UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Department of Computer Systems and Computation

Master's Degree Final Project

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Adaptation and assessment of speaker diarization models for open educational resources

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Vicent Andreu Ciscar Martínez

Directors:
Dr. Alfons Juan Ciscar
Dr. Albert Sanchis Navarro

Experimental director:
Adrià Giménez Pastor

September 17, 2018

Contents

Prologue

In this work we will investigate a quite current task, *Speaker Diarization*. This task aims to find the segments that correspond to each speaker in a multi-speaker audio. Its use covers several real areas such as, for example, the transcription of telephone conversations, broadcast news, movies or domain-specific videos like a surgery operations documentation. In addition, the latest developments in the area are based on a technology as current as artificial neural networks.

In Chapter 1, several aspects encompassed by the task of Speaker Diarization are briefly discussed, at an introductory level. In addition, the motivation that has helped to decide on the realization of this project will be explained and the objectives to be achieved will be defined. Also, the different tools used during the development of this work are pointed out.

In Chapter 2, we cover the work done using a well-know toolkit in Speaker Diarization: the LIUM toolkit. This set of tools has been a basis for a few years now and, as in our case, some authors [8] use it as a baseline for their experiments.

Another toolkit used is Pyannote. This library performs a whole pipeline based on recurrent neural networks and is explained in the Chapter 3. We will also use several tools available in the toolkit to structure the corpus data and to evaluate the results obtained throughout the work.

Chapter 4 is about the segmentation of a corpus of speech data called *Video Actas*. This corpus is composed of audios of plenary sessions of various town halls. It will be explained how the corpus has been obtained, what characteristics it has, the experiments carried out and the results that have been obtained using the LIUM toolkit.

Finally, in Chapter 5 the different conclusions that have been reached from the various experiments carried out, the problems encountered, the results obtained and the corpus used are described.

Chapter 1

Introduction

1.1 Introduction

Automatic Speech Recognition (ASR) is an area seeking to improve current technological systems that try to convert an audio signal into a sequence of words. Although ASR systems are often applied to single-speaker speech recordings, there is an increasing interest in applying these systems to the more challenging case in which multiple speakers are recorded. In this case, a very important (sub)task is to segment the recorded audio into single-speaker segments. This task is called *Speaker Diarization (SD)*.

Speaker Diarization tries to solve the problem of "who spoke when?". For this reason it has utility in most applications that perform an audio or video processing and contain more than one active speaker. Some of these applications are telephone conversations, broadcast news, debates, shows, movies, meetings, domain-specific videos or even lecture or conference recording [14].

Currently the research on Speaker Diarization is progressing to a large extent thanks to the National Institute of Standards and Technology [1] which defined the task evaluation method. In addition, due to the current development of deep learning in different fields, Speaker Diarization systems based on artificial neural networks are showing good results despite the complexity of the task [33]. In this work we will aim to understand where the complexity of several Speaker Diarization systems lies and evaluate the results obtained.

We will explain in the next sections several important aspects for introducing some concepts related to the Speaker Diarization task. First of all we will see what has been the motivation to get into this task of speech recognition and what are the objectives that we want to achieve throughout the work. Then we will provide several concepts that will help us to understand the elements that make up this task and that range from the obtaining of features to the segmentation of the audios in the turns of the different speakers. Finally, in this chapter we will find a brief description of the tools used and, after that, what are the main values that are sought for the

evaluation of a Speaker Diarization system.

1.2 Motivation

During the master's degree, I had the opportunity to develop a speaker recognition system based on Gaussian mixture models. This system was simply to recognize a user by voice and accept him as a client. However, it led me to consider in which other areas a similar recognition was also used. As explained in the Section 1, the objective of a Speaker Diarization system is the location of the fragments pronounced by different speakers along an audio signal. Therefore, to a certain extent, a work on Speaker Diarization allowed me to expand my knowledge in this area.

In addition, we must add the fact that currently the application of Speaker Diarization systems is trying to use real tasks such as the transcription of telephone calls, where it is necessary to know the words spoken by each user, or even the automatic subtitling of TV shows [4]. This diversity in the possibilities of the task was a very interesting addition.

Another aspect that motivated me to do this work was the possibility of working with artificial neural networks. In current systems of Speaker Diarization network architectures are being used close to the state-of-the-art. Some of the systems use convolutional layers, recurrent networks, d-vectors and other very current elements in the development of deep learning. For all this the possibility of conducting an investigation on Speaker Diarization was seen as a good knowledge to acquire.

1.3 Objectives

In the area of speech recognition, much progress has been made in recent years with the arrival of neural networks. Within this area we find the task of speaker diarization which, as discussed above, consists in labeling the fragments of the speakers at the intervals in which they speak. In this regard, the main objectives of this work (and some ideas to achieve them) are:

- **Understand and become familiar with the task of Speaker Diarization**. For this purpose, a literature review will be carried out. Documentation will be sought near the state-of-the-art as well as documentation related to the first functional systems.

- **Apply state-of-the-art Speaker Diarization systems**. We also want to find a toolkit that allows us to obtain a baseline with already tested and reproducible results. The aim is to compare the results obtained with the tests carried out throughout the work. Then we want to find and apply a system that uses automatic neural networks and brings us closer to the current state-of-the-art.

- **Improve the results obtained with the Speaker Diarization systems**. Another objective that we want to reach with this work is the improvement of the toolkits that we can apply. After evaluating each one separately with the default parameters, some way of improving the results obtained will be considered.

- **Find a corpus that is related to the academic or public sphere and try the studied Speaker Diarization techniques/tools on it**. And finally, to make the comparison between the various systems, in addition to the corpus of data with proven results, it is intended to find a corpus that is related to the academic or public sphere. In this way we will observe the results that can be obtained with audios that are currently being used for other tasks and we will evaluate if Speaker Diarization is an aspect to be taken into account.

1.4 Speaker Diarization

The main goal in ASR is to search for the sequence of words pronounced in a given speech audio fragment. For this, this discipline is responsible for the conception and realization of automatic systems that process acoustic signals and transform them into specific sequences of linguistic elements. ASR systems usually require a language model for the search of the pronounced words. In this way they use the syntactic, semantic and pragmatic knowledge to guide the automatic systems towards the minimum possible error (WER) (see Sec. 1.12).

With the advance of new technologies, especially thanks to the considerable increase in computing capacity, some branches of the area of speech recognition have continued to improve. One of them, in which we deepen in this work, is the speaker diarization. As discussed above, this task consists of subdividing the audio, as necessary, to determine "who spoke when?". This is done without knowing a priori the number of speakers that spoke in the audio [33].

To carry out the whole process of diarization, several steps are taken. In Figure

1.1 we can see an example of this process [20]. The different parts are speech activity detection (SAD), segmentation of audio and its subsequent clustering and, finally, a speaker recognition for each cluster. In the next sections, these parts will be explained, which together form the pipeline for the complete conversion of the signal in the different outputs that can be obtained.

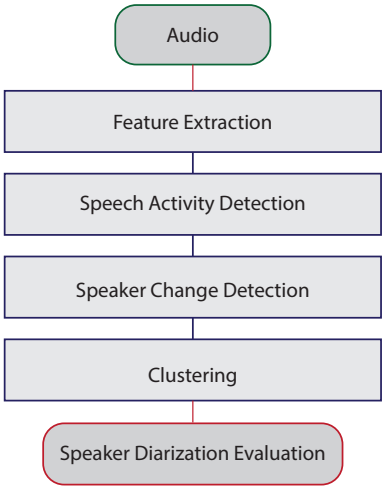

Figure 1.1: Typical Speaker Diarization system

1.5 Feature extraction

The initial part of any speech recognizer is the conversion of the signal into a series of features that will form the input of the system [40]. In the case of speaker diarization the different types of features that can be obtained can be divided into three types: short-term spectrum based features [3], spatial features [38] and long-term features [15].

In most of the speaker diarization systems the short-term spectrum based features are used. In this kind the features are extracted from the short-term spectrum. This is due to the information of the vocal tract characteristics of individual speakers that the short-term spectrum based features carry on. These features are typical extracted for every 10 ms from a window size of around 30 ms. Many systems use Mel frequency cepstral coeficients (MFCCs) as features for speaker diarization [22].

Nowadays, another audio embedding extraction is done after the consecutive segmentations, where specific features such as MFCCs, speaker factors or i-vectors [32] are extracted from the segmented sections.

1.6 Speaker Recognition

The main objective of a speaker recognition system [30] is to identify a person through his or her voice. Being able to make a distinction between the physiological characteristics and the linguistic habits of different speakers is what allows this identification to be carried out correctly. Therefore, the first step is to perform an extraction of characteristics inherent to the speaker and then compare them with other patterns, or models.

Like most recognition systems, the speaker recognizer can be divided into two parts: training and testing. In the training, it seeks the recording of the speakers using a feature extractor and at a later stage the models of each speaker are obtained. In the part of the test the features of the unknown speakers that could appear are extracted, and their models are compared with those already stored. The process finalizes when the system obtain the possible coincidences and shows the speakers that present a more similar features to those sought in the exam.

In general, the process for the implementation of a speaker recognizer starts from the *data acquisition*, which is essential to be able to extract the features of the speaker in the best possible way. For this it is necessary to use an acoustic-electric transducer that transforms the sound pressure into electrical signal and can digitize it. After that, the *feature extraction* is performed. In this task, the audio is processed to extract the vectors that the recognizer needs as input. These vectors present a low level of abstraction and are limited to describing spectral, parametric and temporal features of the audio signal.

From the vectors that contain the features of the signal the training and the test are carried out in the part of *classification* of the system. In this part of the recognizer, the vectors are compared and the probabilities that an unknown speaker coincides with one of the speakers stored in the database are calculated. The decision about the speaker's identification depends on the configuration of the system, but usually it is the one that maximizes the probability.

1.7 Speech Activity Detection

This segmentation consists in dividing the audio into fragments and label them depending on whether they are composed by speech or not. Those that are not speech not only include silences, but also music, breathing sounds, background noise, among others [29].

The first works of speaker diarization developed, tried to obviate the part of SAD, replacing it with another cluster corresponding to the non-speech part, however it was observed to affect negatively the output of the system [23].

Errors that can occur in a Speech Activity Detection (SAD) system affect diarization directly, since missing speech fragments or false alarm detections contribute to the calculation of the *diarization error rate (see Sec. 1.12)*.

1.8 Speaker Change Detection

The Speaker Change Detection (SCD) task is also known as the Segmentation part of a speaker diarization system. The objective of this task is to segment speech fragments into homogeneous subfragments that only correspond to a speaker. Another way of looking at it would be that its objective is to detect where the speaker changes occur. The most common way to perform this part is to compare two speech windows and make hypotheses to see if they are from different speakers or is the same speaker. These windows move along the audio regularly to detect all the points of change.

The hypotheses are estimated thanks to the stored models of the speakers and the calculation of a criterion based on the similarity or distance between them. The main criteria proposed throughout the literature related to the diarization task are the following:

Bayesian information criterion:

The Bayesian Information Criterion (BIC) is a model selector criterion [Schwarz, 1978] used to determine which model best explains the available data. It has been used for SCD in [Chen and Gopalakrishan, 1998].

The value of a model M representing a data set X is given by:

$$BIC(M) = log\mathscr{L}(X|M) - \frac{\lambda}{2}\#(M)log(N) \tag{1.1}$$

where, $\mathcal{L}(X|M)$ is the likelihood of data X given the model M, N is the number of data points in X, $\#(M)$ is the number of parameters in the model M, and λ is a trade-off between the model complexity and how well the models fit the given data set.

The difference in BIC values sought for the comparison between the various models is calculated as follows:

$$\Delta BIC(i,j) = log\mathcal{L}(X_{ij}|M_{ij}) - [log\mathcal{L}(X_i|M_i) + log\mathcal{L}(X_j|M_j)] - \frac{\lambda}{2}\delta_{ij}log(N) \quad (1.2)$$

where, X_{ij} is the combined data and M_{ij} is the estimated model of the combined data, δ_{ij} is the difference in the parameters of both models and X_i or X_j is the data of each subwindow, M_i and M_j are the different models that the system is testing.

Generalized likelihood ratio:

Another criterion that is usually used to perform speaker change detection is the generalized likelihood ratio (GLR) [19, 17, 18, 16]. This criterion is similar to ΔBIC, but does not take into account the complexity of the model.

$$GLR(i,j) = \frac{\mathcal{L}(X_{ij}|M_{ij})}{\mathcal{L}(X_i|M_i) + \mathcal{L}(X_j|M_j)} \quad (1.3)$$

$\mathcal{L}(X|M)$ denotes the likelihood of data X given a model M. On the other hand, X_{ij} and M_{ij} denote the data and model resulting from the combination of segments X_i and X_j.

KL divergence:

The divergence of Kullback Leiber (KL) is used to measure the inequality between two probability distributions. In this way, the distribution of the data in two segments is compared to decide if there has been a change of the speaker or not. When this divergence is modelled by Gaussian distributions it can be obtained with the following formula:

$$KL(X_i||X_j) = \frac{1}{2}tr[(\Sigma_i - \Sigma_j)(\Sigma_j^{-1} - \Sigma_i^{-1}) + (\Sigma_j^{-1} - \Sigma_i^{-1})(\mu_i - \mu_j)(\mu_i - \mu_j)^{'})] \quad (1.4)$$

where, μ_k,Σ_k denote the mean and covariance of data X_k.

A symmetric form is also used (KL2):

$$KL2(X_i, X_j) = KL(X_i||X_j) + KL(X_j||X_i) \quad (1.5)$$

When using the KL criterion, only one Gaussian distribution can be modelled for each segment, this greatly limits the modeling capacity and leads to an expense in computing. For this reason, this criterion is usually only used as a first step, and then improve the segmentation with the BIC criterion.

1.9 Clustering and re-alignment

The clustering part is responsible for taking the homogeneous segments obtained in the previous task and joining them up to the point of only having one cluster for speaker. To perform this task, the same criteria can be used as in the segmentation task, BIC, GLR, KL and KL2 [39].

When the pipeline pass from the segmentation task to the clustering task, the errors of the first task accumulate. To prevent this from happening, after joining two clusters, a re-alignment is made by Viterbi algorithm to stabilize the model of each cluster.

1.10 Speaker diarization systems

Up to this point, we have seen the different tasks that form a complete system of speaker diarization. However, there are several ways to perform some of these tasks and therefore different systems of speaker diarization can be composed [35].

HMM/GMM system:

This type of speaker diarization system [24] represents each speaker as a state within a Hidden Markov Model (HMM) and the probability of emission from each state is modelled using Gaussian Mixture Models (GMM).

The clusters are initialized with a uniform segmentation in the zones where speech has been detected and this generates a group of segments (X_i). Afterwards, each segment is treated as a cluster and modeled as a HMM state. To calculate the probability of emission of each state, the following function must be taken into account:

$$\log b_i(s_t) = \log \sum_{(r)} w_i^{(r)} N(s_t, \mu_i^{(r)}, \Sigma_i^{(r)}), \tag{1.6}$$

where $N()$ is a Gaussian and $w_i^{(r)}$, $\mu_i^{(r)}$ and $\Sigma_i^{(r)}$ are the weights, means and

covariance matrices respectively of the r^{th} Gaussian mixture component of a cluster i. In the previous formula b_i denote the emission probability distribution of one cluster and s_t denote a feature vector on time t.

Clustering in an agglomerative framework starts by over-estimating the number of speaker clusters. At each iterative step, the clusters that are most similar to each other based on a distance measure are merged. The similarity between two clusters is measured using a modified delta Bayesian information criterion [Ajmera et al., 2004]. The modified ΔBIC criterion $\Delta BIC(c_i, c_j)$ for two clusters is given by:

$$\Delta BIC(c_i, c_j) = \sum_{s_t \epsilon \{c_i \cup c_j\}} \log b_{ij}(s_t) - \sum_{s_t \epsilon c_i} \log b_i(s_t) - \sum_{s_t \epsilon c_j} \log b_j(s_t) \tag{1.7}$$

where b_{ij} is the probability distribution estimated over the combined data of cluster c_i and c_j. The clusters that produce the highest ΔBIC score are merged. After each merge step, a Viterbi decoding pass realigns the speech data to the new speaker clusters.

Deep neural network systems:

The improvement of technology that has been in recent times has allowed computers to be able to process much more information. This has allowed the development of artificial neural networks to a large extent. These networks are able to "learn" by means of a given reference to perform a desired task. In the scope of Speaker Diarization, artificial neural networks were introduced to improve some parts of the entire pipeline [27].

In the part of speaker recognition it was seen that one way to improve the features extracted from each speaker was to obtain d-vectors. These vectors collect information from the speaker from single-speaker audio and using deep neural networks (DNN). Many audios of the same speaker are passed through the network and at the end the weights of the penultimate layer are saved as the identification vector of that speaker. This vector is used as a comparison reference in a speaker recognition system. Although the d-vectors gave a good result, research continues on new systems to improve the extraction of characteristics [34].

Nowadays, neural networks are being used to improve steps of the Speaker Diarization task such as speech activity detection, speech change detection, speech turn embedding, clustering, etc. The architectures of the networks are very different and in recent years there has been a shift from the use of convolutional networks to the addition of recurring layers (LSTMs). A more detailed description of a system that

uses recurring networks can be seen in Chapter 3.

1.11 Computer tools

For the realization of this work, the tools used have been:

- **LIUM** [10]. This toolkit is a software dedicated to speaker diarization. It is written in Java and was developed for the French ESTER2 evaluation campaign, where it obtained the best results for the task of speaker diarization of broadcast news in 2008.
- **TensorFlow** [11]. TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.
- **Jupyter Notebook** [13]. Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis.
- **Python** [28]. Python is an interpreted high-level programming language for general-purpose programming.
- **pyannote.metrics** [6]. A toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems.
- **pyannote.audio** [37]. A toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems.
- **Yaafee** [9]. A toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems.

1.12 Evaluation of results: DER and WER

- *DER*:

  For the evaluation of Speaker Diarization systems, the main evaluation measure is the Diarization Error Rate (DER). It was introduced by the NIST in 2000.

The calculation of the DER is done in two steps, the first is to make a mapping between the labels returned by the system and the identities of the speakers stored in the references. The next step is to calculate the error that occurs once the mapping is done.

For the total calculation of this evaluation value it must be taken into account that the error can occur for three reasons. These types of errors are given by the context in the diarization system:

- The confusion error, when the speaker labeling given by the system and the one in the reference do not coincide during the mapping.
- The loss error, when in the reference there is a fragment of speech that does not occur in the output of the system.
- The false alarm error, when a speech fragment is detected by the system and in the reference corresponds to a segment without speech.

These previous errors can occur during a certain period of time, to calculate the error we must add these times together. In case of overlapping speech between several speakers, the complexity of the loss error and the false alarm error increase considerably. The formula for calculating the DER is as follows:

$$DER = \frac{confusion + miss + falsealarm}{totalreferencespeechtime} \tag{1.8}$$

Some flexibility must be added to the calculation of the DER to take into account the possible human error made in annotating the reference. Therefore, a margin of $+/-250ms$ around every reference boundary is usually used.

- *WER*:

  The Word Error Rate (WER) is kept as the primary evaluation metric for this evaluation of ASR. That metric basically counts the number of word deletions, insertions and substitutions in the output of the automatic transcription system compared to a reference transcription produced by humans. It can be computed as:

$$WER = \frac{S + D + I}{N} \tag{1.9}$$

  where S is the number of substitutions, D is the number of deletions, I is the number of insertions and N is the number of words in the reference transcription.

Chapter 2

LIUM toolkit

2.1 Introduction

In the first part of the work we have introduced the concept of Speaker Diarization from a series of theoretical sections that try to cover the most important points of the task. After showing the tools in which we will work and defined the objectives we want to achieve, in this chapter we will begin to perform the first experiments to understand the task in a more practical way. For this reason we wanted to start with a toolkit that has been often used as a baseline for comprison purposes [8, 21].

The software that we are going to use is known as LIUM_SpkDiarization [10]. This toolkit was developed by the computer science lab of the University of Le Mans in 2010. It is a software capable of carrying out the entire speaker diarization process. It is based on hierarchical agglomerative clustering methods using measures such as BIC (Eq. 1.1) or GLR (eq. 1.3) and also on Gaussian Mixture Models. For the evaluation of the toolkit and its comparison with the rest of the systems that are going to be tested in this work, the AMI corpus will be used [25].

This Chapter is organised as follows. First, in Section 2.2, the basic methodology is briefly discussed. Then, in Section 2.3, we focus on the system that LIUM uses. After that, there are four Sections covering different parts of the LIUM system. Then, empirical results are reported in Section 2.8 and the main conclusions drawn are given in Section 2.9.

2.2 Methodology

After the installation of the LIUM toolkit, several tests will be performed with different audios to observe the correct functioning of the system and familiarize with the format of the output file of the software.

We want to use the same form of evaluation for all the programs that we will see in the work, so we will use pyanote.metrics in all cases for the calculation of DER.

For that, python files will be implemented that would serve as calls to functions and thus not do so from the linux terminal.

With the whole system prepared, the different audios of the test will be segmented. First we will call the main function of the toolkit for obtaining a first segmentation. Then we will use the necessary scripts to obtain the feature vectors and carry out the Speaker Diarization process with LIUM that allows us to modify the different parameters of the software.

Once the output files of the system have been obtained, we will implement a file with python to evaluate the system and obtain the DER from the test set. After that, all subsequent tests with this set of tools will be repeated by the same steps described in this section and making only changes to the parameters of the scripts provided by LIUM.

2.3 Speaker Diarization with LIUM

The system implemented in the toolkit created by LIUM performs all the necessary pipeline to obtain an adequate final segmentation due to a speaker diarization system. For this, a series of functions are available that are responsible for performing the following steps: Calculation of the features of the audios, initial segmentation, initial clustering, second segmentation based on Viterbi, speech detection, detection of the genre and the band of the channel and a final clustering based on Gaussian mixture models. In the figure 2.1 we will see in greater detail each one of these parts [10].

2.4 Computation of features

The first steps of the system uses the feature vectors obtained in the module of feature extraction. To obtain this vector, the Sphinx 4 tool is used. This tool obtains a vector of features per frame from the audio that it takes by input. The output is composed by 13 Mel-frequency cepstral coefficients (MFCCs) with energy and they are not normalized. That is because at this point we need all the information of the audio signal and the normalization is used to remove the background sounds.

The MFCCs are usually derived by taking the Fourier transform of a signal and then applying another steps like mapping the power of the spectrum onto the Mel

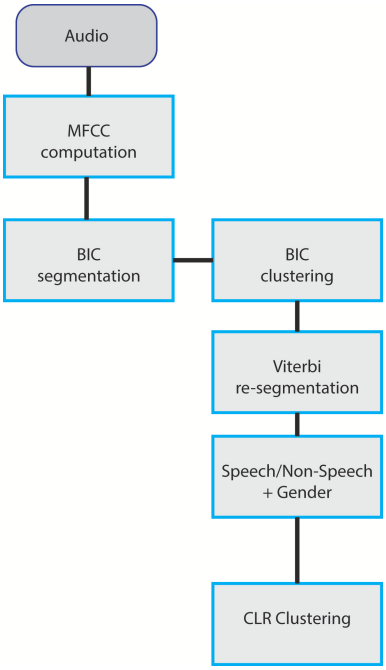

Figure 2.1: Speaker Diarization System with LIUM

scale or taking the discrete cosine transform of the list of Mel log powers.

2.5 Segmentation

A big part of the task of Speaker Diarization is the segmentation of the signal with the ultimate goal of subdividing it in fragments that each of them corresponds to a single speaker. In addition, clustering algorithms are also used to group segments that correspond to the same speaker and thus have a structure formed by a speaker per cluster.

Among the segmentation techniques implemented by the LIUM toolkit we have:

- *Segmentation based on BIC*

  From the features obtained in the previous step, a first segmentation of the audio signal is performed. For this, the toolkit uses an algorithm that detects the segment boundaries corresponding to the changes between speakers, treating the non-speech part as another speaker for now.

The audio is segmented taking into account the change points that are determined with GMM of complete covariance matrices. The entire signal is traversed using a window of 5 seconds and possible changes are detected using a generalized likelihood ratio (GLR) (see Sec. 1.8). This ratio is used to determine if in the middle of a window has occurred a change of speaker and if the value of comparison is above the ratio the fragment is segmented.

To complete this step, a second pass is made through the signal to group different consecutive segments of the same speaker. It is verified that they are from the same speaker by calculating ΔBIC.

- *BIC Clustering*

  Starting from the segmentation carried out in the previous step, another subsequent step is done where a first grouping of segments into clusters is made. Each cluster is modelled as a GMM with a complete covariance matrix and the input is initialized with a cluster for each segment. The algorithm is based on a hierarchical agglomerative clustering and the ΔBIC criterion is used to determine if it is necessary to group several segments in the same cluster or not. At the end of this step, the objective is to unify some segments of the same speaker into the same cluster.

- *Segmentation based on Viterbi decoding*

  The next step improves segmentation by Viterbi decoding. The clusters are modeled by hidden Markov models with a single state which is represented by a Gaussian mixture with 8 components and diagonal covariance matrix. The model is learned by an EM-ML algorithm using the segments of the cluster. This algorithm improve the model by maximizing the likelihood in each of their iterations. The log-penalty used to compare between two HMMs was fixed experimentally with the corpus ESTER 2 [26].

  To adjust the point where the segments are separated, a modification is made taking into account the areas of low energy and trying to obtain a segmentation avoiding fragmentation in the middle of words.

2.6 Speech and Gender detection

After the segmentations performed, a series of detection is carried out in the system implemented in the LIUM toolkit. The first one serves to locate non-speech areas,

an aspect necessary for the final evaluation of the result obtained. The second tries to extract more information from the signal and the speakers.

The detections that are made in the system are:

- *Speech detection*

  Up to this point the system treated the segments corresponding to the non-speech zones as if they corresponded to another speaker. The next step is to look for the non-speech segments and separate them from the rest. For this, a decoding by Viterbi is performed with 8 one-state HMMs like the described previously. The eight models consist of 2 of silence, 3 of wide band speech, 1 of narrow band speech, 1 of jingles and 1 of music. Each one is represented with a GMM of 64 diagonal components trained with EM-ML using the ESTER 1 corpus [12].

  At this point in the system the energy of the features is eliminated, since it is no longer necessary for the following steps.

- *Gender and Bandwith detection*

  Detection of gender and bandwidth is done using a GMM (with 128 diagonal components) for each of the 4 combinations of gender (male/female) and bandwidth (narrow/wide band). Each cluster is labeled according to the characteristics of the GMM which maximizes likelihood over the features of the cluster.

2.7 GMM-based speaker clustering

In the segmentation and clustering steps above, features were used unnormalized in order to preserve information on the background environment, which helps differentiating between speakers. Now we must perform a normalization to remove the information from the environment before performing a hierarchical agglomerative clustering to finally relate the speakers and the clusters one by one.

To adapt the mean of each cluster, a UBM (Universal Background Model) model is used, made up of the fusion of the GMMs used in the previous gender and bandwidth detection. In each iteration, if two clusters maximize the measure proposed by Cross Likelihood Ratio (CLR), and by Cross Entropy, they will be merged. It is due to the fact that both clusters probably have segments of the same speaker.

In this final part we obtain a final segmentation that will be compared with the annotated reference and helps us determine the performance of the toolkit.

2.8 Experiments

The LIUM toolkit was seen as a good basis for the study of the task of this work since it is based on systems that already gave good results before the introduction of neural networks in the area of speaker diarization. Once the entire installation process was completed, the familiarization process with the toolkit began. The main calls to this toolkit are made with functions implemented in java and the quickest way to use it is to call its main function directly. However, this function performs the whole process of speaker diarization, but does not allow too much variation to improve the system itself.

After verifying that the system was able to obtain a segmentation from an input audio, a proper corpus was searched for its evaluation. The AMI corpus [25] was chosen because it is accessible for free and because it has been used as a reference task in speaker diarization [8]. This corpus is a multi-modal dataset consisting of 100 hours of meeting recordings and 150 speakers in it. And it was separated in 115 files of training 20 files of development and 21 files of test.

Table 2.1: AMI corpus statistics

Dataset	**# speakers**	**# hours**	**# training**	**# dev.**	**# test**
AMI	150	100	115	20	21

The next step consists in searching a better way to carry out the speaker diarization process and for that, scripts were found available on the toolkit documentation web. These scripts implement the calls to the functions of each one of the parts of the system pipeline. In the following paragraphs we will explain the different implementations, with python, made to facilitate the control of the toolkit and a better explanation about the possibilities of the functions in the scripts of the LIUM toolkit.

As we said several programs were carried out in python for a greater control of the task and the subsequent evaluation through the library "pyannote.metrics" (A library of Speaker Diarization evaluation). Before explaining the different experiments performed with the toolkit, we will detail the processes performed by two of the main files implemented. The first one, "Segmenta_corpus_AMI", is responsible of making

convenient calls to the toolkit for its use. The second, "Evaluation_AMI", collects the output obtained by the LIUM software, and matches it to the structure necessary to carry out its evaluation.

- *Segmenta_corpus_AMI*

  One of the implemented files is "Segmenta_corpus_AMI.py", in which all the segmentation and clustering process is done using the LIUM toolkit. This file has several versions: the first calls the main function that perform the entire pipeline, obtaining a segmented file for each input audio and not allowing the modification of the default parameters. The second version is responsible for calling a script which uses all the necessary functions to perform each step of the task. This second version allows the modification of several parameters and is the one that has allowed a greater variation to minimize the error sought.

  In all the implemented files that use the AMI corpus, the "pyannote.database" library has been used. In that library the references of the corpus are stored and it offers a defined structure that separates the corpus data in training, development and test. In addition, this library helps organizing the data structure so that it is easier to manage the location of the corpus audios to prepare the input to the system.

  As mentioned previously, the second version of the program uses a script available in the toolkit. In this script, called "diarization.sh", the whole process of Speaker Diarization is done like in the case of using the main function. But in this case it uses different functions and allows the user to modify each one of the parameters that participate in the whole process.

  The system takes as input an audio and a series of parameters that indicate how the feature vectors should be structured. After doing the feature extraction with this information, the first segmentation is performed using the functions MsegInit, MDecode and MSeg. After that, the first clustering is performed with MClust and the models are trained for each Speaker using MTrainInit and MTrainEM. The next step is to perform a Viterbi decoding with MDecode and an adjustment from the lower energy zones with SAdjSeg (see Sec. 2.5).

  Taking the information of the first call to MDecode as part of the input, the non-Speech zones are searched and filtered with SFliter and the fragments of more than 20 seconds are segmented, to facilitate the subsequent clustering, with SSplitSeg. The next step is not significant in our case, since it try to label the audios in genre and audio band with MScore (see Sec. 2.6). To complete

the Speaker Diarization process a last clustering process is performed with the call to the MClust function (see Sec. 2.7).

Repeating the process for each of the test audios, we obtain several files with the necessary segmentations to make the comparison with the available reference and perform the evaluation of the system.

- *Evaluation_ AMI*

  For evaluating the segmentations obtained with the LIUM toolkit, another file called Evaluation_AMI was implemented. The python code that we find in it is divided into two parts: The part of parsing the files into the correct structure format and the evaluation part using pyannote.metrics.

  In the first part, the different output files obtained with the toolkit are parsed and storage in variables with the format implemented in pyannote.core. LIUM saves the segmentations made following a format close to the MDTM or STM NIST format:

  ```
  alzira20180327 1 11 389 F S U S0
  ```

  In which we can find the name of the file, the channel number, the start of the segment in frames, the lenght of the segment in frames, the speaker gender, the type of band, the type of environment and the speaker label.

  In contrast, in the structure used by pyannote we need to define a variable annotation with the label of the speaker. The variable is a cluster defined with the start and end time of the fragment as we can see in the Fig. 2.2.

```
In [1]: from pyannote.core import Annotation, Segment

In [2]: annotation = Annotation()
   ...: annotation[Segment(1, 5)] = 'Carol'
   ...: annotation[Segment(6, 8)] = 'Bob'
   ...: annotation[Segment(12, 18)] = 'Carol'
   ...: annotation[Segment(7, 20)] = 'Alice'
   ...:
```

Figure 2.2: Example of pyannote.core annotation

In the evaluation part of this file, the names of the audios of the test part of AMI are obtained using the pyannote.database library. Then the DER metric is calculated using pyannote.metrics and the system returns a report with the different errors of each one of the test files and the total obtained, which forms the results that we will interpret.

Once we have seen the files implemented in this chapter, we will show the different experiments carried out for the evaluation of the LIUM toolkit. At this point of the work was where one of the biggest problems was found. In [8] showed that using the default parameters of the toolkit, they obtained a DER of 25%, however in our case the first results were approximately 53% of DER.

The Table 2.2 shows the values obtained with the experiments that we are going to explain below.

Table 2.2: DER results with AMI test

LIUM toolkit results with AMI				
Test	false alarm	m. detection	confusion	DER
1	9,21	17,66	26,40	53,27
2	5,99	12,97	12,82	31,78
3	12,14	19,75	13,03	44,92
4	5,12	20,80	8,92	34,84
5	6,91	7,25	13,37	27,52
6	6,76	7,60	12,71	27,07

As we commented before, LIUM toolkit has a main function that can be called and it performs the whole process of speaker diarization. In the experiment 1 we use that form of calling the software. However, this method does not allow modification of the system parameters. In the file implemented for this purpose we only pass to the system the names of the audios using a loop and reading them from the AMI test part. This resulted in a series of files corresponding to the segmentations of each audio.

The evaluation of these files gave us a value of DER of 53%. We can see also that the values of confusion, 26.4, and miss detection, 16.66, are quite elevated. We aim for a value of 25% so we needed to improve the results but we couldn't modify the system by this way of processing the signals of audio.

At this point we changed the form of calling the software and we started to use a script that allows us to see every function needed to do the entire pipeline of the Speaker Diarization process. The experiment 2 was carried out without any modification of this script.

The result obtained was a DER of 31%. This was a good improvement of the first result obtained but it was still far from the desired result. We can see in the Table 2.2 that in this test the false alarm is one of the lowest values of all the experiments.

Analyzing the error produced by the system, it was seen that an important part of the error was found in the speaker recognition part of the system. To improve this part, the parameter relative to the components of the GMM used for the clustering was changed. First a value of 128 components was used (experiment 3), and then a value of 12 components (experiment 4).

The DER increased considerably to a 44,9% error in the experiment 3 and not so much for the experiment 4 that gave us a value of 34,8% of DER. These two values were greater than that obtained with the second experiment, so we had to find another way to improve the result.

After carrying out another verification segmentation and obtaining the same result, a possible solution was searched through the network. Navigating between articles and forums of people who used the same toolkit, a user was discovered which indicated that the LIUM toolkit had a problem when the speakers changed briefly in a few seconds [2]. This was precisely the problem of the AMI corpus, so the suggestion of the user was tested and the size of the initial segmentation window was decreased.

At the end we find that the problem occurs in the precision of the cut between fragments of different speakers in the initial segmentation. To solve this, the window size was changed to 100 ms. and the GMM configurations of the two experiments with the best results so far were also projected, experiments 5 and 6. With this, values of 27,5% and 27,1% of DER were obtained. That values were close to the 25% of DER searched, so we saw that this toolkit obtain good results despite the fact that the AMI corpus is complicated.

2.9 Conclusions

After becoming familiar with the main concepts of the task of Speaker Diarization, we went to use a toolkit with base results to establish the starting point of the practical part of the work. We saw that this toolkit, known as LIUM, was based on Gaussian mixtures models, hidden models of Markov and Hierarchical agglomerative clustering. And we also tried to obtain a similar result to 25% of DER that was established in [8] using the AMI corpus.

To find the value that was wanted to use as a reference, we implement several files that would serve to make the experiments repetitively adequate. In the article the authors only said that they had used the LIUM standard parameters, so the tests

performed consisted of several tests that were approaching the value sought.

In the end we managed to reduce the error to 27% of DER. With the process of achieving this result we observe that the error occurred in the accuracy of the segmentation of the signal and this increased the values of false alarm and miss detection in the calculation of the DER (Eq. 1.8. We correct this error by changing the size of the window that crosses the signal in the first segmentation that is made in the LIUM Speaker Diarization process.

Chapter 3

Toolkit Pyannote.audio

3.1 Introduction

After studying the LIUM toolkit in Chapter 2, where a good result was obtained to use as a basis for the subsequent experiments, we wanted to use another toolkit that was closer to the current state-of-the-art and based on recurrent neural networks. For that we decide to use a toolkit that is currently being developed by Hervé Bredin and is know as Pyannote [5]. Among all the tools covered by this software we can find pyannote.audio. This tool was created in 2017 and has a tutorial to make speaker diarization with the AMI corpus.

In addition to this tool, Pyannote has other tools that are necessary in various parts of the Speaker Diarization system. To save and use the various corpus, we will use pyannote.database. This library helps to create a structure using the elements implemented in pyannote.core that serves to facilitate access to corpus information. We can also find the tool pyannote.metrics [6], which we use for the evaluation of all systems made in this work.

The organization of this Chapter starts in the Section 3.2 by defining the methodology that will be followed in this part of the work. It continue in the Section 3.3, and the following five Sections, explaining the development carried out in each of the parts that make up the entire Speaker Diarization system implemented in the pyannote.audio library. After that, in Section 3.9 it is shown how the different experiments have been developed and how we have solved the complications that have encountered. In the Section 3.10, after showing the results obtained in the evaluation of the experiments, we present the conclusions we have reached.

3.2 Methodology

The pyannote toolkit has many different libraries to help with the speaker Diarization process. The pyannote.audio library is in charge of training the network, the pyannote.database library is responsible for structuring the corpora using the pyan-

note.core structures and the pyannote.metrics library serves to evaluate the results.

The way in which a complete system of speaker diarization is built in this Chapter is to large extent inspired by a tutorial on pyannote.audio that can be found in [5]. Except for the feature extraction part, the rest of the parts are structured in: preparation of the corpus, training of the network, validation of the network and testing of the obtained files. The last part, which tunes the parameters of each step and gathers the entire pipeline, is in charge of obtaining the final segmentation of the signals of audio. We will evaluate the DER from the output of the entire Speaker Diarization system.

In this chapter various experiments will be carried out in each of the pipeline parts. However, several experiments will also be carried out repeating the same methodology explained in this section.

3.3 Speaker Diarization with artificial neural networks

The latest studies concerning the area of speaker diarization are based on the use of neural networks for the training of a system capable of performing the entire process that gives us the segmentation of audio labeled with his speakers.

The pyannote toolkit tool that is responsible for this task is based on different systems that perform four main steps and then join them. These four steps cover the different parts of Speaker Diarization described so far and are the following: Computation of features, Speech activity detection, Speech Change Detection and the obtaining of Speaker Embeddings. The last three parts are based on neural recurrent layers of LSTMs. The entire pipeline is explained with more detail in the Section 3.8.

3.4 Computation of features

The first step in a typical process of speaker diarization is the feature extraction. The pyannote toolkit relies on the toolbox Yaafe, an audio features extractor. The current version is from 2011, but its feature computation is already reliable. This toolbox transform an audio in a group of MFCCs using a 25ms-long sliding windows on steps of 10ms. The MFCCs extracted are formed by the energy first derivative,

the energy second derivative, 19 coefficients, their first derivatives and their second derivatives. Then they are normalized using short term standardization with a window of 3 seconds.

This first part of the pipeline is responsible for transforming the audio into a series of feature vectors using but the toolkit is already responsible for making the call to Yaafe using pyannote-speech-feature and it is only necessary to modify a configuration file called "config.yml" to specify the details of the MFCCs that are to be obtained.

This step has only been done once and its results have been used in the various experiments carried out. To do this, the feature vectors are stored in a folder that will serve as an input to the next system steps.

3.5 Speech activity detection

This part of the speaker diarization process aims, first, to find the speech parts of the audio and differentiate them from the parts of background noise, music or silence, and second to detect the edges where these different fragments are separated, as well as the speaking changes between different speakers, as accurately as possible.

For both speech activity detection and speech change detection, systems based on LSTMs have shown a good result. For this reason, the toolkit proposes the use of bi-LSTMs to predict whether or not a frame corresponds to a specific region of speech [36]. The architecture used in this part is proposed as a problem of labeling sequences where binary labeling is sought, being 1 if there is a speaker change and 0 when there is not. Therefore, the goal is to find a function f that relates a sequence of features to the tagged sequence. The formula for finding f is based on the binary cross-entropy and we can see it in Eq. 3.1.

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

where x is a feature extracted from the audio, y is the binary label of that feature, and T is the total number of features extracted. We can see in Fig. 3.1 the type of annotation that is given in the reference to train the network in the part of speaker change detection.

The proposed Speaker Diarization process starts by detecting the speech zones

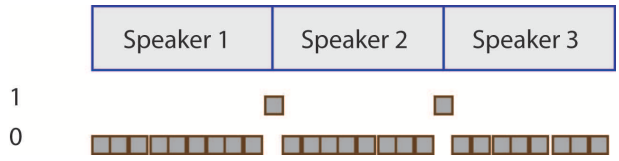

Figure 3.1: Reference annotation for SCD

and the no-speech zones [36]. For this, the pyannote-speech-detection function of the toolkit is used. First of all we have to configure the system to train a network with bidirectional LSTMs that will use the features obtained in the previous step and the references that the system has saved in pyannote.database. Then is time to pass to the training. The network was trained doing 1000 iterations, as indicated in the tutorial. However, it was thought that the result could improve with more iterations, and they were changed to evaluate their behaviour with different number of iterations. We can see in Fig. 3.2 an example of the error values that have been obtained in the training part of one of the experiments carried out.

Figure 3.2: SAD training error by iteration

At the same time, a validation was carried out using the development part of the corpus. In this part we can see a result more approximate to the result obtained in the test part since these audios have not been used in training. Once the network is trained, the results of the validation are observed to find the iteration in which the system has given the best result. In Fig. 3.3 we can see the results obtained in the validation part of the best experiment carried out.

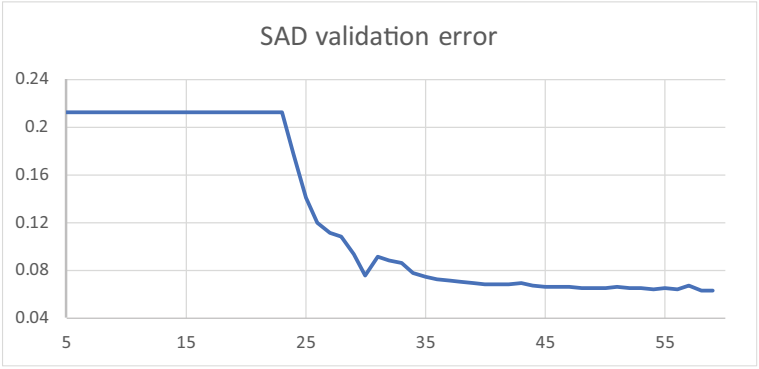

Figure 3.3: SAD validation error by iteration

For the evaluation part of the initial segmentation obtained in this part of speech acivity detection, a file in Python called Evaluation_AMI_pyannAudio_SAD.ipynb was created in which the obtained scores were loaded, binarized to obtain the speech regions and evaluated using the DER. The results of this evaluation can be seen in the Section 3.9.

3.6 Speech change detection

The second part of the system is the task entrusted to perform the speech change detection. The architecture used in this part is composed by two Bi-LSTMs and a multi-layer perceptron (MLP) whose weights are shared across the sequence. This MLP is made of three fully connected feedforward layers, using tanh activation function and sigmoid for the last layer.

One of the important parts in a Speaker Diarization system is the precise detection of each fragment in which the audio will be segmented. In this part, using the pyannote-change-detection function, another artificial neural network is trained to perform the speech change detection [36]. The network is configured so that the duration of the subsequence is 3.2 seconds, also 10 hours of audio are used per epoch and each batch has 32 sub-sequences. The network uses the StackedRNN architecture and is composed of bidirectional LSTMs.

The process followed to train the network is similar to the one done in the previous section. With pyannote.database the system is already prepared to find the data of the AMI corpus and we simply have to pass the corresponding protocol to the functions. In this case the AMI.SpeakerDiarization.MixHeadset protocol. We

start the process training the recurrent network while the validation is carried out in parallel. In Fig. 3.4 we can see the error obtained in the different iterations of the training. After that, once we obtain the iteration with the best result, the evaluation is carried out using Evaluation_AMI_pyannAudio_SCD.ipynb.

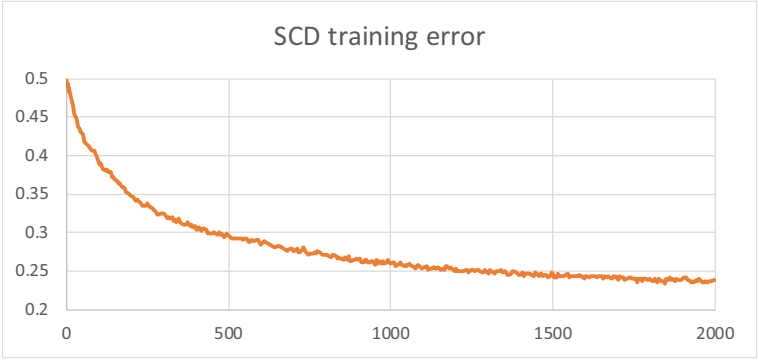

Figure 3.4: SCD train error by iteration

In the validation part of the speaker change detection in pyannote.audio the systems try to improve the coverage respect a defined value of purity. By default the value of purity is 80%, but we will change it due to the improve of the system. We have an example of the improvement of the coverage in Fig. 3.5.

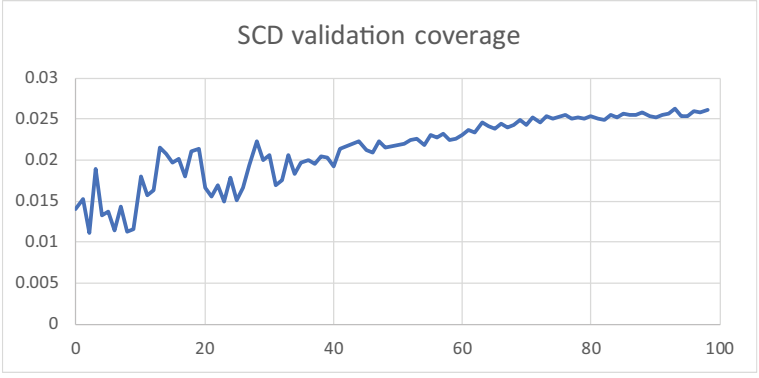

Figure 3.5: SCD validation coverage by iteration

3.7 Speaker embbedings

In this part, the main objective is to train a speech sequence embeddings based on recurrent neural networks to get closer to the optimal function that achieve the best comparison on speaker recognition.

For the training part, a triplet loss comparison is done [7]. It consist in a triplet composed with features of a speaker (anchor), features of another sequence of the same speaker (positive) and the last element is formed by a different sequence of a different speaker (negative). The network tries to maximize the value of a searched formula in case of the two values of the same speaker (comparison between the anchor and the positive) and minimize it if the comparison is between different speakers (comparison between the anchor and the negative). The distance is calculated using the Eq. 3.2.

$$\Delta_\tau = ||f(x_a^\tau) - f(x_p^\tau)||_2^2 - ||f(x_a^\tau) - f(x_n^\tau)||_2^2 \tag{3.2}$$

And the lost that we try to maximize is defined by the Eq. 3.3.

$$\mathscr{L}(\mathscr{T}) = \sum_{\tau \in \mathscr{T}} max(0, \Delta_\tau + \alpha) \tag{3.3}$$

In the formulas above, τ is a possible triplet, $\mathscr{T}$ is the number of all possible triplets and α is a safety margin. We can see an scheme of the triplet sampling in Fig. 3.6.

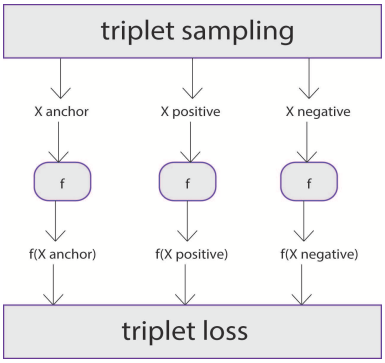

Figure 3.6: Triplet loss training

The first steps made in the speaker Diarization system proposed by Pyannote have allowed to find the parts of speech and in them to detect the parts of changes between speakers. Finally, this last part aims to create the speech turn embbedings. The architecture used in this part is called TristouNet and it is based on training a network using as a loss function triplets (Eq. 3.2. The margin α used in the system is 0.2 and the network consists of recurring bidirectional LSTMs. In this step we started a training with 1000 iterations calling the pyannote-speaker-embedding function, but later it was varied to improve the result obtained in the validation. The error that we have obtained in the training of one of the experiments can be seen in the figure 3.7.

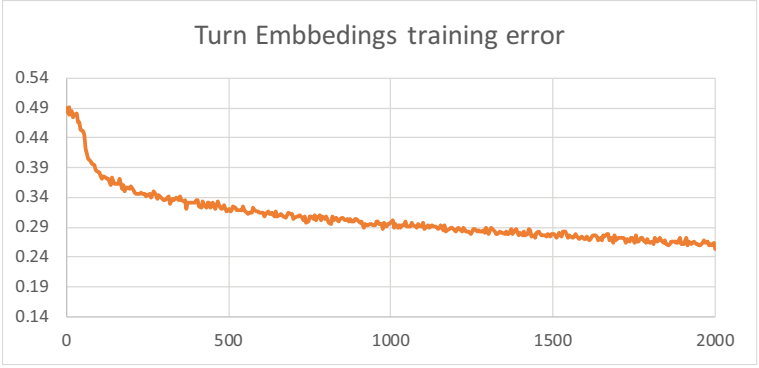

Figure 3.7: Speech turn embeddings train error by iteration

This part does not have a test evaluation method, we can simply observe the different embeddings obtained. These are used in the next step when the whole system is joined and the entire network is applied to the test part of the corpus. A form of evaluate this part of the system is follow the value obtained in the validation of the network (see Fig. 3.8).

3.8 Pipeline

The steps described above make up the whole system of speaker diarization, however it is necessary to put them together to create the whole process from start to finish and optimize its parameters. All the steps were based on recurrent neural networks,

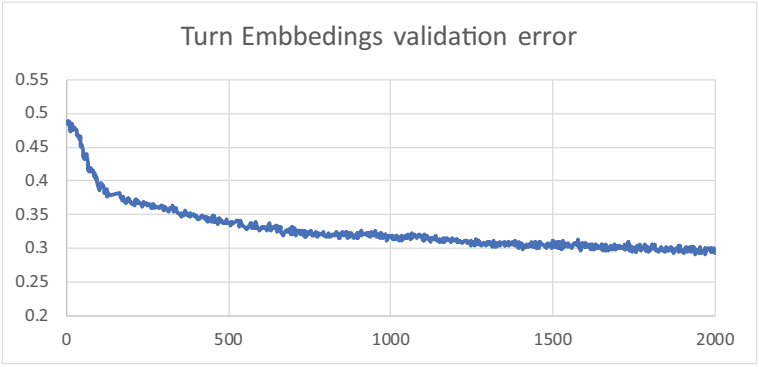

Figure 3.8: Speech turn embeddings validation error by iteration

except the final clustering after obtaining the embbedings. In Fig. 3.9 we can see an outline of the whole pipeline [37].

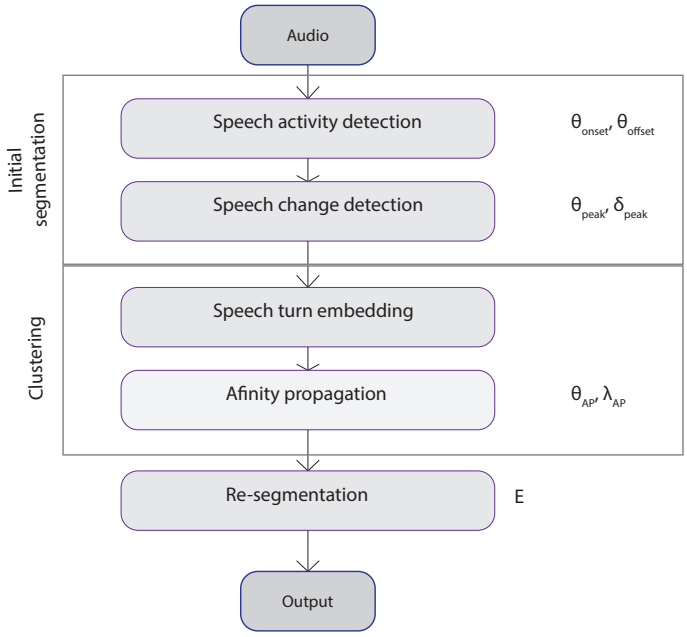

Figure 3.9: Speaker Diarization pipeline with pyannote.audio

The system of speaker diarization proposed by Pyannote is composed of two first parts, speech activity detection and speech change detection, that aim to obtain a homogeneous segmentation per speaker. This two parts are modelled as a classifica-

tion task of an improved binary system with deep learning approaches. After that the system performs a speech turn embedding part and a clustering based on affinity propagation. The output of the system goes through a re-segmentation that serves to refine the edges of the obtained segments.

At this point in the system of speaker diarization proposed it is only necessary to join the entire pipeline using the saved results of the previous four steps. For this, the pyannote-pipeline function is responsible for completing the process and tuning the parameters of each part of the system. For doing that the first step is to provide a configuration file with the path to the results of applying the networks in speech activity detection, speaker change detection and speaker embeddings. After that, the function is called to train the network modifying each one of the parameters that we can see in the scheme shown before (see Fig. 3.9). In this step, the system uses the train and development data to obtain the best result by performing iterations until the user stops it.

At the end the pyannote-pipeline function is applied, but using the "apply" parameter, to perform the speaker diarization segmentation. For that it was necessary to implement a file called Evaluation_AMI_pyannoteAud_v2.ipynb to evaluate the system in the same way as have been done with the LIUM toolkit, using pyannote.metrics to get the DER.

3.9 Experiments

Observing the pyannote documentation for doing the evaluation part of the LIUM toolkit with pyannote.metrics, a tool was found that carried out speaker diarization with recurrent neural networks and was also in the toolkit, pyannote.audio. In addition, this tool had a tutorial to perform every part of a speaker diarization system with the AMI corpus, which was very consistent with the work that was being done.

As we explain in the previous chapter, the AMI corpus [25] is a multi-modal dataset consisting of 100 hours of meeting recordings and have 150 speakers in it. Its references are stored in pyannote.database separated in 115 files of training, 20 files of development and 21 files of test. (see Table 2.1)

The tool pyannote.audio has been designed to perform Speaker Diarization using artificial neural networks. For this, it has different functions that serve to carry out the training, development and testing of each of the parts in which the developers have seen fit to separate the entire pipeline (see Fig. 3.9).

To carry out the whole process of speaker diarization the toolkit is separated into several parts, as well as the tutorial. After the extraction of features (see Sec. 3.4), the first part of the system performs speech activity detection (see Sec. 3.5), where the speech audio and the parts of silence, background sound or music are detected. The second part looks for the points of change between speakers or non-speech parts (see Sec. 3.6). The third part of the system is responsible for obtaining embeddings (see Sec. 3.7) and the last one merge the whole system and obtains the final segmentation (see Sec. 3.8). Although the tutorial seemed to perfectly comply with what was sought, it had several problems.

We contributed to the pyannote.audio library by sending to the author the information about the errors that we found when performing the tutorial. A problem in the code occurred when the evaluation file wanted to load the files obtained when applying any of the segmentations. When the scores were saved, it called the "precomputed" function that theoretically should return the scores. However, the library used to obtain the path no longer returned it with the same format and therefore gave an error. To solve it, first an implementation was done to obtain the path, then we load the data with numpy and we call directly the internal function that returned the scores.

As shown in the previous sections, the pyannote.audio toolkit performs different parts of the speaker diarization process separately and then brings them together in a single system. Some of these parts allow to be evaluated before performing the application of the complete system and the segmentation of the audios in the AMI corpus test part. Although the evaluation of the DER can only be done in the last part, an evaluation of each part is necessary to choose the best composition of the network that will be later chosen and joined. We must also have to say that the parameters we observe in the following tables are tuned in the part in which the pipeline is joined, but they help us with the choice of the best partial result.

In the tutorial the results that should be obtained at the end of each of the parts were not indicated and therefore a series of experiments was necessary to evaluate the values that could be obtained. In Table 3.1 we can see the different tests performed in the speech activity detection part. The parameters that we have modified in the training of the network were the number of iterations, the onset and the offset.

The iterations shown in the table have three different values, the tutorial indicated that the value to use was 50, 1999 is the value corresponding to the last iteration and the value of 203 iterations corresponds to the iteration with the best result obtained in the development task of the network. We observe in the table that the detection error

Table 3.1: SAD tests with AMI

Speech Activity Detection				
Test	Iterations	onset	offset	Detection error rate
1	50	0,25	0,25	8,9
2	293	0,25	0,25	6,8
3	1999	0,25	0,25	7,2
4	50	0,35	0,25	8,2
5	293	0,35	0,25	6,6
6	1999	0,35	0,25	7
7	50	0,35	0,35	7,7
8	293	0,35	0,35	**6,3**
9	1999	0,35	0,35	6,8

rate has given a better result for an onset and an offset of 0.35, with 293 iterations of the network.

The values to improve in the speech change detection part are purity and coverage. The two values affect the calculation of the DER in the final step and therefore an trade-off between them is sought. In the Table 3.2 we can see the results for the different tests.

Table 3.2: SCD tests with AMI

Speech Change Detection				
Test	Iterations	alpha	Purity	Coverage
1	50	0,5	38,6	100
2	1821	0,5	74	7,8
3	1999	0,5	**74,2**	**7,9**
4	50	0,2	40,4	86,7
5	1821	0,2	79,6	3,2
6	1999	0,2	79,6	3,2

We have tried to get the maximum value of purity but at the same time tried to maximize the coverage. In the end it was considered that the best option was given in test 3, with a purity of 74,2% and a coverage of 7,9%.

The last step that is carried out with pyannote.audio, that allows us the comparison with the toolkit of LIUM, is to segment the audios with the whole process. First SAD, then SCD and finally the obtaining of the embedings, but this time the system connects all the steps in the training stage and then the user passes the audios through the network.

The tutorial did not indicate which was the target DER that was sought, but it was expected to obtain a value better or close to the 27% achieved with the LIUM. The values of diarization error rate (DER) obtained can be seen in the table 3.3.

Table 3.3: DER results with AMI test

Speaker Diarization with pyannote.audio						
Test	Iterations	correct	false alarm	m. detection	confusion	DER
1	1000	42,59	2,97	6,14	51,27	60,38
2	2000	58,96	4,82	2,74	38,30	45,86
3	2000	59,06	2,48	6,31	34,64	43,42
4	4000	**61,59**	2,85	4,87	33,54	**41,25**

The experiment 1 was carried out using exactly the same values that the tutorial told us. That was training each part using 100 iterations but applying the networks using the weights of the iteration number 50. This gave a DER value of 60,38%, which was far from the expected DER. It seems that the tutorial was in the process of development and the indicated parameters corresponded to the corpus used in the articles [36] and not to the AMI corpus.

After that, in experiment 2, it was decided to increase the number of iterations of the system due to the fact the result of the previous experiment show that it was necessary a greater training of the networks. Then we apply each step of the system using the saved weights of the last training iterations. This improved the DER to 45,86%, still far from the value sought.

In the SCD part of the system (see Sec. 3.6), the function used for the validation of the network obtains the maximum value of coverage from a value of purity modified by the user. In experiment 3, we repeat the whole process done in the experiment 2 but changing this value from 80% to 70%. This allowed the DER to improve to 43,17%.

The final test, the experiment 4, consisted of repeating the whole process of speaker diarization, with 70% purity in the SCD part, but this time using the trained networks of the experiments with the best results observed in the Table 3.1 and the Table 3.2. This improved the value of DER until obtaining a value of 41.25%. We also observed that the correct part of the test is 61.594%.

Despite the fact that the the false alarm and the miss detection values are quite low, the value given in the calculation of the confusion makes the final DER quite distant from the 27% obtained with LIUM. That can be because the last part of the

pipeline, the speech turn embedding (see Sec. 3.7), does not have good results with corpus with fast changes of speaker and overlap between speakers.

3.10 Conclusions

Nowadays, artificial neural networks are being used in an immense amount of research areas. In the task of Speaker Diarization they are also being used and in this chapter we have seen an example of software that used deep learning. The toolkit created by Pyannote has allowed us not only to perform segmentation of audio signals from the AMI corpus through the pyannote.audio tool, it also allowed us to structure the corpus with pyannote.database and evaluate both this system and the Chapter 2 with pyannote.metrics.

In order to obtain the entire pipeline, it was necessary to train each network of each part of the process separately and then unite the entire system. These parts were the feature extraction, speech activity detection, speech change detection and speaker turns embeddings. By following the tutorial of pyannote.audio we saw that it was still in development, since it was not until the end of its use when the complete documentation was available and we also found some errors in the provided code that had to be corrected.

Once we got the system working completely, the AMI corpus was used to follow the tutorial and the results were improved until obtaining a 41.25% of DER. This error was somewhat high, although it was seen that it was in the confusion part of the error calculation that it raised the value up to that point. Therefore, we conclude that in the last part of the system, where the embeddings of the speakers are obtained and the last clustering is done, the system still had to improve for the AMI corpus. A corpus with many brief speaker changes and various overlaps of different speakers.

Chapter 4

Evaluation of the corpus Video Actas

4.1 Introduction

In previous chapters we have been able to test different software, which has led us to have a greater knowledge of how to implement a Speaker Diarization system and the problems that may appear in its development. For further study of this speaker diarization systems, we intend to use a public-sector corpus that allow us to apply the processes implemented so far in a real task.

On one hand, after a collaborative search between the research group, a website was found where video records of plenary sessions of different town halls are available. The web [31] is accessible to any user and among all its resources it has the audio of the plenary sessions and the references with the speakers' turns. On the other hand, the software developed by LIUM has shown results close to the expected in the corpus used so far, the corpus AMI. Thus, in these experiments carried out in this part of the work these two elements will be used together. We will use the system of speaker diarization proposed in the LIUM toolkit with the Video Actas corpus.

This chapter will show the development followed to extract and parse the necessary information from the web, its segmentation using the LIUM toolkit and the evaluation of a selection of some data from Video Actas for test.

4.2 Experiments

There are few corpora in the network with a good labeling that allows to perform a task of Speaker Diarization on it. Most of the articles that talk about this area use their own corpus or payment corpus. In the first parts of this work, a free access corpus called AMI has been used, however, for this chapter we searched for a corpus related to public domain administrations.

The corpus that will be used in this chapter will be called Video Actas. It is composed by data obtained from a website where the plenary sessions of different

town halls are stored [31] and in Table 4.1 it is described the amount of data that we used for test.

Table 4.1: Video Actas corpus statistics

Dataset	**# cities**	**# hours**	**# test files**	**# test cities**
Video Actas	27	2461	41	5

The first necessary step to be able to use this corpus was the realization of web scraping to download the audio files and the references of the corpus. Due to this the plenary sessions available until July 2018 on the web were stored.

The next step was to select a group of files that would serve as a test of the system. For this, data were selected from Blanes, Pego, Catarroja, Teruel and Alzira. After that it was necessary to parse the documents obtained with the web scraping to the format that we will use with pyannote.metrics for the evaluation of this test data. And also the conversion of the audios from m4a to wav. For these last two steps, two files were implemented in python to perform the tasks automatically, parseVAtoPyann.ipynb and towav_videoact.ipynb.

When performing the pairing we found a problem with this corpus. The way to record the turn of the speakers is not precise and does not inform when a speaker has finished spoken, only when the next speaker starts. This causes it to be a very noisy corpus in that its references are not well targeted and do not help to observe the non-speech fragments well.

Finally, all the test files were segmented with the LIUM toolkit, using the parameters of the best test performed with the AMI corpus (see Sec. 2.8) applying the implemented file LIUM_videoact.ipynb. We have the Table 4.2 in which we can appreciate all the results of this process.

As mentioned in before, the Video Actas corpus have a not very good references and we will see that the DER results are somewhat high due to the confusion factor of the formula. We can see in the Table 4.2, that the DER obtained was 40,65%. This DER is quite different from the one obtained with the AMI corpus that was 27%. We also observe that what most affects the value of DER obtained is confusion. This is partly due to the error noted in the reference. By having more non-speech parts targeted as if they were speech zones, this makes false alarms very low (non-speech zones detected as speech) and that confusion is high (area that is detected as speech but is not known if it correspond to the same speaker).

4.3 Conclusions

To get closer to a real task, we obtained a corpus from a public domain website where the plenary sessions of different town halls are kept, the corpus Video Actas. After downloading the information by web-scraping, references were parsed to the format used throughout the work for the different evaluations. In addition, the audio files were converted to the format used by the software developed by LIUM. This toolkit was used to perform the Speaker Diarization process in this chapter.

The results obtained when evaluating the output segmentation of LIUM were not very positive. The DER obtained was 40,65%, a very high value. The conclusion from the interpretation of the data was that the error occurred because the references of the corpus were not entirely correct. In the corpus only the information regarding the beginning of the turn of each speaker was available, but the non-speech zones were not well targeted. This entailed that the value of the calculated error was not significant with the real result of the system and that in these cases it may be a good idea to apply a known speech activity detection system and modify the reference.

Table 4.2: LIUM tests with Video Actes

File	false alarm	missed detection	confusion	DER
blanes20180409	11,01	7,08	12,44	30,53
blanes20180531	0,86	14,99	29,23	45,07
blanes20161027	1,55	4,42	29,72	35,69
catarroja20171005	1,14	8,78	28,41	38,33
teruel20180314	2,23	6,30	19,37	27,90
catarroja20180426	0,03	17,19	29,80	47,03
teruel20180404	3,20	4,66	29,48	37,34
blanes20170126	0,56	12,58	25,86	39,00
alzira20180530	0,13	3,05	30,69	33,88
blanes20180222	1,14	18,75	29,43	49,32
blanes20170306	1,87	5,85	23,67	31,38
blanes20160721	2,34	12,14	28,33	42,81
blanes20171130	0,97	8,48	31,21	40,65
catarroja20180222	0,15	19,94	28,46	48,55
blanes20161213	0,98	22,22	20,66	43,86
catarroja20170525	0,85	23,09	26,86	50,81
catarroja20171130	0,32	19,68	28,04	48,04
blanes20170427	1,60	6,33	30,95	38,88
blanes20160630	0,94	12,96	28,89	42,80
blanes20160929	1,62	4,69	31,55	37,86
blanes20170223	1,00	12,28	27,56	40,84
blanes20160817	1,28	10,48	27,55	39,31
blanes20180426	0,57	16,09	28,53	45,19
blanes20170525	1,88	5,13	28,19	35,21
teruel20180305	17,22	4,59	26,38	48,19
pego20171102	2,44	8,33	20,28	31,05
catarroja20170629	3,71	9,04	30,90	43,66
blanes20171004	19,15	5,81	32,83	57,79
alzira20180327	0,54	0,92	31,80	33,25
catarroja20171026	0,42	25,60	24,92	50,93
catarroja20171115	3,82	19,09	27,11	50,02
catarroja20170619	2,73	14,02	27,85	44,60
blanes20171117	1,36	5,75	17,65	24,77
blanes20171228	2,00	6,75	31,01	39,77
catarroja20170427	0,43	8,38	31,45	40,26
catarroja20170914	4,76	36,70	20,13	61,59
catarroja20170727	0,70	23,94	28,94	53,58
alzira20180502	1,29	2,08	22,77	26,13
alzira20180425	0,87	1,73	29,97	32,57
blanes20161124	1,55	4,42	29,72	35,69
teruel20180507	7,80	22,18	21,25	51,23
Total	1,29	8,78	28,41	40,65

Chapter 5

Conclusions

Throughout the work done we have achieved the objectives that we defined at the beginning of the approach. We have managed in becoming familiar and understanding the task of Speaker Diarization. Also we have apply two different systems, one of them close to the state-of-the-art, to the AMI corpus. Another objective that we achieved was improving the results obtained with the systems throughout the different experiments carried out. The last objective was to find a corpus related to the public sphere, and for that we used a corpus created from the information on the Video Actas website.

The first tool used to obtain the segmentation that is sought in a speaker diarization system has been LIUM. This tool has allowed us to follow a whole pipeline based on hidden Markov models and Gaussian mixture models. With it, a DER of 27,07% was obtained. This result has been close to the 25% reference of other authors who used the same toolkit, but it has been difficult to achieve due to the brief explanations about the process followed.

The other tool used has been pyannote.audio. This tool uses an architecture close to the state-of-the-art with recurrent neural networks for different steps. It has been seen that in the parts of Speech activity detection and Speech change detection the software work better than the LIUM tool, with a false alarm of 2,8% and a miss detection of 4,8% in pyannote with respect to a false alarm of 6,7% and a miss detection of 7,6% in LIUM. However, the final DER was 41,25% due to the confusion. We can say then that in this developing system there is a need to improve the part of the final clustering in order to obtain a better result. This may be due to the fact that the AMI corpus has many brief changes and in these cases the previous systems continue to give good results.

Regarding the corpus Video Actas it can be said that the speaker diarization obtains a acceptable result, nevertheless it is necessary to have a good reference to be able to correctly evaluate the obtained result. It is necessary to emphasize that if automatic systems are wanted to be used in a near future to extract information about public data, all the turns of each speaker and the interventions of the rest of the attendees in the plenary sessions must be taken into account.

Bibliography

[1] Nathional institute of standards and technology - www.nist.gov, 2009.

[2] Phone conversation diarization with lium - dsp.stackexchange.com, 2017.

[3] Leigh D. Alsteris and Kuldip K. Paliwal. Short-time phase spectrum in speech processing: A review and some experimental results. *Digital Signal Processing*, 17(3):578 – 616, 2007.

[4] X. Bost, G. Linarès, and S. Gueye. Audiovisual speaker diarization of tv series. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4799–4803, April 2015.

[5] Hervé Bredin. Pyannote multimedia processing - pyannote.github.io.

[6] Hervé 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.

[7] Hervé Bredin. TristouNet: Triplet Loss for Speaker Turn Embedding. In *42nd IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017*, 2017.

[8] P. Cyrta, T. Trzciński, and W. Stokowiec. Speaker Diarization using Deep Recurrent Convolutional Neural Networks for Speaker Embeddings. *ArXiv e-prints*, August 2017.

[9] Benoit Mathieu *et al.* Yaafe, an easy to use and efficient audio feature extraction software. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 441–446, Utrecht, The Netherlands, August 9-13 2010. `http://ismir2010.ismir.net/proceedings/ismir2010-75.pdf`.

[10] M. Rouvier *et al.* An Open-source State-of-the-art Toolbox for Broadcast News Diarization. Lyon, France, August 2013.

[11] Martín Abadi *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[12] S. Galliano *et al.* Corpus description of the ester evaluation campaign for the rich transcription of french broadcast news. In *In Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC 2006*, pages 315–320, 2006.

[13] Thomas Kluyver *et al.* Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[14] Xavier Anguera Miró *et al.* Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:356–370, 2012.

[15] 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*, 17(5):985–993, July 2009.

[16] Rashmi Gangadharaiah, Balakrishnan Narayanaswamy, and Narayanaswamy Balakrishnan. A novel method for two-speaker segmentation. 01 2004.

[17] H. Gish, M. . Siu, and R. Rohlicek. Segregation of speakers for speech recognition and speaker identification. In *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 873–876 vol.2, April 1991.

[18] Kyu Jeong Han and Shrikanth Narayanan. Agglomerative hierarchical speaker clustering using incremental gaussian mixture cluster modeling. In *INTERSPEECH*, 2008.

[19] Qin Jin, Kornel Laskowski, Tanja Schultz, and Alex Waibel. Speaker segmentation and clustering in meetings. In *In Proceedings of the 8th International Conference on Spoken Language Processing, Jeju Island, Korea*, 2004.

[20] Sukhvinder Kaur, J. S. Sohal, and Indira Gujral. Speech activity detection and its evaluation in speaker diarization system. 2017.

[21] Eva Kiktova and Jozef Juhar. Comparison of diarization tools for building speaker database. 13:314–319, 11 2015.

[22] Akshay Kumar and Anurendra Kumar. Unsupervised speaker diarization.

[23] Lie Lu, Hong-Jiang Zhang, and Hao Jiang. Content analysis for audio classification and segmentation. *IEEE Transactions on Speech and Audio Processing*, 10(7):504–516, Oct 2002.

[24] S. Madikeri and H. Bourlard. Kl-hmm based speaker diarization system for meetings. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4435–4439, April 2015.

[25] Iain *et al.* Mccowan. The ami meeting corpus. 01 2005.

[26] S. Meignier and T. Merlin. LIUM SpkDiarization: An Open Source Toolkit For Diarization. In *Proc. CMU SPUD Workshop*, Dallas (Texas, USA), March 2010.

[27] V. Subba Ramaiah and R. Rajeswara Rao. Speaker diarization system using hxlps and deep neural network. *Alexandria Engineering Journal*, 57(1):255 – 266, 2018.

[28] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[29] Md Sahidullah and Goutam Saha. Comparison of speech activity detection techniques for speaker recognition. 10 2012.

[30] Zia *et al.* Saquib. A survey on automatic speaker recognition systems. 123:134–145, 01 2010.

[31] Ambiser Innovaciones S.L. Sistema de gestión de actas municipales - www.videoacta.es.

[32] Pulkit Verma and Pradip K. Das. i-vectors in speech processing applications: a survey. *International Journal of Speech Technology*, 18(4):529–546, Dec 2015.

[33] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopez Moreno. Speaker diarization with lstm. *arXiv*, 2018.

[34] S. H. Yella, A. Stolcke, and M. Slaney. Artificial neural network features for speaker diarization. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 402–406, Dec 2014.

[35] Sree Harsha Yella. *Speaker diarization of spontaneous meeting room conversations.* PhD thesis, EPFL, Lausanne, January 2015.

[36] Ruiqing Yin, Hervé Bredin, and Claude Barras. Speaker Change Detection in Broadcast TV using Bidirectional Long Short-Term Memory Networks. In

18th Annual Conference of the International Speech Communication Association, Interspeech 2017, Stockholm, Sweden, August 2017.

[37] Ruiqing Yin, Hervé Bredin, and Claude Barras. Neural Speech Turn Segmentation and Affinity Propagation for Speaker Diarization. In *19th Annual Conference of the International Speech Communication Association, Interspeech 2018*, Hyderabad, India, September 2018.

[38] M. Zelenak, C. Segura, J. Luque, and J. Hernando. Simultaneous speech detection with spatial features for speaker diarization. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):436–446, Feb 2012.

[39] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 515–524, New York, NY, USA, 2002. ACM.

[40] W. Zhu, W. Guo, and G. Hu. Feature mapping for speaker diarization in noisy conditions. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5445–5449, March 2017.

List of Figures

List of Tables