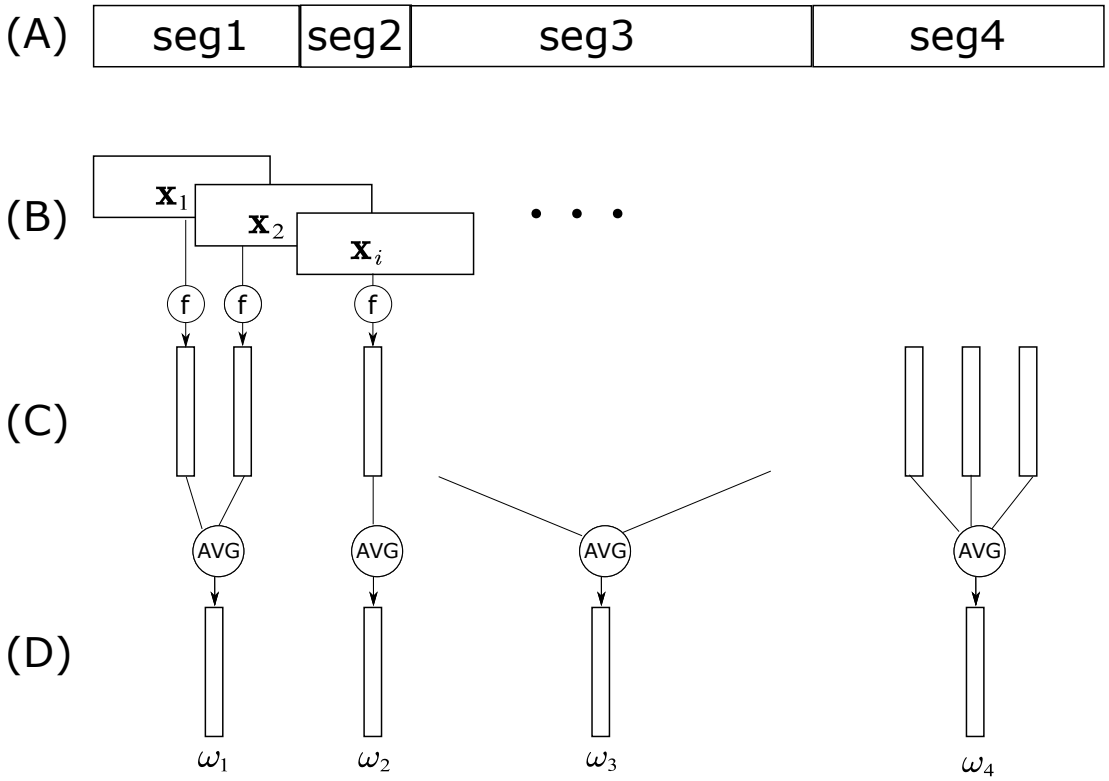

Figure 4.3: Aggregation of fixed-length subsequence embeddings.

two different speakers) are expected to be close to (resp. far from) each other according to their cosine distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{f(\mathbf{x}_i) \cdot f(\mathbf{x}_j)}{|f(\mathbf{x}_i)| \cdot |f(\mathbf{x}_j|)} \tag{4.2}$$

Two data augmentation strategies are applied in this embedding system. Noise from MUSAN dataset [98] is added to the audio during training. Similar to the embedding system proposed in [76], in training process, the length of the input speech segments is sampled from a uniform distribution within [500ms, 1500ms]. Even though our speaker embedding system can embed speech segments of variable lengths, we use the second aggregation strategy because speech turn may be longer than 1500ms. The sliding window is fixed to 1s in our experiments. As shown in Figure 4.3, the embedding of segment i is denoted as ω_i in the next

section.

4.3 Clustering by affinity propagation

Hierarchical agglomerative clustering is the most used clustering method in speaker diarization systems. Even though hierarchical agglomerative clustering is easy to understand and implement, its weaknesses are obvious:

1. It cannot pull back the previous decision. Once an example has been assigned to a wrong cluster, it cannot be moved out. And it will affect the next decision.

2. As introduced in Section 2.5.1.1, it relies on linkage criteria to compute the distance between two clusters. For single and complete linkage, only a single pair of examples from two clusters will be considered for distance computation, ignoring the global information.

3. It is very sensitive to outliers. In complete linkage, a single data point far from the center can increase the distance to other clusters dramatically and completely change the final clustering. An example is shown in Figure 4.4. The four data points $\{d_2, d_3, d_4, d_5\}$ are split because of the outlier d_1 at the left edge and it does not find the most intuitive cluster structure in this example [4].

The affinity Propagation (AP) algorithm [15] does not require a prior choice of the number of clusters contrary to other popular clustering methods. All speech segments are potential cluster centers (exemplars). Taking as input the pairwise similarities between all pairs of speech segments, AP will select the exemplars and associate all other speech segments to an exemplar. In our case, the similarity between i^{th} and j^{th} speech segments is the negative cosine distance between their embeddings: $s(i, j) = -d(\omega_i, \omega_j)$

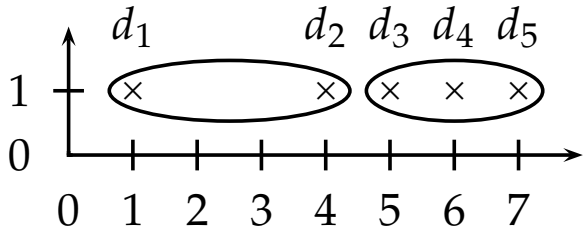

Figure 4.4: Outliers in complete-link clustering. The five data points have the x-coordinates $1 + 2\epsilon, 4, 5 + 2\epsilon, 6$ and $7 - \epsilon$. Complete-link clustering creates the two clusters shown as ellipses. The most intuitive two-clusters clustering is $\{\{d_1\}, \{d_2, d_3, d_4, d_5\}\}$, but in complete-link clustering, the outlier d_1 splits $\{d_2, d_3, d_4, d_5\}$. Figure taken from [4].

On the diagonal of the similarity matrix, $s(k, k)$ is set to the preference value θ_{AP}, a hyper parameter which influences the choice of ω_k as exemplar and thus the final number of clusters. AP clustering can be viewed as a "message passing" process between speech segments with two kinds of message: responsibility and availability. Responsibility $r(i, k)$ is a message sent from segment i to k that quantifies how well-suited x_k is to serve as the exemplar for x_i. Availability $a(i, k)$ is a message sent from segment k to i that represents how appropriate it would be for segment i to pick segment k as its exemplar.

4.3.1 Implementation details

Dataset. The REPERE corpus is used for training the neural networks used in VAD, SCD, and embeddings. The ETAPE TV development subset is used for hyper-parameter tuning.

Feature extraction. Each part of the diarization pipeline shares the same set of input features extracted every 10ms on a 25ms window using Yaafe toolkit [88]: 19 mel-frequency cepstral coefficients (MFCC), their first and second derivatives, and the first and second derivatives of the energy (amounting to a total of 59 dimensions).

Initial segmentation. The experiments use the same initial segmentation re-

sults reported in Chapter 3. Both VAD and SCD are tuned independently according to the detection error rate (VAD) and segmentation coverage and purity (SCD).

Sequence embedding. Implementation details are identical to the ones used in [95]. It is trained on REPERE dataset and 192-dimensional embeddings are extracted every 0.4s on sub-sequences of duration 1s.

4.3.2 Results and discussions

	DER	FA	Miss	Confusion	Purity	Coverage
AP	31.28	3.95	6.97	20.36	77.54	76.48
HAC	35.99	3.95	6.97	25.06	75.14	75.29

Table 4.1: Performance on ETAPE TV test set of hierarchical agglomerative clustering and affinity propagation (AP).

Table 4.1 summarizes the results of two clustering methods. Affinity propagation shows a much better performance than hierarchical agglomerative clustering with complete-link on the ETAPE TV dataset according to DER (31.28% vs. 35.99%). Both purity and coverage are improved when we switch from hierarchical agglomerative clustering to affinity propagation. A detailed file-wise analysis shows that affinity propagation consistently outperforms the hierarchical agglomerative clustering on every file. For hierarchical agglomerative clustering, other linkages were also tested (average, pool) but found to lead to worse performance.

4.3.3 Discussions

An example of clustering results of affinity propagation and hierarchical agglomerative clustering from ETAPE dataset is shown in Figure 4.5. Segment embedding vectors are converted to 2 dimensional by t-SNE [99]. Different colors represent different speakers, and the point size corresponds to the segment duration. From

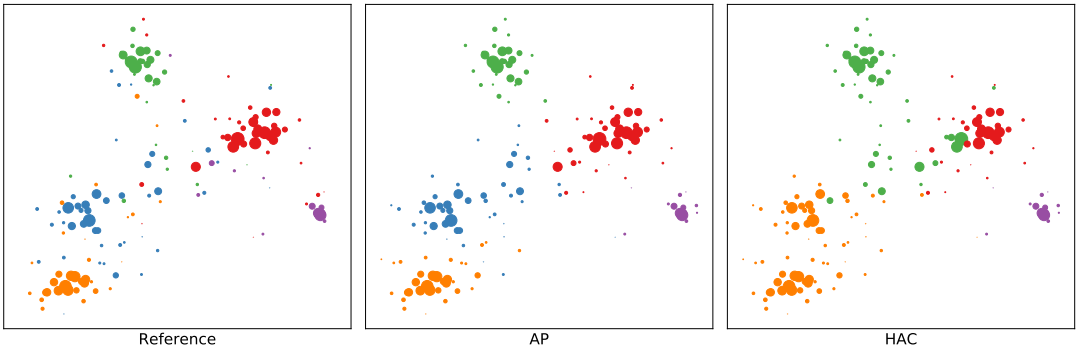

Figure 4.5: Clustering results of affinity propagation and hierarchical agglomerative clustering on an example from ETAPE dataset. The embeddings are converted to 2 dimensional by t-SNE. Each color represents the corresponding speaker in Figure 4.6 and the point size corresponds to the segment duration.

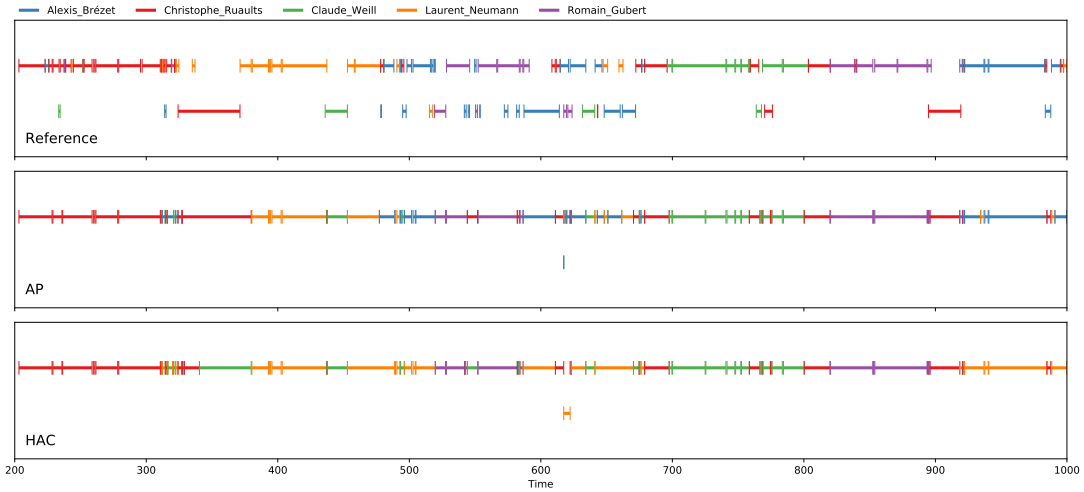

Figure 4.6: Diarization results of affinity propagation and hierarchical agglomerative clustering on an example from ETAPE dataset.

Figure 4.5, we can find that almost all the long speech segments are grouped correctly in the result of affinity propagation, while in hierarchical agglomerative clustering, the number of clusters is not correctly detected and a number of long segments are assigned to the wrong clusters. We can also find that in this example, the main source of clustering error is from short segments in both approaches. That may be because it is difficult for our speaker embedding system to extract

speaker information from very short speech segments. In addition, as shown in Figure 4.6, in some short speech segments, there is more than one speaker speaking. However, our speaker embedding system is trained with pure segments and may be confused in overlapped speech segments. Therefore, traditional clustering methods such as affinity propagation and hierarchical agglomerative clustering cannot handle these short segments directly, and that leads us to use sequential information to improve the similarity matrix for clustering in Section 4.4.

4.4 Improved similarity matrix

Most existing clustering methods including hierarchical agglomerative clustering and spectral clustering, are based on a similarity matrix which is computed between each pair of segment embeddings independently. The similarity metric could be the cosine distance (for d-vector) or PLDA (for i-vector or x-vector). However, the sequential information is always ignored during the computation. In this section, we show how to improve the similarity matrix with sequential information.

Because we focus on the clustering step, we choose to use oracle VAD in this section, followed by uniform segmentation. In the clustering step, we use i-vector and x-vector as our embedding system, and Bi-LSTM is proposed to model the similarity matrix $\boldsymbol{S}$. Finally, spectral clustering is applied on top of the improved similarity matrix.

4.4.1 Bi-LSTM similarity measurement

Let $\mathbf{x} \in \mathcal{X}$ denote a sequence of speaker embedding vectors (*e.g.* i-vector, x-vector) extracted from a set of speech segments: $\mathbf{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2 ... \boldsymbol{x}_n)$, where n is the total number of segments for this audio file. Let $\boldsymbol{S}$ be a similarity matrix, where $\boldsymbol{S}_{i,j}$ is the similarity between segments i and j. The objective is to find a function $f : \mathcal{X} \rightarrow \boldsymbol{S}$ that maps the entire speaker embedding sequence into a

similarity matrix.

A similarity matrix is robust against speaker index changes or flipping. Therefore, we utilize $\boldsymbol{S}$ as the label of the entire speaker embedding sequence $\boldsymbol{x}$ for supervised diarization learning.

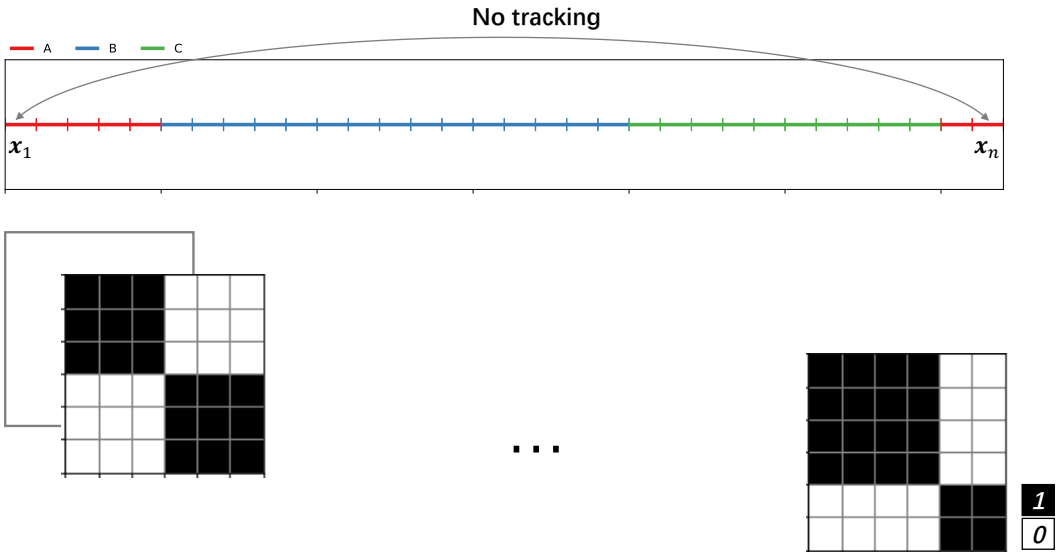

Figure 4.7: Processing the entire n segments with a sliding window. The similarity between segment $\boldsymbol{x}_1$ and the segment $\boldsymbol{x}_n$ cannot be directly measured due to the limited window size.

Since the number of segments n may be huge and vary between files, it is difficult to train such a function f directly. If we process the entire n segments in an m-segment ($m < n$) sliding window manner, the size of input and label vectors is fixed, which could make the training stage easier. However, such a system eventually generates a diagonal block similarity matrix. Since part of information in the matrix is lost, it easily fails to track different speakers among different windows. An example is shown in Figure 4.7. The similarity between segment $\boldsymbol{x}_1$ and the segment $\boldsymbol{x}_n$ cannot be directly measured due to the limited window size. Therefore, the system does not know that $\boldsymbol{x}_1$ and $\boldsymbol{x}_n$ are from the same speaker A.

In the proposed approach, we address this problem as a row by row sequence

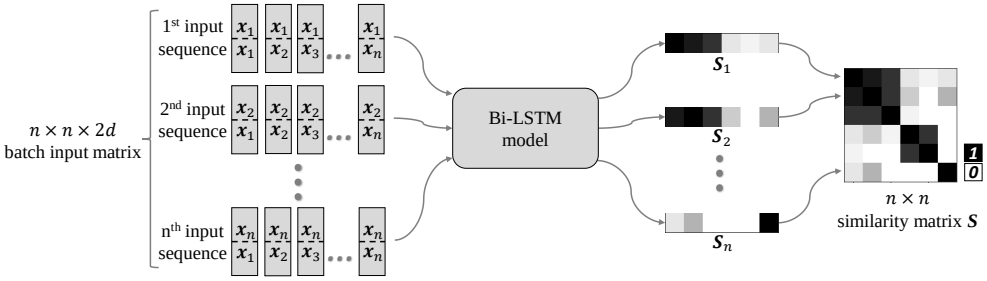

Figure 4.8: Bi-LSTM similarity measurement for a similarity matrix. Figure taken from [5].

labeling task such that $\boldsymbol{S}_{i,j} = 1$ if segment i and j are from the same speaker, and $\boldsymbol{S}_{i,j} = 0$ otherwise. The i^{th} row in the similarity matrix $\boldsymbol{S}_{i.}$ is calculated as follows:

$$\boldsymbol{S}_{i.} = f(\boldsymbol{x}_i, \mathbf{x}) \tag{4.3}$$

We propose to model the function f with a stacked Bi-LSTMs like VAD and SCD. As depicted in Fig. 4.8, for row i, the input at time j is the concatenation of $\boldsymbol{x}_i$ and current embedding vector $\boldsymbol{x}_j$. The similarity between segment i and segment j can be defined as follows:

$$\boldsymbol{S}_{i,j} = f_{\text{LSTM}}(\boldsymbol{x}_i, \mathbf{x})_j = f_{\text{LSTM}} \left(\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_i \end{bmatrix}, \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{x}_i \end{bmatrix}, \cdots, \begin{bmatrix} \boldsymbol{x}_n \\ \boldsymbol{x}_i \end{bmatrix} \right)_j \tag{4.4}$$

Once this Bi-LSTM model is trained, we apply this model on a speaker embedding sequence $\mathbf{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2 ... \boldsymbol{x}_n)$ n times, each time to perform inference for one row $\boldsymbol{S}_{i.}$ of the similarity matrix $\boldsymbol{S}$.

4.4.2 Implementation details

4.4.2.1 Initial segmentation

All experiments share the same initial segmentation step. Non-speech regions are first removed by an oracle VAD. Then, a sliding window is applied on speech

regions to generate segments. The sliding window is 1.5s long with a step size of 0.75s (50% overlapping). In training process, the corresponding speaker for each segment is the dominant speaker who occupies the most in the central 0.75s. The reference similarity matrix $\boldsymbol{S}_{Ref}$ consists only of 1 or 0, representing whether a pair of segments is from the same speaker or not.

4.4.2.2 Embedding systems

Two embedding systems are applied and compared in the proposed system: i-vector and x-vector. For i-vector, 20-dimensional MFCCs with delta and delta-delta coefficients are extracted to train a 2048-component GMM-UBM model. Supervectors of GMM is then projected into 128-dimensional i-vectors through the total variability matrix $\boldsymbol{T}$. The whole i-vector system is based on the kaldi/egs/callhome_diarization/v1 scripts [100; 101]. For x-vector, 23-dimensional MFCCs are extracted and followed by the sliding-window based cepstral mean normalization. Reverberation, noise, music, and babble noises are added to audio files for data augmentation. The whole x-vector system is based on the kaldi/egs/callhome_diarization/v2 scripts [101; 102].

4.4.2.3 Network architecture

Similar to the VAD and SCD, the architecture includes two Bi-LSTM layers followed by one fully connected layers (FC). Both Bi-LSTM layers have 512 outputs, 256 forward and 256 backward separately. The fully connected layer is 64-dimensional with the ReLU activation function. The output layer is 1-dimensional with a sigmoid activation function to output a similarity score between 0 and 1.

4.4.2.4 Spectral clustering

The similarity matrix obtained with the LSTM is post-processed by a normalization step proposed in [16] before spectral clustering:

1. Symmetrization: $\boldsymbol{Y}_{i,j} = max(\boldsymbol{S}_{i,j}, \boldsymbol{S}_{j,i})$
2. Diffusion: $\boldsymbol{Y} = \boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}$
3. Row-wise max normalization: $\boldsymbol{S}_{i,j} = \boldsymbol{Y}_{i,j}/max_k\,\boldsymbol{Y}_{i,k}$

In spectral clustering, the cluster number is selected by thresholding the eigenvalues of the normalized Laplacian matrix.

4.4.2.5 Baseline

The similarity matrix in baselines is measured by PLDA:

$$S_{i,j} = f_{\mathrm{PLDA}}(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{4.5}$$

As a hypothesis testing based method, PLDA generates both negative and positive scores, which is not supported in spectral clustering. We normalize PLDA scores by a logistic function:

$$g(x) = \frac{1}{1 + e^{-5x}} \tag{4.6}$$

4.4.2.6 Dataset

i-vectors and x-vectors are trained on a collection of SRE-databases including SRE 2004, 2005, 2006, 2008 and Switchboard. To compare with other systems, the CALLHOME dataset is used for evaluation. Similar to [76], a 5-fold validation is carried out on the dataset. The 500 utterances are split into 5 subsets uniformly and each time one subset is drawn as the evaluation dataset while the other four are used for training Bi-LSTM model. The reported diarization error rate is the average of the 5-fold evaluation results. In baseline, we also conduct the 5-fold validation where four training subsets are used for whitening PLDA including mean subtraction, full rank PCA mapping, and length normalization.

4.4.3 Evaluation metrics

Speaker diarization systems are usually evaluated through Diarization Error Rate (DER). In order to be comparable with other systems, the short collars centered on each speech turn boundary (0.25s on both sides) and overlapping speech are ignored. DER has three components: false alarm (FA), miss, and speaker confusion, among which FA and miss are mostly caused by VAD errors. Since an oracle VAD is employed in our implementation, we exclude FA and Miss from our evaluations. The DER referred here is the speaker confusion.

4.4.4 Training and testing process

In the training process, we reshape both the batch output and the ground truth similarity matrix into n^2 vectors and adopt the binary cross-entropy loss. Stochastic gradient descent optimizer is employed with a learning rate initialized at 0.01 and divided by 10 every 40 epochs. The whole model training process terminates after 100 epochs, and then the training outputs are used to tune thresholds for clustering systems. In the evaluation process, the learned thresholds are applied to the testing dataset, and the system is evaluated by DER.

4.4.5 Results

Description	Embedding	DER (%)
PLDA	i-vector	10.13
	x-vector	8.05
LSTM	i-vector	**8.53**
	x-vector	**7.73**
	Fusion	**6.63**
Recent works	Wang et al. [16]	12.0
	Sell at al. [78]	11.5
	Romero et al. [60]	9.9
	Zhang et al. [76](5-fold)	7.6

Table 4.2: DER (%) on CALLHOME dataset for different systems.

Table 4.2 summarizes the main experimental results. All systems share the same initial segmentation step and the spectral clustering method. The proposed pipeline reaches a better performance than PLDA baseline (8.53% vs. 10.13% for i-vector and 7.73% vs. 8.05% for x-vector). The proposed systems based on i-vector and x-vector are fused at the similarity matrix level. The fusion is performed by the weighted sum of their similarity matrices, and the resulting system outperforms all recent diarization systems on CALLHOME dataset.

4.4.6 Discussions

To analyze the behavior of the proposed system, we conduct Student's t-test on the results of PLDA and LSTM similarity measurement with the i-vector embedding system. The 500 utterances in CALLHOME dataset are first sorted by increasing duration and then split into five groups. In other words, the first group contains the 100 shortest utterances, while the last group contains the longest ones. Next, the t-test analysis is performed on each group independently.

The null (H_0) and alternative (H_1) hypotheses are:

$$H_0 : \mathrm{DER}_{\mathrm{plda}} = \mathrm{DER}_{\mathrm{lstm}}, \quad H_1 : \mathrm{DER}_{\mathrm{plda}} \neq \mathrm{DER}_{\mathrm{lstm}}$$

The p-value is set to 0.05 and thus accept H_0 if the t-value is in (-1.96, 1.96), otherwise, reject H_0. The results are shown in Table 4.3. H_0 is accepted in short utterance groups while rejected in long utterance groups with 95% confidence. In addition, $\mathrm{DER}_{\mathrm{LSTM}}$ are smaller than $\mathrm{DER}_{\mathrm{plda}}$ for long utterances. PLDA model ignores context information while Bi-LSTM model takes full advantage of sequential information from forward and backward sequences. LSTM outperforms PLDA in longer utterances because longer utterances may include more sequential information than short utterances.

sorted utterances	DER_{plda}	DER_{LSTM}	t-value	H_0
$1^{th} \sim 100^{th}$	6.6	5.5	-1.22	accepted
$101^{th} \sim 200^{th}$	5.7	5.3	-0.35	accepted
$201^{th} \sim 300^{th}$	6.1	3.9	-2.16	rejected
$301^{th} \sim 400^{th}$	9.2	7.5	-2.11	rejected
$401^{th} \sim 500^{th}$	13.9	11.6	-2.38	rejected

Table 4.3: T-test in five groups with sorted durations. Table taken from [5].

4.5 Conclusion

In this chapter, we split clustering into three sub-steps: speech turn embedding, similarity measurement, and clustering. We extract segment embedding vectors and then measure the similarity matrix. We also show that the affinity propagation outperforms the standard HAC with complete-link. In addition, we use Bi-LSTM to improve the similarity matrix with i-vector and x-vector embedding systems. The fusion system with spectral clustering achieves state-of-the-art performance with a 6.63% DER on CALLHOME dataset.

The proposed systems is a step towards an integrated end-to-end neural approach to speaker diarization. However, the proposed diarization systems still rely on a traditional clustering method, such as affinity propagation and spectral clustering. That leads us to Chapter 5, where we propose to formulate the clustering step as a supervised classification task that can be handled by neural approaches.

Chapter 5

End-to-End Sequential Clustering

5.1 Introduction

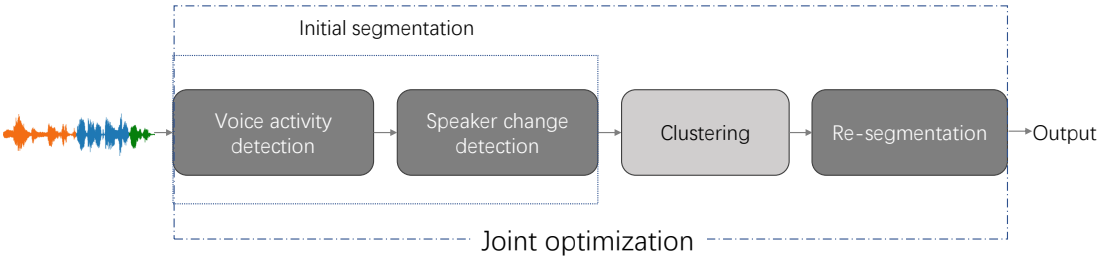

Figure 5.1: Diarization pipeline. We propose to jointly optimize the hyper-parameters of the whole diarization pipeline.

As depicted in Figure 5.1, we were able to replace most diarization steps by neural approaches and are getting closer to obtaining a fully end-to-end neural speaker diarization. In this chapter, to get even closer, we first propose to jointly optimize the hyper-parameters of the whole diarization pipeline. This is summarized in Section 5.2. The next step, described in Section 5.3, is to formulate the clustering step as a supervised classification task that can be handled by neural approaches. As shown in Figure 5.2, at the end of this chapter, all modules will be based on neural networks.

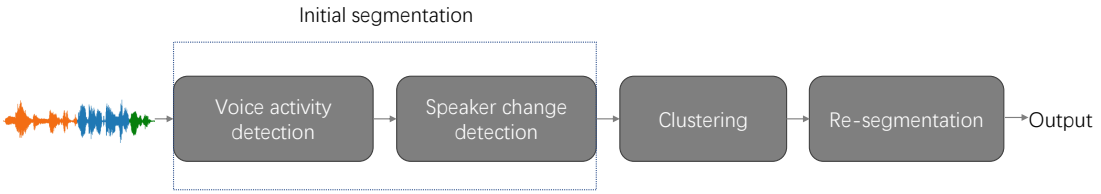

Figure 5.2: Diarization pipeline. In this chapter, we propose to rely on recurrent neural networks for all modules.

5.2 Hyper-parameters optimization

5.2.1 Hyper-parameters

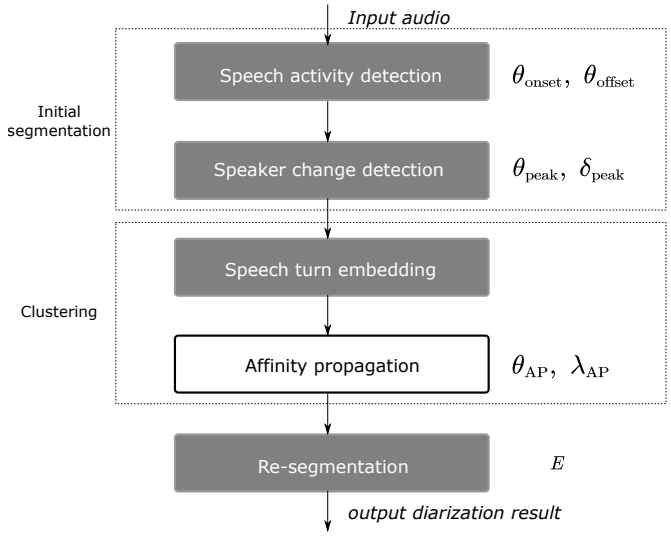

Figure 5.3: Diarization pipeline and hyper-parameters.

Our proposed speaker diarization pipeline consists of four consecutive modules: VAD, SCD, clustering, and re-segmentation. As shown in Figure 5.3, for VAD, the hyper-parameters include θ_{onset} and θ_{offset}, which are used to post-process the sequence of speech scores for the detection of the start and end time of speech regions. For SCD, the hyper-parameters include δ_{peak} and θ_{peak}. In the sequence of scores, all local maxima on a sliding window of duration δ_{peak} exceeding a threshold θ_{peak} are marked as speaker change points. Since we use affinity

propagation in the clustering step, the hyper-parameters include preference value θ_{AP} and damping factor λ_{AP}. For re-segmentation, the hyper-parameter is E, which is the number of epochs in the self-training step.

5.2.2 Separate vs. joint optimization

Each step of the pipeline introduced in Chapter 3 and Chapter 4 can be optimized separately on ETAPE TV development set. VAD hyper-parameters are optimized to minimize the detection error rate introduced in Section 2.8.1. For SCD, the system is evaluated with dual metrics purity and coverage (introduced in Section 2.8.2) that can be combined into a single F1 score. However, since errors made in the initial segmentation step cannot be corrected in the clustering step, high purity is more important than high coverage. Therefore, we tune SCD hyper-parameters to maximize coverage under the constraint the purity has to be at least 94%. For clustering and re-segmentation, hyper-parameters are tuned to minimize the diarization error rate.

In joint optimization, all the hyper-parameters in our proposed diarization pipeline are jointly optimized. More precisely, we use the Tree-structured Parzen Estimator hyper-parameter optimization approach [103] available in *scikit-optimize* [89] to automatically select the set of hyper-parameters that minimizes diarization error rate. Note that hyper-parameter E for re-segmentation is tuned separately, but it should ideally be optimized with the rest of the pipeline.

5.2.3 Results

Table 5.1 summarizes the results of jointly and separately optimized diarization pipelines. It shows that the jointly optimized pipeline performs better according to the diarization error rate (28.84% vs. 31.28%) where the confusion is decreased 3.55% at the expense of the increase (1.16%) of false alarm rate.

	DER	FA	Miss	Confusion	Purity	Coverage
Separate optimization	31.28	3.95	6.97	20.36	77.54	76.48
Joint optimization	28.84	5.11	6.91	16.81	78.49	82.63
Joint optimization (VAD)	27.84	3.82	7.30	16.71	78.49	82.63

Table 5.1: Performance of different diarization pipelines. The evaluation metrics include diarization error rate (DER), false alarm rate (FA), missed speech rate (Miss), confusion, purity and coverage.

5.2.4 Analysis

Even though the jointly optimized pipeline shows a better performance than separately optimized pipeline, the false alarm rate is increased. As shown in Figure 5.4, the jointly optimized pipeline "prefers" to ignore short non-speech segments. To take advantage of our separately optimized VAD system, we can also post-process the result by removing the non-speech part in the separately optimized VAD results. This operation brings a 1% decrease in diarization error rate as shown in Table 5.1.

We also do an analysis of SCD in both pipelines. Minimum segment duration δ_{peak} converges to zero in an separately optimized SCD system. However, in the jointly optimized pipeline, minimum segment duration δ_{peak} is converged to be around 3s. That may be because longer segments are easier to cluster. As shown in Figure 5.4, the confusion error is mostly caused by short segments. That also explains why the jointly optimized pipeline prefers to ignore short non-speech segments.

5.3 Neural sequential clustering

Given an audio recording, the clustering step in speaker diarization system aims at grouping the speech turns according to the speaker identities. Since it does not need to determine the actual speaker identities, any permutation of the labels are equivalent (*e.g.* 'aabbcc' is equivalent to 'bbaacc').

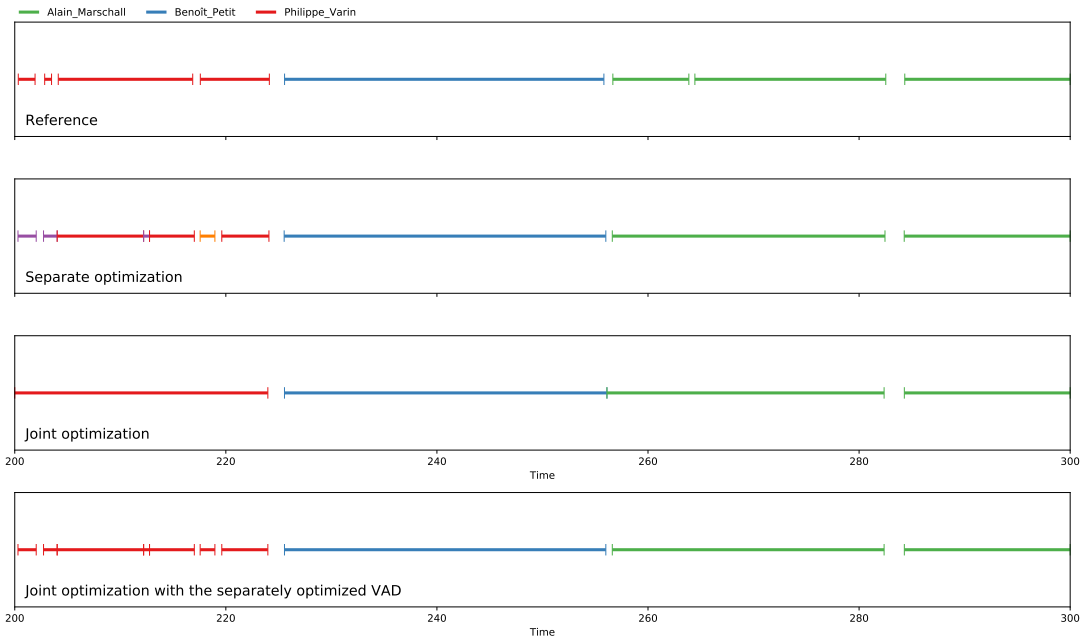

Figure 5.4: An example of diarization results in different pipelines.

5.3.1 Motivations

Most clustering algorithms such as hierarchical agglomerative clustering and spectral clustering need to be provided with a number of clusters or a stopping criterion to determine how many clusters should be generated. However, in a speaker diarization task, audio files vary in number of speakers, and a global optimal threshold may not be optimal for each file. Recently, computer vision and natural language processing tasks improved a lot thanks to the end-to-end learning.

An end-to-end system is usually composed of neural networks and treated as an adaptive black box that generates the prediction from the input data directly without any intermediate steps. However, clustering is an unsupervised task, while almost all the existing end-to-end systems are supervised. It is difficult to design a differentiable loss function close to diarization error rate or to standard clustering metrics.

Even though in Chapter 4, we propose to use stacked RNNs to improve the

similarity matrix, the proposed system is trained on a supervised binary classification task and still relies on the spectral clustering backend. An end-to-end sequential clustering system should be able to map the input vectors sequence directly to the cluster labels bypassing the similarity matrix. More precisely, we would like a sequential clustering system that takes a sequence of speaker embeddings (*e.g.* extracted on a 1s sliding window) as input and returns a sequence of cluster labels of the same length.

We are going to work on a toy problem as a Proof of Concept (PoC), where we propose to address the sequential clustering as a sequence labeling task similarly to VAD and SCD introduced in Chapter 3.

5.3.2 Principle

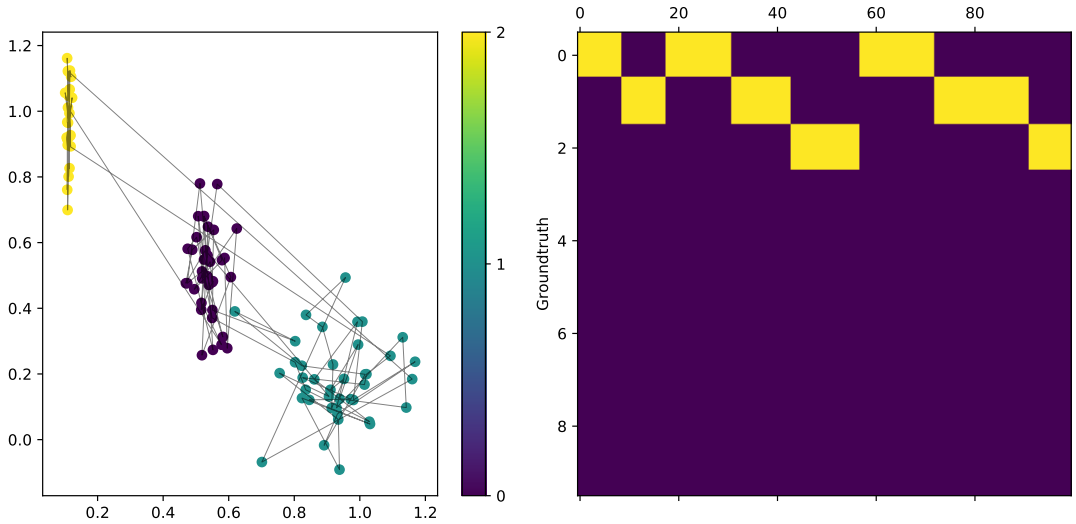

Figure 5.5: An example of sequential clustering.

Let $\mathbf{x} \in \mathcal{X}$ be a sequence of segment embedding vectors as shown in the left part of Figure 5.5: $\mathbf{x} = (x_1, \ldots, x_N)$, and let $\mathbf{y} \in \mathcal{Y}$ be the corresponding sequence of clustering labels as shown in the right part of Figure 5.5: $\mathbf{y} = (y_1, \ldots, y_N)$ and $y_i \in \{0, \ldots, n_{\max}\}$, where N is the length of sequence and $n_{\max}$ is the maximum

number of speakers estimated on the training set. Because this is a clustering task, it is also correct to predict any permutation of y cluster indices as shown in Figure 5.6. The objective is to train a function $g : \mathcal{X} \to \mathcal{Y}$ that matches an

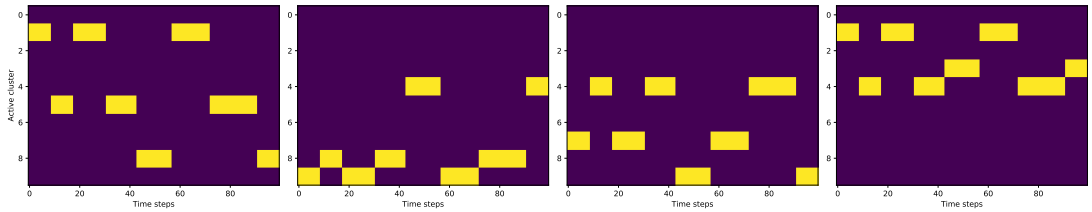

Figure 5.6: All four predictions are equivalent because they all are permutations of the same clustering result.

embedding sequence $\mathbf{x}$ to the corresponding label sequence $\mathbf{y}$.

5.3.3 Loss function

Learning tasks can be considered as optimization problems seeking to minimize a loss function that measures prediction inaccuracy. For neural approaches, almost every loss function is designed for supervised learning. Speaker diarization is usually evaluated by the diarization error rate. However, it is not differentiable because it relies internally on the Hungarian algorithm that is solved by dynamic programming. As shown in Figure 5.6, any permutation of reference cluster indices is also correct. Therefore, an alternative loss function is motivated by the permutation invariant training [104]:

$$\min_{r \in R} L(r, \hat{y}) \tag{5.1}$$

where R is the set of permutation of reference y cluster indices, $\hat{y}$ is the prediction, and L is any traditional loss function for classification tasks such as mean squared error or category cross-entropy. This loss function first determines the optimal output-target assignment and then computes the loss. However, it may cause a high cost of computation during training.

To simplify the sequential clustering task, we convert it into a supervised sequence labeling task: the first speaker in a sequence should be labeled as '1', second as '2' and the other speakers are labeled according to their chronological order as shown in Figure 5.5. Then, the category cross-entropy can be used to train the system.

5.3.4 Model architectures

In this paragraph, we describe the different network architectures for sequential clustering.

5.3.4.1 Stacked RNNs

Stacked RNNs are the most used architecture for sequence labeling tasks, and they have been successfully applied in our previous works in VAD and SCD. Therefore, in this section, stacked RNNs are also applied to model the function g.

5.3.4.2 Encoder-decoder

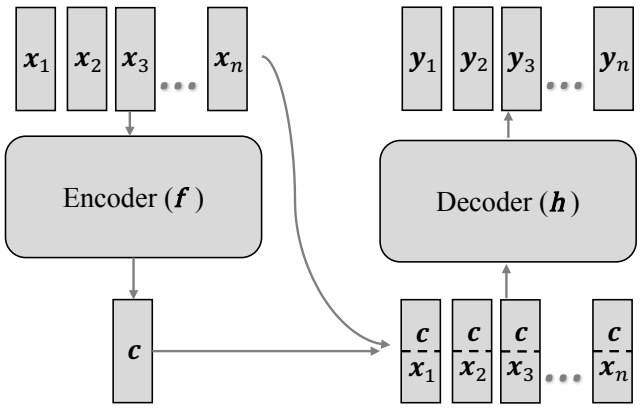

Figure 5.7: Encoder-decoder for sequential clustering.

Motivated by the successful application of encoder-decoder in machine trans-

lation and other sequence-to-sequence tasks, we propose an encoder-decoder architecture for sequential clustering. Generally, the encoder f aims at mapping an input sequence into an internal representation vector $\boldsymbol{c}$ which is then used to generate an output sequence by the decoder h.

Our proposed architecture for sequential clustering task is shown in Figure 5.7. The encoder consists of stacked RNNs which read the embedding vector one by one. The final hidden state of the final RNN is defined as the context vector $\boldsymbol{c}$, which represents the summary of the input sequence.

$$\boldsymbol{c} = f(\mathbf{x}) \tag{5.2}$$

We expect $\boldsymbol{c}$ to contain information about the whole input sequence (such as the number of clusters, the position of the centroids, *etc.*). This will be discussed in Section 5.3.9.1.

The decoder is another RNN which is used to generate the output sequence of labels. Unlike the decoder in traditional architecture proposed in [34], the input of our proposed decoder consists of two parts: the context vector $\boldsymbol{c}$ and the original embedding vectors. The input at timestep t is the concatenation of $\boldsymbol{c}$ and $\mathbf{x}_t$. Therefore, the output sequence is computed using:

$$\mathbf{y} = h(\mathbf{x}, \boldsymbol{c}) = h\left(\begin{bmatrix}\boldsymbol{c}\\ \boldsymbol{x}_1\end{bmatrix}, \begin{bmatrix}\boldsymbol{c}\\ \boldsymbol{x}_2\end{bmatrix}, \cdots, \begin{bmatrix}\boldsymbol{c}\\ \boldsymbol{x}_n\end{bmatrix}\right) \tag{5.3}$$

The architecture of the decoder is motivated by our previous work in Chapter 4, in which we used stacked RNNs to predict the similarity matrix, and where the input of RNN is the concatenation of two embedding vectors.

The intuition is that context $\boldsymbol{c}$ contains the centroids information, and decoder RNN could compare $\boldsymbol{c}$ with the input embedding vectors to guess the cluster label and smooth the resulting sequence temporally. While in traditional sequence-to-sequence tasks, input and output sequences can have different lengths, it is not

the case in our sequential clustering task.

5.3.5 Simulated data

An end-to-end system usually needs to be trained with numerous data. Since we work on a toy problem as a proof of concept, we start with some toy simulated data. The generated sequence should include sequential information. Our proposed simulated data generative process involves two parts: label generation ($\mathbf{y}$) and embedding generation ($\mathbf{x}$). To simplify the visualization of the clustering results, the dimension of the embedding vectors is fixed to 2, and the sequence length is also fixed to 100.

5.3.5.1 Label generation y

Label generation aims at modeling the generative process of speaker turns. We use two strategies to generate the label sequences: toy and mimic.

The toy generator relies on a traditional Markov model. For each sequence, the number of clusters is first initialized randomly, and then the prior probability and the transition matrix are also randomly initialized. Since the speech turns in a real conversation are not uniform distributed, the label duration is modeled by a discrete Poisson distribution.

Mimic generator relies on a real diarization dataset which includes the annotation of "who spoke when". As shown in Figure 5.8, an annotation file is first randomly selected from the dataset (Part A). Then the duration of each segment is randomly modified (up to 20%), and the labels are also randomly modified with the probability of 0.05 (Part B). The output sequence of labels is a random part of the modified annotation file without non-speech (Part C & D).

Both toy and mimic label generation techniques share the same post-processing step: rename the labels to make sure clusters are numbered in chronological order.

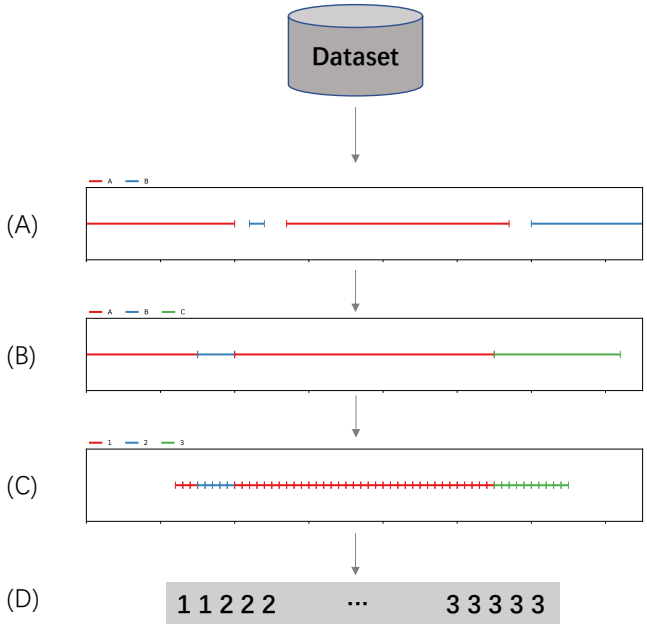

Figure 5.8: Mimic label generation.

5.3.5.2 Embedding generation (x)

Embedding generation aims at modeling the generative process of speaker embeddings. For a generated sequence of labels, each label corresponds to a cluster (speaker), modeled by a Gaussian model. For instance, in Figure 5.5, there are three clusters, whose means and variances are initialized randomly.

5.3.6 Baselines

The proposed end-to-end clustering system is compared with three baselines. The first one is hierarchical agglomerative clustering with complete or pooling linkage which were introduced in 2.5.1.1. The second one is affinity propagation (AF), which has been successfully applied in our previous work. The third one is UIS-RNN, which is essentially a mixture of RNN and parametric models. Similar to our proposed approaches, UIS-RNN also models sequential information. Therefore, it is expected to be the best of the three baselines.

5.3.7 Implementation details

5.3.7.1 Data

The length of the generated sequence is fixed to 100. For toy label generator, the number of clusters is sampled from a discrete uniform distribution over $[1, 10)$ and the λ in Poisson distribution is 10. In other words, the average length of speech turns is 10. For mimic label generator, the REPERE database serves as conversation templates. For embedding generator, the cluster centers and variance are sampled from the continuous uniform distribution over $[0.0, 1.0)$.

5.3.7.2 Stacked RNNs

Different from our previous task such as VAD, SCD, we use the Gated Recurrent Units (GRU) as RNN instead of the Long Short-Term Memory (LSTM). As shown in Figure 5.9, the architecture is composed of three parts: linear, RNNs, and output. Linear is a fully connected layer without activation function, which is used to transform the input data dimension from 2 to the same dimension as the hidden size in RNN. RNN is composed of several bi-directional RNN layers (2 for toy data, 3 for mimic data). All the RNN layers have 256 (128×2) outputs. Because we do not model overlap and each point belongs to exactly one cluster, the output layer is a linear layer with a softmax activation function.

5.3.7.3 Encoder-decoder architecture

As shown in Figure 5.10, the encoder is composed of two parts: linear and RNNs. Similar to stacked RNNs, linear is used to transform the input data dimension. RNNs is composed of several bi-directional RNN layers (1 for toy data, 2 for mimic data) and the output size is 256 (128×2). The decoder is composed of a single bi-directional RNN and an output layer, where the input of RNN is the concatenation of the last hidden state (128×2) of the encoder and the transformation of original input (128). The output layer is a linear layer with a

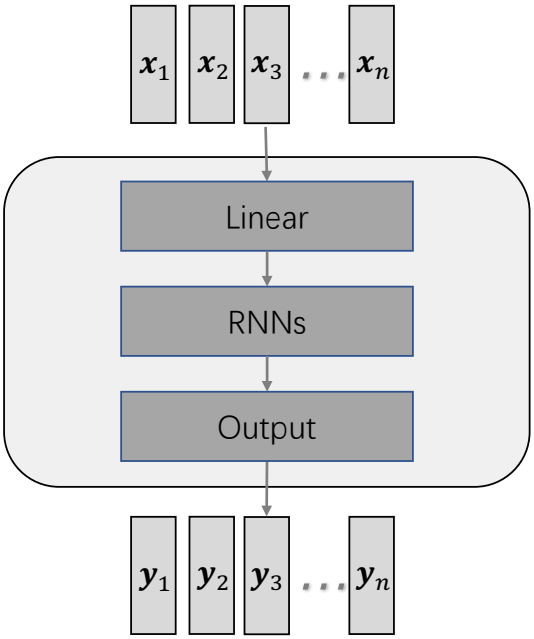

Figure 5.9: Stacked RNNs.

softmax activation function like stacked RNNs.

5.3.7.4 Training and testing

In the training process, the Adam optimizer is employed with a learning rate initialized at 0.001 and divided by 10 every 200 epoch. The model training process terminates after 500 epochs. Then a development set with 1000 sequences is used to select the best epoch. In the testing process, the model is evaluated on a test set which contains 1000 sequences.

Note that all the data is generated randomly and relies on a random seed. We ensure that training, development, and test sets are different by using a different random seed for each of them.

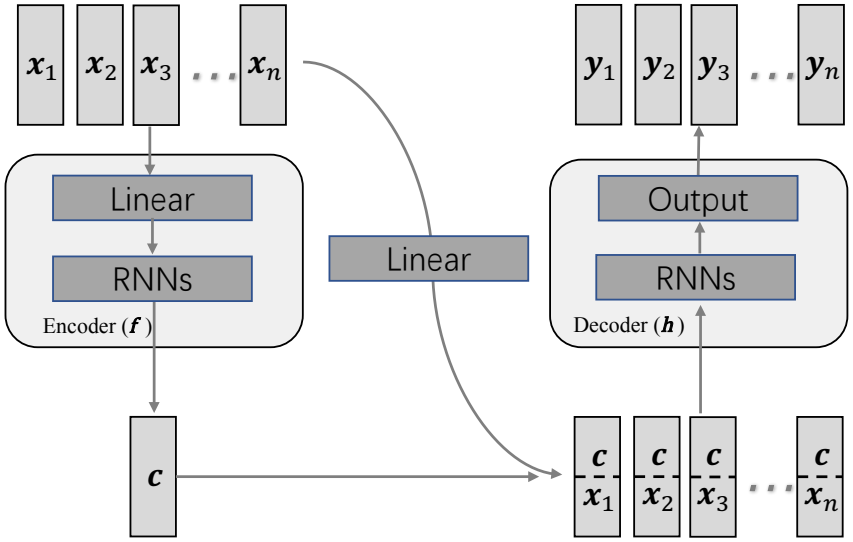

Figure 5.10: Encoder-decoder.

5.3.7.5 Hyper-parameters tuning for baselines

For each baseline, 1000 sequences taken from the training set are used to tune the hyper-parameters. The threshold θ_{HAC} for hierarchical agglomerative clustering and preference value θ_{AP}, damping factor λ_{AP} for affinity propagation are tuned by *scikit-optimize* [89] in order to minimize the diarization error rate. UIS-RNN is trained with 20000 epoch with its official code in github[1], and the model of the last epoch is selected.

5.3.8 Results

All systems are evaluated by diarization error rate, purity, and coverage. Since we exclude the non-speech in our generated data, the DER referred here is the class confusion. Table 5.2 summarizes the main experimental results on toy data. The top three systems (stacked RNNs, encoder-decoder, and UIS-RNN) are based on RNNs, and they can model the sequential information during the training process, while the traditional clustering methods such as hierarchical agglomerative

[1]https://github.com/google/uis-rnn

	DER	Purity	Coverage
Stacked RNNs	7.4	94.04	95.88
Encoder-decoder	8.5	93.18	94.85
UIS-RNN	14.6	85.50	97.77
HAC (pool)	23.0	81.43	89.19
HAC (average)	23.5	82.45	87.44
AF	24.8	82.71	84.36

Table 5.2: Results of different systems on toy data.

	DER	Purity	Coverage
Stacked RNNs	10.78	92.35	94.80
Encoder-decoder	13.12	90.59	92.63
UIS-RNN	13.65	86.50	98.21
HAC (pool)	29.61	78.94	90.00
HAC (average)	28.47	78.67	90.29
AF	25.84	77.41	89.69

Table 5.3: Results of different systems on mimic data.

clustering and affinity propagation just process the segments independently. The experimental results on the toy data show that the RNN-based systems lead to significant improvements over conventional systems. Stacked RNNs reaches the best performance.

An example of clustering results of traditional methods is shown in Figure 5.11. We can find that most data points have been grouped to the correct clusters. However, there are some fast speech turns in predicted label sequences. The results of RNN-based methods are shown in Figure 5.12. The results have been smoothed, and almost all the data points are grouped into the correct clusters.

When we switch to the mimic data, all the performances of different systems degrade a little, except the UIS-RNN. That may be because UIS-RNN is more suitable to model the real speech turn information.

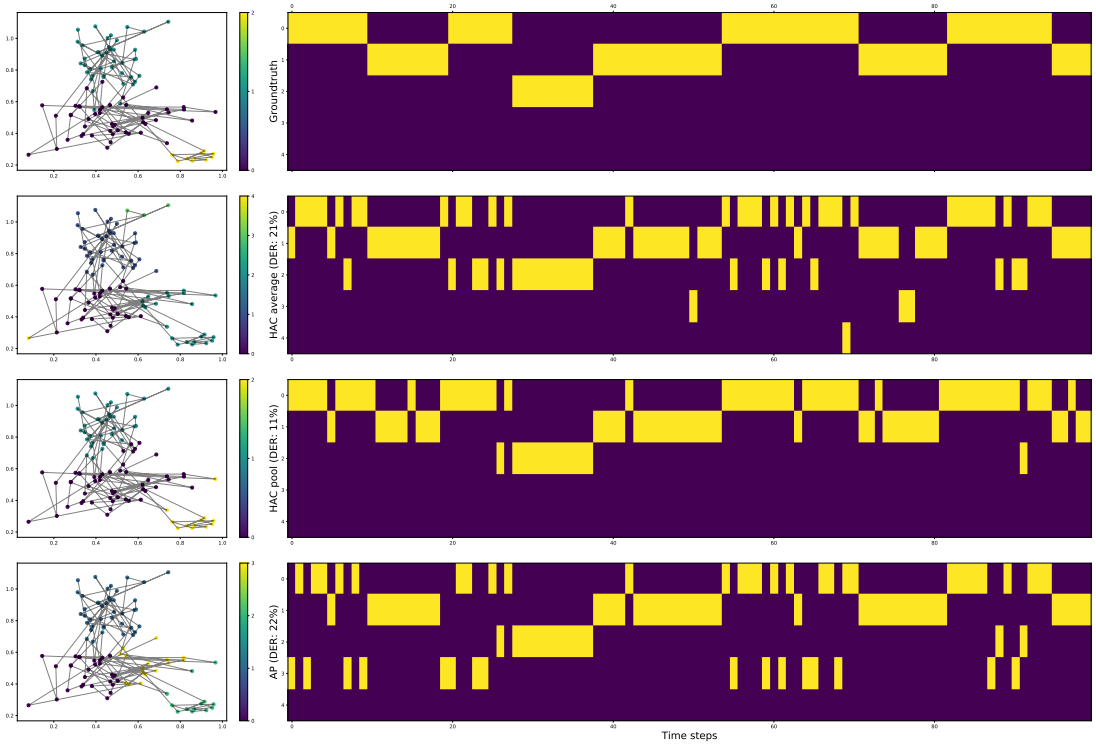

Figure 5.11: Clustering results of traditional methods.

5.3.9 Discussions

5.3.9.1 What does the encoder do?

We anticipate that our proposed encoder-decoder imitates the process of humans doing the clustering, where one first guesses the centroids based on the data points (encoder) and then aligns the data points to the clusters according to the distance between data points and corresponding centroids (decoder). We expect that the context vector $\boldsymbol{c}$ contains the centroids information and the decoder is able to compare the data points with $\boldsymbol{c}$. As we have already successfully applied stacked RNNs to improve the similarity matrix, we also choose bi-directional RNN as decoder. However, neural network modules are like black boxes, and it is difficult to analyze their inner behavior directly.

To well understand the working mechanism of encoder-decoder, as shown in

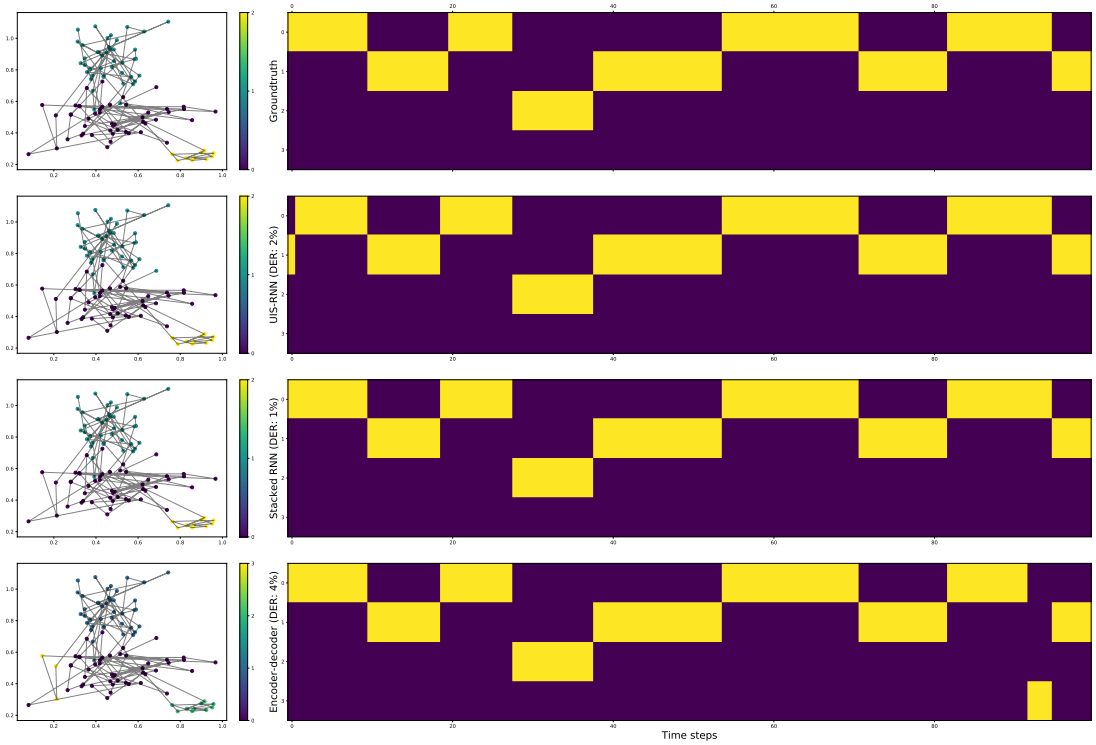

Figure 5.12: Clustering results of RNN-based methods.

Figure 5.13, we replace the decoder by MLP to predict the number of clusters of an input sequence. The encoder is taken from a trained encoder-decoder model, and its parameters are frozen. The MLP is composed of two fully connected layers, which are 128- and 10-dimensional respectively. This model is trained with 50 epochs and the model of the last epoch is used to test on 1000 randomly generated sequences.

We compute the absolute difference between the predicted number of clusters and the reference number of clusters. The difference distribution is shown in the left part of Figure 5.14. It shows that more than 60% are predicted correctly, and when we choose a tolerance of 1, it reaches 90%. The right part of Figure 5.14 is the distribution of number of clusters. If we use a naive classification model that always predicts 5, only 21% is correct. It means that the context vector does contain information about the number of clusters.

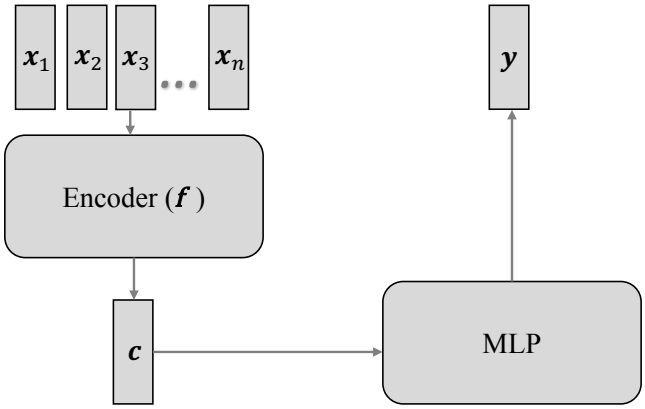

Figure 5.13: The architecture used to predict the number of clusters of an input sequence.

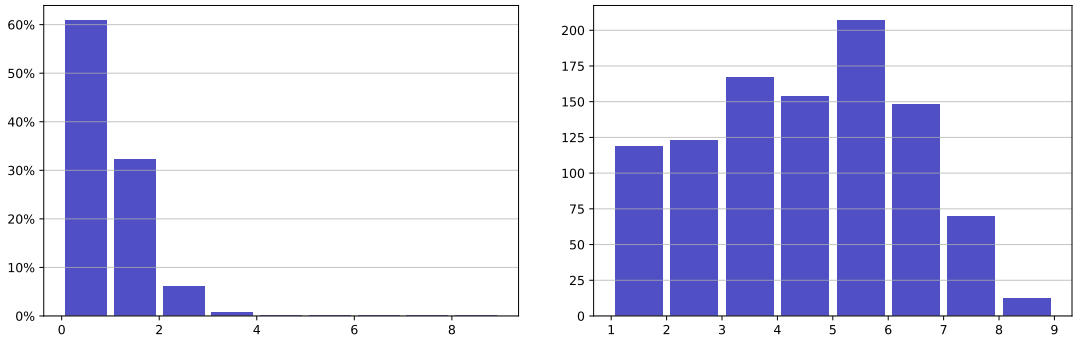

Figure 5.14: The difference between the predicted number of clusters and the reference number of clusters (left). The distribution of number of clusters (right). Experiments are conducted on toy data.

5.3.9.2 Neural sequential clustering on long sequences

In real speaker diarization datasets such as ETAPE and AMI, the conversations may last more than half an hour. However, in our previous experiments, the sequence length is fixed to 100. Since most speaker embedding systems are designed to embed the audio segments of 1s, the duration of the sequences in our previous experiments would only be 1 minute 40 seconds. Since LSTM and GRU are designed to memorize long-term dependencies of sequences, we also conduct the experiments with longer sequences where the length of sequences is extended

to 600. The results of different systems are presented in Table 5.4. All systems show a significant decrease in performance, compared with the results on short sequences.

	DER	Purity	Coverage
Stacked RNNs	11.8	90.22	93.43
Encoder-decoder	13.3	88.61	91.95
UIS-RNN	15.83	86.21	95.84
HAC (pool)	32.17	77.71	88.87
HAC (average)	31.33	77.37	89.07
AF	28.50	76.91	86.60

Table 5.4: Results on long sequences.

5.3.9.3 Sequential clustering with stacked unidirectional RNNs.

Bi-directional RNN can process the sequence from forward and backward directions at the same time. However, this architecture is restricted to offline clustering. To adapt the stacked RNNs to online sequential clustering, we re-ran the experiments with unidirectional RNN where the backward direction is discarded. Similarly to our proposed stacked bi-directional RNNs, it is composed of two standard RNN layers. Both of them are 256-dimensional. Table 5.5 presents the results of stacked unidirectional RNNs. The performance degrades a lot compared to the stacked bi-directional RNNs. Nevertheless, it still better than traditional clustering methods, even though they are offline.

Length	DER	Purity	Coverage
100	13.9	89.62	91.10
600	17.3	85.95	89.71

Table 5.5: Results of stacked unidirectional RNNs.

5.4 Conclusion

In this chapter, we first introduce the joint optimization for our proposed diarization pipeline. Compared to the pipeline with separately optimized modules, the new pipeline shows a significant improvement. In addition, we propose to do end-to-end sequential clustering directly with stacked RNNs and an encoder-decoder model. The experiments are conducted on toy data, and our proposed systems show a much better performance than traditional clustering algorithms. The main reason may be because the sequential information is modeled by these RNN-based methods. In addition, the stacked unidirectional RNNs are also successfully applied in our experiments which may lead to an online sequential clustering system in the future.

Sequential clustering is an important task not only in speaker diarization but also in other applications with time series data. For example, wearable sensor data can be expressed as a timeline of a few actions (walking, sleeping *etc.*) [105]. In the future, we will test our proposed methods with real speaker embeddings and other sequential clustering applications.

Chapter 6

Conclusions and Perspectives

6.1 Conclusions

Overall, the main topic of this thesis is to improve the speaker diarization system with neural networks. In this thesis, all modules of our proposed diarization systems are addressed with neural network approaches. The main contributions of this thesis are summarized as follows:

- **First contribution.** We show that both the initial segmentation and the final re-segmentation can be formulated as a set of frame-wise sequence labeling problems on top of MFCC features, addressed using bidirectional LSTMs. The proposed methods lead to significant performance improvement in broadcast TV dataset. Recently, LSTM-based methods also achieve state-of-the-art performance on most other sequence labeling tasks, comparing with other probabilistic methods. That may be because the LSTMs can learn the context required to make a prediction at each time step. Because conversational speech is usually highly structured, contextual information is critical for segmentation tasks. This type of information is difficult to capture by probabilistic models.

- **Second contribution.** Traditional clustering modules in diarization systems rely on variations of Hierarchical Agglomerative Clustering (HAC)

approaches and use BIC, CLR or i-vector to compute similarities between clusters. In recent years, the performance of state-of-the-art speaker recognition systems has improved enormously, thanks to the neural-based speaker embedding systems. We propose to use affinity propagation clustering on top of a neural speaker embedding system introduced in [48; 95]. Experiments on a broadcast TV dataset show that affinity propagation clustering is more suitable than hierarchical agglomerative clustering when applied to neural speaker embeddings. In addition, we propose to improve the similarity matrix by bidirectional LSTM and then apply spectral clustering on top of the improved similarity matrix. The proposed system achieves state-of-the-art performance in the CALLHOME telephone conversation dataset. The analysis shows that the improvement mainly results from the sequence modeling of the LSTM model on longer recordings.

- **Third contribution.** While speaker diarization modules are usually tuned empirically and independently from each other, we propose to jointly optimize the whole diarization pipeline composed of neural-based segmentation and affinity propagation. Compared to the pipeline with separately optimized modules, the new pipeline shows a significant improvement on a broadcast TV dataset.

- **Fourth contribution.** We formulated sequential clustering as a supervised sequence labeling task and addressed it with stacked RNNs. To better understand its behavior, the analysis is based on a proposed encoder-decoder architecture. Our proposed systems bring a significant improvement compared with traditional clustering methods on toy examples. It appears that stacked RNNs is capable to model the whole sequence.

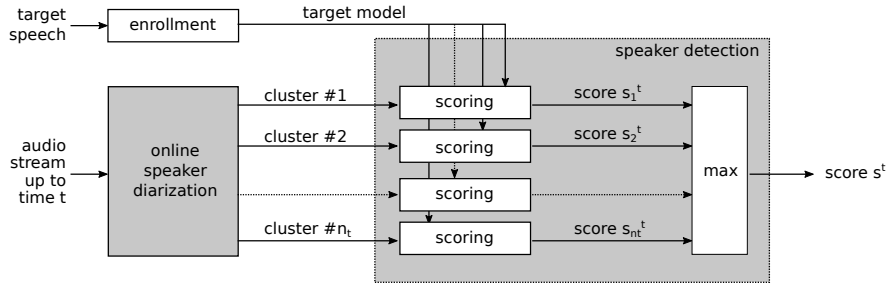

Figure 6.1: Common architecture to proposed LLSS solutions. At any time t, online speaker diarization provides a set of n_t speaker clusters $\{c_i^t\}_{1\leq i\leq n_t}$. Speaker detection is then applied to compare the speech segments in each cluster c_i^t against a pre-trained target speaker model, thereby giving scores (or likelihood-ratios) s_i^t. A final score at time t is defined as the maximum score over all clusters: $s^t = \max_{1\leq i\leq n_t} s_i^t$. We provide several backends. Our proposed d-vector embedding backend achieve the best performance. Figure taken from [6].

6.2 Perspectives

Due to limited time, some promising research perspectives could not be investigated during my thesis.

6.2.1 Sequential clustering in real diarization scenarios

In Chapter 5, we proposed to use stacked RNNs and encoder-decoder for the sequential clustering task. Even though the two proposed models show an excellent performance with toy data, we did not have time to test them in real diarization scenarios. Our short term goal for the sequential clustering task is to explore the applicability of our systems in real scenarios and try other neural network architectures. Recently, transformer [106], encoder-decoder with attention mechanism [2] and Neural Turing Machine (NTM) [107] have been successfully applied for sequence to sequence tasks in natural language processing domain, such as machine translation. These architectures could be also used for the sequential clustering task. In addition, the proposed loss function (categorical cross-entropy) assumes that cluster indices are ordered chronologically. We would like to relax

this constraint by investigating permutation invariant losses closer to the standard diarization error rate evaluation metric.

6.2.2 Overlapped speech detection

Overlapped speech is a very common phenomenon in human conversations like meetings and phone calls. Our proposed systems in this thesis can only assign speech segments to one speaker, thus incurring missed speech errors in overlapped speech regions where two or more speakers are active. Preliminary experiments show that overlapped speech detection can also be formulated as a sequence labeling problem (y = 1 for overlap, y = 0 otherwise), addressed using bidirectional LSTMs like VAD and SCD. Our short term goal is to integrate the LSTM-based overlapped detection into our proposed diarization systems. In addition, our proposed end-to-end sequential clustering models in Chapter 5 cannot model overlapped speech. Our long term goal is to handle overlapped speech during the sequential clustering. Therefore, the neural network architectures proposed in Chapter 5 should be modified. For instance, the activation function in output layers could be switched from softmax to sigmoid, while the loss function could be replaced by mean squared error.

6.2.3 Online diarization system

Speaker diarization is often used as a preprocessing step in some other applications such as ASR. In some scenarios like meetings and lectures, the ASR system should be in real time. In [6], we proposed a new task termed low-latency speaker spotting (LLSS). It consists in determining as early as possible when a specific speaker starts talking in an audio stream. Our proposed system architecture for LLSS is depicted in Figure 6.1, which combines online speaker diarization with speaker detection approach. With the growth of these types of applications, online diarization systems become more and more important in speech processing domain. For initial segmentation, our proposed system can be done in an online

manner, with a latency of 3.2s (the sliding window size). For sequential clustering, in Chapter 5, we tried to do it by a standard RNN. However, the performance degraded a lot on toy data, comparing with bidirectional RNN. Our short term goal for this task is to adapt our proposed systems to an online manner and apply them to real diarization scenarios. Our long term goal is developing an adequate architecture for online speaker diarization.

6.2.4 End-to-end diarization system

Even though some parts of the proposed diarization system are based on neural approaches, the system still relies on hand-crafted features (MFCC), and this is therefore not an end-to-end speaker diarization system. An end-to-end system should be able to map the waveform directly to the diarization result bypassing the feature extraction and other steps. Rather than employing standard hand-crafted features, [108] proposes a novel CNN architecture, called SincNet, to learn low-level speech representations from waveforms directly. The proposed architecture converges faster and performs better than a standard CNN on raw waveforms in the speaker verification task. Our preliminary experiments on VAD and SCD achieved the same conclusion. It seems that SincNet is powerful enough to replace the traditional hand-crafted feature extractors in speech processing tasks. Our short term goal for this task is to replace the MFCC feature extractor by SincNet in our proposed systems. Our long term goal is developing a real end-to-end speaker diarization system.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016. xiii, 13, 15, 16, 17, 19

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *ICLR 2015, International Conference on Learning Representations*, 2015. xiii, 18, 19, 98

[3] M. Wang and W. Deng, "Deep Face Recognition: A Survey," *arXiv preprint arXiv:1804.06655*, 2018. xiii, 22

[4] C. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval," *Natural Language Engineering*, vol. 16, no. 1, p. 385, 2010. xv, 64, 65

[5] Q. Lin, R. Yin, M. Li, H. Bredin, and C. Barras, "Recurrent Neural Network Based Segments Similarity Measurement with Spectral Clustering for Speaker Diarization," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*, (Graz, Austria), September 2019. xv, xvii, 59, 70, 75

[6] J. Patino, R. Yin, H. Delgado, H. Bredin, A. Komaty, G. Wisniewski, C. Barras, N. Evans, and S. Marcel, "Low-latency Speaker Spotting with Online Diarization and Detection," in *Odyssey 2018, The Speaker and Language Recognition Workshop*, 2018. xvi, 98, 99

[7] S. E. Tranter and D. A. Reynolds, "An Overview of Automatic Speaker Diarization Systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557–1565, 2006. 1, 7

[8] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "First DIHARD Challenge Evaluation Plan," 2018. 2

[9] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-End Text-Dependent Speaker Verification," in *ICASSP 2016, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5115–5119, IEEE, 2016. 2, 21

[10] A. Graves, "Neural Networks," in *Supervised Sequence Labelling with Recurrent Neural Networks*, pp. 15–35, Springer, 2012. 2, 40, 42

[11] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM Neural Networks for Language Modeling," in *Interspeech 2012, 13th Annual Conference of the International Speech Communication Association*, pp. 194–197, 2012. 3, 40

[12] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *NIPS 2014, Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014. 3, 40

[13] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep Neural Networks for Small Footprint Text-dependent Speaker Verification," in *ICASSP 2014, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4052–4056, IEEE, 2014. 3, 21

[14] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN Embeddings for Speaker Recognition," in *ICASSP 2018, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5329–5333, IEEE, 2018. 3, 21, 61

[15] B. J. Frey and D. Dueck, "Clustering by Passing Messages Between Data Points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007. 3, 29, 64

[16] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker Diarization with LSTM," in *ICASSP 2018, IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2018. 3, 24, 28, 29, 31, 59, 71, 73

[17] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker diarization: A Review of Recent Research," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 356–370, 2012. 7, 23, 26

[18] C. Barras, X. Zhu, S. Meignier, and J. L. Gauvain, "Multi-Stage Speaker Diarization of Broadcast New," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, pp. 1505–1512, Sept. 2006. 8, 49

[19] J. Ajmera and C. Wooters, "A Robust Speaker Clustering Algorithm," in *ASRU 2003, IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 411–416, IEEE, 2003. 8

[20] M. Nosratighods, E. Ambikairajah, and J. Epps, "Speaker Verification Using A Novel Set of Dynamic Features," in *ICPR 2006, 18th International Conference on Pattern Recognition*, vol. 4, pp. 266–269, IEEE, 2006. 9

[21] P. Rose, *Forensic Speaker Identification*. CRC Press, 2002. 9

[22] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke, "Modeling Prosodic Feature Sequences for Speaker Recognition," *Speech Communication*, vol. 46, no. 3-4, pp. 455–472, 2005. 9

[23] A. Adami, R. Mihaescu, D. Reynolds, and J. Godfrey, "Modeling Prosodic Dynamics for Speaker Recognition," in *ICASSP 2003, IEEE International*

Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. IV – 788, 05 2003. 9

[24] G. Friedland, O. Vinyals, Y. Huang, and C. Muller, "Prosodic and Other Long-Term Features for Speaker Diarization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 5, pp. 985–993, 2009. 9

[25] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989. 11

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, p. 436, 2015. 12

[27] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations," in *ICML 2009, 26th Annual International Conference on Machine Learning*, pp. 609–616, ACM, 2009. 12

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A Large-Scale Hierarchical Image Database," in *CVPR 2009, IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, IEEE, 2009. 12

[29] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A Large-Scale Speaker Identification Dataset," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, (Stockholm, Sweden), August 2017. 12

[30] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *AISTATS 2011, 14th International Conference on Artificial In-*

telligence and Statistics, (Ft. Lauderdale, FL, USA), pp. 315–323, April 2011. 14

[31] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *ICML 2013, 30th International Conference on Machine Learning*, (Atlanta, USA), p. 3, June 2013. 14

[32] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," in *ICLR 2016, International Conference on Learning Representations*, (San Juan, Puerto Rico), May 2016. 14

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification," in *IEEE International Conference on Computer Vision*, (Santiago, Chile), pp. 1026–1034, December 2015. 14

[34] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *EMNLP 2014, Conference on Empirical Methods in Natural Language Processing*, 2014. 17, 18, 84

[35] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," 2016. 19

[36] D. Reynolds, *Universal Background Models*, pp. 1349–1352. Boston, MA: Springer, 2009. 20

[37] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000. 20

[38] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint Factor Analysis versus Eigenchannels in Speaker Recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007. 20

[39] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011. 20, 60

[40] T. Yamada, L. Wang, and A. Kai, "Improvement of Distant-Talking Speaker Identification Using Bottleneck Features of DNN.," in *Interspeech 2013, 14th Annual Conference of the International Speech Communication Association*, pp. 3661–3664, 2013. 21

[41] S. H. Ghalehjegh and R. C. Rose, "Deep Bottleneck Features for i-vector Based Text-Independent Speaker Verification," in *ASRU 2015, IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 555–560, IEEE, 2015. 21

[42] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification.," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, (Stockholm, Sweden), pp. 999–1003, August 2017. 21

[43] V. Peddinti, D. Povey, and S. Khudanpur, "A Time Delay Neural Network Architecture for Efficient Modeling of Long Temporal Contexts," in *Interspeech 2015, 16th Annual Conference of the International Speech Communication Association*, 2015. 21

[44] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep Neural Network-Based Speaker Embeddings for End-

to-End Speaker Verification," in *SLT 2016, IEEE Spoken Language Technology Workshop*, pp. 165–170, IEEE, 2016. 21

[45] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep Learning Face Representation by Joint Identification-Verification," in *Advances in Neural Information Processing Systems*, pp. 1988–1996, 2014. 22, 61

[46] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A Unified Embedding for Face Recognition and Clustering," in *CVPR 2015, IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015. 22, 61

[47] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep Speaker Recognition," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association*, 2018. 22

[48] H. Bredin, "TristouNet: Triplet Loss for Speaker Turn Embedding," in *ICASSP 2017, IEEE International Conference on Acoustics, Speech, and Signal Processing*, (New Orleans, USA), March 2017. 22, 25, 34, 40, 48, 56, 59, 62, 97

[49] C. Zhang and K. Koishida, "End-to-End Text-Independent Speaker Verification with Flexibility in Utterance Duration," in *ASRU 2017, IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 584–590, 2017. 22, 62

[50] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A Discriminative Feature Learning Approach for Deep Face Recognition," in *ECCV 2016, European Conference on Computer Vision*, pp. 499–515, Springer, 2016. 22

[51] J. Ramirez, J. M. Górriz, and J. C. Segura, "Voice Activity Detection. Fundamentals and Speech Recognition System Robustness," in *Robust Speech Recognition and Understanding*, InTech, 2007. 23

[52] K.-H. Woo, T.-Y. Yang, K.-J. Park, and C. Lee, "Robust Voice Activity Detection Algorithm for Estimating Noise Spectrum," *Electronics Letters*, vol. 36, no. 2, pp. 180–181, 2000. 23

[53] M. Marzinzik and B. Kollmeier, "Speech Pause Detection for Noise Spectrum Estimation by Tracking Power Envelope Dynamics," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 2, pp. 109–118, 2002. 23

[54] R. Tucker, "Voice Activity Detection Using A Periodicity Measure," *IEE Proceedings I (Communications, Speech and Vision)*, vol. 139, no. 4, pp. 377–380, 1992. 23

[55] E. Nemer, R. Goubran, and S. Mahmoud, "Robust Voice Activity Detection Using Higher-Order Statistics in the LPC Residual Domain," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 217–231, 2001. 23

[56] E. Rentzeperis, A. Stergiou, C. Boukis, A. Pnevmatikakis, and L. C. Polymenakos, "The 2006 Athens Information Technology Speech Activity Detection and Speaker Diarization Systems," in *International Workshop on Machine Learning for Multimodal Interaction*, pp. 385–395, Springer, 2006. 24

[57] A. Temko, D. Macho, and C. Nadeu, "Enhanced SVM Training for Robust Speech Activity Detection," in *ICASSP 2007, IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. IV–1025, IEEE, 2007. 24

[58] T. Ng, B. Zhang, L. Nguyen, S. Matsoukas, X. Zhou, N. Mesgarani, K. Veselỳ, and P. Matějka, "Developing A Speech Activity Detection System for the DARPA RATS Program," in *Interspeech 2012, 13th Annual*

Conference of the International Speech Communication Association, 2012. 24

[59] N. Ryant, M. Liberman, and J. Yuan, "Speech Activity Detection on Youtube Using Deep Neural Networks," in *Interspeech 2013, 14th Annual Conference of the International Speech Communication Association*, pp. 728–731, Lyon, France, 2013. 24

[60] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker Diarization Using Deep Neural Network Embeddings," in *ICASSP 2017, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4930–4934, IEEE, 2017. 24, 73

[61] M. A. Siegler, U. Jain, B. Raj, and R. M. Stern, "Automatic Segmentation, Classification and Clustering of Broadcast News Audio," in *Proc. DARPA speech recognition workshop*, vol. 1997, 1997. 25, 27, 40, 48

[62] S. Chen and P. Gopalakrishnan, "Speaker, Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, vol. 8, pp. 127–132, Virginia, USA, 1998. 25, 27, 40, 48

[63] B. Desplanques, K. Demuynck, and J.-P. Martens, "Factor Analysis for Speaker Segmentation and Improved Speaker Diarization," in *Interspeech 2015, 16th Annual Conference of the International Speech Communication Association*, pp. 3081–3085, 2015. 25, 40

[64] V. Gupta, "Speaker Change Point Detection Using Deep Neural Nets," in *ICASSP 2015, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4420–4424, IEEE, 2015. 25

[65] M. Hrúz and Z. Zajíc, "Convolutional Neural Network for Speaker Change Detection in Telephone Speaker Diarization System," in *ICASSP 2017,*

IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4945–4949, IEEE, 2017. 25

[66] D. Dimitriadis and P. Fousek, "Developing On-Line Speaker Diarization System.," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, (Stockholm, Sweden), pp. 2739–2743, August 2017. 25, 31

[67] S. Bozonnet, N. W. Evans, and C. Fredouille, "The LIA-EURECOM RT'09 Speaker Diarization System: Enhancements in Speaker Modelling and Cluster Purification," in *ICASSP 2010, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4958–4961, IEEE, 2010. 26

[68] Q. Jin and T. Schultz, "Speaker Segmentation and Clustering in Meetings," in *ICSLP 2004, 18th International Conference on Spoken Language Processing*, 2004. 27

[69] S. J. Prince and J. H. Elder, "Probabilistic Linear Discriminant Analysis for Inferences about Identity," in *ICCV 2007, 11th IEEE International Conference on Computer Vision*, pp. 1–8, 2007. 27

[70] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," in *18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007. 28

[71] O. Ben-Harush, O. Ben-Harush, I. Lapidot, and H. Guterman, "Initialization of Iterative-Based Speaker Diarization Systems for Telephone Conversations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 414–425, 2012. 28

[72] S. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised Methods for Speaker Diarization: An Integrated and Iterative Approach," *IEEE*

Transactions on Audio, Speech, and Language Processing, vol. 21, no. 10, pp. 2015–2028, 2013. 29

[73] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized End-to-End Loss for Speaker Verification," in *ICASSP 2018, IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 4879–4883, 2018. 29

[74] W. Zhu and J. Pelecanos, "Online Speaker Diarization using Adapted i-vector Transforms," in *ICASSP 2016, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5045–5049, IEEE, 2016. 31

[75] K. Church, W. Zhu, J. Vopicka, J. Pelecanos, D. Dimitriadis, and P. Fousek, "Speaker diarization: a perspective on challenges and opportunities from theory to practice," in *ICASSP 2017, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4950–4954, IEEE, 2017. 31

[76] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully Supervised Speaker Diarization," in *ICASSP 2019, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6301–6305, IEEE, 2019. 31, 62, 63, 72, 73

[77] C. Wooters and M. Huijbregts, "The ICSI RT07s speaker diarization system," in *Multimodal Technologies for Perception of Humans*, pp. 509–519, Springer, 2007. 32

[78] G. Sell and D. Garcia-Romero, "Diarization Resegmentation in the Factor Analysis Subspace," in *ICASSP 2015, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4794–4798, IEEE, 2015. 32, 73

[79] A. Giraudel, M. Carré, V. Mapelli, J. Kahn, O. Galibert, and L. Quintard,

"The REPERE Corpus: A Multimodal Corpus for Person Recognition.," in *LREC*, pp. 1102–1107, 2012. 32

[80] G. Gravier, G. Adda, N. Paulson, M. Carré, A. Giraudel, and O. Galibert, "The ETAPE Corpus for the Evaluation of Speech-based TV Content Processing in the French Language," in *LREC - Eighth international conference on Language Resources and Evaluation*, (Turkey), p. na, 2012. 32

[81] H. Bredin, "pyannote.metrics: A Toolkit for Reproducible Evaluation, Diagnostic, and Error Analysis of Speaker Diarization Systems," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, (Stockholm, Sweden), August 2017. 35, 37, 38

[82] M. Cettolo, "Segmentation, Classification and Clustering of An Italian Broadcast News Corpus," in *Content-Based Multimedia Information Access-Volume 1*, pp. 372–381, 2000. 35, 37

[83] J.-L. Gauvain, L. Lamel, and G. Adda, "Partitioning and transcription of broadcast news data.," in *ICSLP 1998, 5th International Conference on Spoken Language Processing*, vol. 98, pp. 1335–1338, 1998. 35, 37

[84] G. Gelly and J.-L. Gauvain, "Minimum Word Error Training of RNN-based Voice Activity Detection.," in *Interspeech 2015, 16th Annual Conference of the International Speech Communication Association*, pp. 2650–2654, 2015. 40, 41, 43

[85] C. Barras, X. Zhu, S. Meignier, and J.-L. Gauvain, "Improving Speaker Diarization," in *RT-04F Workshop*, 2004. 40

[86] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 273–278, IEEE, 2013. 42

[87] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 42

[88] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, "YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software.," in *ISMIR 2010, 11th International Society for Music Information Retrieval Conference*, pp. 441–446, 2010. 43, 65

[89] G. M. Kumar and T. Head, "Scikit-Optimize," 2017. 44, 78, 89

[90] P.-A. Broux, F. Desnous, A. Larcher, S. Petitrenaud, J. Carrive, and S. Meignier, "S4D: Speaker Diarization Toolkit in Python," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association*, September 2018. 45

[91] J. G. Fiscus, N. Radde, J. S. Garofolo, A. Le, J. Ajot, and C. Laprun, "The Rich Transcription 2005 spring meeting recognition evaluation," in *International Workshop on Machine Learning for Multimodal Interaction (MLMI)*, pp. 369–389, Springer, 2005. 47

[92] O. Galibert, J. Leixa, G. Adda, K. Choukri, and G. Gravier, "The ETAPE Speech Processing Evaluation.," in *LREC 2014, Language Resources and Evaluation*, pp. 3995–3999, 2014. 47

[93] O. Galibert, "Methodologies for the Evaluation of Speaker Diarization and Automatic Speech Recognition in the Presence of Overlapping Speech," in *Interspeech 2013, 14th Annual Conference of the International Speech Communication Association*, pp. 1131–1134, 2013. 47

[94] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "The Second DIHARD Diarization Challenge: Dataset, task, and baselines," in *Interspeech 2019, 20th Annual Conference of the Inter-*

national Speech Communication Association, (Graz, Austria), September 2019. 55

[95] G. Wisniewski, H. Bredin, G. Gelly, and C. Barras, "Combining Speaker Turn Embedding and Incremental Structure Prediction for Low-Latency Speaker Diarization," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, August 2017. 59, 66, 97

[96] X. Zhang, J. Gao, P. Lu, and Y. Yan, "A Novel Speaker Clustering Algorithm via Supervised Affinity Propagation," *ICASSP 2008, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4369–4372, 2008. 59

[97] G. Gelly and J.-L. Gauvain, "Spoken Language Identification using LSTM-based Angular Proximity," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, August 2017. 62

[98] D. Snyder, G. Chen, and D. Povey, "Musan: A Music, Speech, and Noise Corpus," *arXiv preprint arXiv:1510.08484*, 2015. 63

[99] L. v. d. Maaten and G. Hinton, "Visualizing Data Using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008. 66

[100] G. Sell and D. Garcia-Romero, "Speaker Diarization with PLDA i-vector Scoring and Unsupervised Calibration," in *SLT 2014, IEEE Spoken Language Technology Workshop*, pp. 413–417, IEEE, 2014. 71

[101] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *ASRU 2011, IEEE Workshop on Automatic Speech Recognition and Understanding*, no. EPFL-CONF-192584, IEEE Signal Processing Society, 2011. 71

[102] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, *et al.*, "Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association*, pp. 2808–2812, 2018. 71

[103] J. Bergstra, D. Yamins, and D. Cox, "Making A Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures," in *International Conference on Machine Learning*, pp. 115–123, 2013. 78

[104] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation Invariant Training of Deep Models for Speaker-Independent Multi-Talker Speech Separation," in *ICASSP 2017, IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 241–245, 2017. 82

[105] D. Hallac, S. Vare, S. Boyd, and J. Leskovec, "Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data," in *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 215–223, 2017. 95

[106] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," in *NIPS 2017, Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017. 98

[107] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," *arXiv preprint arXiv:1410.5401*, 2014. 98

[108] M. Ravanelli and Y. Bengio, "Speaker Recognition from Raw Waveform with SincNet," in *SLT 2018, IEEE Spoken Language Technology Workshop*, pp. 1021–1028, 2018. 100

[109] J. Patino, H. Delgado, R. Yin, H. Bredin, C. Barras, and N. Evans, "ODESSA at Albayzin Speaker Diarization Challenge 2018," in *IberSPEECH 2018*, 2018.

[110] R. Yin, H. Bredin, and C. Barras, "Neural speech turn segmentation and affinity propagation for speaker diarization," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association*, 2018.

[111] F.-F. Li, A. Karpathy, and J. Johnson, "CS231n: Convolutional Neural Networks for Visual Recognition," *University Lecture*, 2015.

[112] Y. Liu, M. Russell, and M. Carey, "The Role of Dynamic Features in Text-Dependent and-Independent Speaker Verification," in *ICASSP 2006, IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 1, IEEE, 2006.

[113] H. Bredin, "pyannote.audio." `https://github.com/pyannote/pyannote-audio`, 2017.

[114] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[115] S. Galliano, E. Geoffrois, D. Mostefa, K. Choukri, J.-F. Bonastre, and G. Gravier, "The ESTER Phase II Evaluation Campaign for the Rich Transcription of French Broadcast News," in *9h European Conference on Speech Communication and Technology*, 2005.

[116] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons, 2012.

[117] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms," in

Proceedings of the 12th Python in Science Conference, pp. 13–20, Citeseer, 2013.

[118] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-parameter Optimization," in *Advances in Neural Information Processing Systems*, pp. 2546–2554, 2011.

[119] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. October, pp. 2825–2830, 2011.

[120] D. Müllner, "Modern Hierarchical, Agglomerative Clustering Algorithms," *arXiv preprint arXiv:1109.2378*, 2011.

[121] S. Meignier and T. Merlin, "LIUM SpkDiarization: An Open Source Toolkit for Diarization," in *CMU SPUD Workshop*, 2010.

[122] A. Larcher, K. A. Lee, and S. Meignier, "An Extensible Speaker Identification Sidekit in Python," in *ICASSP 2016, IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 5095–5099, IEEE, 2016.

[123] P.-A. Broux, D. Doukhan, S. Petitrenaud, S. Meignier, and J. Carrive, "An Active Learning Method for Speaker Identity Annotation in Audio Recordings," in *MMDA@ ECAI*, pp. 23–27, 2016.

[124] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining* (J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, eds.), (Berlin, Heidelberg), pp. 160–172, Springer Berlin Heidelberg, 2013.

[125] M. Rouvier, G. Dupuy, P. Gay, E. el Khoury, T. Merlin, and S. Meignier, "An Open-source State-of-the-art Toolbox for Broadcast News Diariza-

tion," in *Interspeech 2013, 14th Annual Conference of the International Speech Communication Association*, 2013.

[126] R. Yin, H. Bredin, and C. Barras, "Speaker Change Detection in Broadcast TV using Bidirectional Long Short-Term Memory Networks," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, (Stockholm, Sweden), August 2017.

[127] S. H. Yella, A. Stolcke, and M. Slaney, "Artificial neural network features for speaker diarization," in *SLT 2014, Spoken Language Technology Workshop*, pp. 402–406, IEEE, 2014.

[128] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ICLR 2015, International Conference on Learning Representations*, 2015.

[129] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *International Workshop on Multiple Classifier Systems*, pp. 1–15, Springer, 2000.

[130] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Étapes vers un système neuronal de bout en bout pour la tâche de segmentation et de regroupement en locuteurs

Mots clés : segmentation et regroupement en locuteurs, détection des changements de locuteurs, segmentation, LSTM, propagation d'affinité, partitionnement spectral

Résumé : Dans cette thèse, nous proposons de traiter le problème de segmentation et regroupement en locuteurs à l'aide d'approches neuronales.
Nous formulons d'abord le problème de la segmentation initiale (détection de l'activité vocale et des changements entre locuteurs) et de la re-segmentation finale sous la forme d'un ensemble de problèmes d'étiquetage de séquence, puis nous les résolvons avec des réseaux neuronaux récurrents de type Bi-LSTM (*Bidirectional Long Short-Term Memory*).
Au stade du regroupement des régions de parole, nous proposons d'utiliser l'algorithme de propagation d'affinité à partir de plongements neuronaux de ces tours de parole dans l'espace vectoriel des locuteurs. Des expériences sur un jeu de données télévisées montrent que le regroupement par propagation d'affinité est plus approprié que le regroupement hiérarchique agglomératif lorsqu'il est appliquée à des plongements neuronaux de locuteurs. La segmentation basée sur les réseaux récurrents et la propagation d'affinité sont également combinées et optimisées conjointement pour former une chaîne de regroupement en locuteurs. Comparé à un système dont les modules sont optimisés indépendamment, la nouvelle chaîne de traitements apporte une amélioration significative. De plus, nous proposons d'améliorer l'estimation de la matrice de similarité par des réseaux neuronaux récurrents, puis d'appliquer un partitionnement spectral à partir de cette matrice de similarité améliorée. Le système proposé atteint des performances à l'état de l'art sur la base de données de conversation téléphonique CALLHOME.
Enfin, nous formulons le regroupement des tours de parole en mode séquentiel sous la forme d'une tâche supervisée d'étiquetage de séquence et abordons ce problème avec des réseaux récurrents empilés. Pour mieux comprendre le comportement du système, une analyse basée sur une architecture de codeur-décodeur est proposée. Sur des exemples synthétiques, nos systèmes apportent une amélioration significative par rapport aux méthodes de regroupement traditionnelles.

Title : Steps towards end-to-end neural speaker diarization

Keywords : speaker diarization, speaker change detection, speech segmentation, LSTM, affinity propagation, spectral clustering

Abstract : In this thesis, we propose to address speaker diarization with neural network approaches.
We first formulate both the initial segmentation (voice activity detection and speaker change detection) and the final re-segmentation as a set of sequence labeling problems and then address them with bidirectional Long Short-Term Memory (Bi-LSTM) networks.
In the speech turn clustering stage, we propose to use affinity propagation on top of neural speaker embeddings. Experiments on a broadcast TV dataset show that affinity propagation clustering is more suitable than hierarchical agglomerative clustering when applied to neural speaker embeddings. The LSTM-based segmentation and affinity propagation clustering are also combined and jointly optimized to form a speaker diarization pipeline. Compared to the pipeline with independently optimized modules, the new pipeline brings a significant improvement. In addition, we propose to improve the similarity matrix by bidirectional LSTM and then apply spectral clustering on top of the improved similarity matrix. The proposed system achieves state-of-the-art performance in the CALLHOME telephone conversation dataset.
Finally, we formulate sequential clustering as a supervised sequence labeling task and address it with stacked RNNs. To better understand its behavior, the analysis is based on a proposed encoder-decoder architecture. Our proposed systems bring a significant improvement compared with traditional clustering methods on toy examples.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France