Zurich University
of Applied Sciences

School of
Engineering

InIT Institute of Applied
Information Technology

Bachelor thesis (Computer science)

Voice Recognition with Deep Neural Networks

Author	Niclas Simmler Amin Trabi
Main supervisor	Thilo Stadelmann Oliver Dürr
Date	03.06.2018

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	Unterschriften:
…………………………………	………………………………………………………...
	………………………………………………………
	………………………………………………………

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Bachelorarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Zusammenfassung

Gemäss Definition ist ein Hauptbestandteil der *Stimmerkennung* die Identifikation eines Sprechers und die Zuweisung der Stimme zu diesem Sprecher. Eine typische Anwendung davon ist die *Sprecherdiarisierung*. Für eine beliebige Tonaufnahme (z.B. ein Telefonanruf, eine Besprechungsaufnahme oder eine Tonspur eines TV-Programms) kann die Sprecherdiarisierung zusammen mit einem Sprache-zu-Text Modul verwendet werden um die Tonquelle zu transkribieren. In diesem Fall würde das Sprache-zu-Text Modul den Text aus den gesprochenen Wörtern generieren und das Sprecherdiarisierung-Modul für die Problemstellung "wer wann gesprochen hat" zuständig sein.
In den vergangenen Jahren konnte das Institut für angewandte Informatik (InIT) grosse Erfolge im Bereich der Sprechererkennung verbuchen. Das Gebiet der Sprecherdiarisierung ist aber noch Neuland. Die vorliegende Arbeit versucht diese neuen, auf maschinellem Lernen basierenden, entwickelten Ansätze zu vereinen. Dieses Projekt soll dazu dienen, die Fähigkeiten dieser neuen Methoden zu demonstrieren. An die Leistung eines hoch entwickelten Sprecherdiarisierung-Systems heranzukommen ist zwar unwahrscheinlich aber dennoch wünschenswert. In erster Linie soll dieses Projekt Schwachstellen in der kombinierten Verwendung für Sprecherdiariserung dieser neuen Techniken aufdecken. Dies hat zum Ziel, dass neue Projekte vereinfacht an die Resultate dieser Arbeit anknüpfen können.
Verschiedene Experimente eruieren die Qualität der einzelnen Komponenten im System, um zukünftigen Projekten eine solide Basis mit Anknüpfungsmöglichkeiten zu bieten. Das entwickelte System ist ebenfalls fähig, gegen die *NIST RT-09 evaluation campaign* anzutreten, um es mit anderen Sprecherdiariserung-Systemen zu vergleichen. Leider ist die gemessene Leistung bei weitem nicht wettbewerbsfähig.

Abstract

By definition, an integral part of *voice recognition* is the identification of a speaker and attribution of voice to said speaker. Unifying those tasks is called *speaker diarization*. Given an arbitrary recording (or stream) of voice (i.e., a phone call, a meeting recording or the audio track of a television program), speaker diarization can be utilized, in conjunction with a speech-to-text module, to transcribe an audio source. Whereas the speech-to-text module would generate text from the spoken words, the speaker diarization module would answer the question "to whom the spoken word belongs."
In recent years, the Institute of Applied Information Technology (InIT) has made great progress in the area of speaker clustering and other voice recognition related topics. However, they have yet to cope with speaker diarization. This project tries to unify some of the newly developed deep learning approaches by the InIT. This work is expected to demonstrate how well these deep learning techniques work.
Matching the performance of a sophisticated state-of-the-art speaker diarization system is a difficult task and not expected from this project, but desired. Nevertheless, this project should pinpoint the weak spots of those new techniques when unifying them to a speaker diarization system. This way, new projects can emerge based on the results of this bachelor thesis.
Various experiments will show how well each component of the system performs and where future work needs to draw on to improve the performance of this system. This system is also able to evaluate on the *NIST RT-09 evaluation campaign*, which makes it comparable with other speaker diarization systems. Unfortunately, the performance of the system is by far not as good as the state-of-the-art systems.

Foreword

Artificial intelligence has been a topic of great interest for both of us, which is why it was without a question to team up and tackle this challenging topic.

We would like to thank various people for their inputs during the past weeks while we worked on this project.

Thilo Stadelmann & Oliver Dürr (ZHAW InIT) Thanks for supporting us during this past semester moreover, giving us the opportunity to work on such an exciting project.

Feliks Hibraj, Sebastiano Vascon & Ismail Elezi (Ca' Foscari University) Thanks for your helping and giving inputs on how to incorporate the dominant set clustering algorithm.

Jonathan Fiscus & Michael Garris (NIST) Thanks for the willingness to help and effort to get the RT-09 dataset to us.

Lukas Müller & Mario Marti (ZHAW Students) Thanks for their inputs and the exchange of experience from their bachelor thesis.

Savin Niederer & Benjamin Heusser (ZHAW Students) Thanks for their help setting up the ZHAW Deep Voice Suite.

Contents

1 Introduction

This chapter serves as an introduction to the speaker diarization topic. First, an overview of past work is given by talking about the origins of the topic in the background section and the motivation to do speaker diarization. Followed by an explanation of the pursued primary goal. At last, a general overview of the project is given by describing the content of each upcoming chapter.

1.1 Background & Motivation

Speaker diarization is an immensely important task in processing the spoken word. Mainly because it answers the questions "which person spoke what part at which time and for how long?" during a conversation. Two tasks have to be completed before speaker diarization can take place.

Speaker identification is the process of identifying different speakers. Basically, speaker identification answers to the question "who is speaking in this conversation?"

Speaker clustering is the follow-up process to identify where spoken snippets or phrases are attributed to a speakers identity. Speaker clustering gives an idea of "to whom are these spoken phrases belonging?"

Knowing who is speaking and which spoken word is belonging to whom enables us to do speaker diarization.
According to "Speaker diarization: A review of recent research" [1], speaker diarization has made great strides because of the National Institute of Standards and Technology (NIST) sponsored Rich Transcription Evaluation campaigns. The most recent Rich Transcription campaign (RT-09) [2] has produced various state-of-the-art systems using Gaussian Mixture Models (GMM) combined with Mel frequency cepstral coefficient (MFCC) as their clustering method of choice, such as the ICSI RT-09 Speaker Diarization System (ICSI system) [3]. There have been other Rich Transcription Evaluation campaigns, such as the ESTER Rich Transcription evaluation campaign, with its latest installment ESTER2 [4]. Because of ESTER2, another state-of-the-art system called "the LIUM open-source speaker diarization toolbox" (LIUM system) [5] has emerged. LIUM also uses GMMs in its architecture. Another noteworthy system is the "LIA-EURECOM RT-09 Speaker Diarization System" (LIA-EURECOM system) [6].

The Institute of Applied Information Technology (InIT), a department of the ZHAW School of Engineering, has made significant progress in the speaker clustering area using various types of neuronal networks, such as a convolutional neuronal network (CNN) approach in Lukic et al. [7] and a recurring neuronal network (RNN) approach in Gerber et al. [8]
During the project, advantage is being taken off their achievements to build a speaker diarization system which should be able to compete alongside the previously mentioned ICSI and LIUM system.

1.2 Goals

The primary goal is to showcase the improvement the newly developed deep learning approaches have produced over the previously used GMM/MFCC approaches, which were applied in the ICSI and LIUM systems. To be able to achieve this, a speaker diarization system needs to be built which can be benchmarked using the NIST RT-09 campaign.

1.3 Structure

The project is structured as follows:

- In chapter 2 an introduction will be given to the theoretical foundation needed to understand the system. Additionally some possible challenges will be shown.
- In chapter 3 the system will be dissected to get a detailed understanding of each part.
- In chapter 4 the results of the experiments will be described and discussed.
- In chapter 5 the results will be reviewed in regards to the goals and also some points will be outlined on where future work can be improved.

2 Theory

This chapter gives an overview of tools and topics relevant for this project. It starts by describing the datasets used and ends by describing an LSTM neural network and the dominant sets algorithm, which is the clustering algorithm used in this project.

2.1 Speaker Diarization

As aforementioned, speaker diarization is the task of identifying "who spoke what at which time." In theory, a speaker diarization system consists of various components. Intuitively a system would follow the following steps. This blueprint is also roughly incorporated in current state-of-the-art systems [3] [5] [6].

1. Preprocessing of the audio
2. Extracting the number of different speakers
3. Detecting the change of speaker
4. Assigning snippets of audio (the spoken phrases) to a speaker (also known as clustering)
5. Providing metrics to measure the performance

As it is usually not known how many speakers are in a given audio stream, speaker diarization can be considered as an unsupervised machine learning problem. There have not yet been any speaker diarization projects by the InIT. This work can be considered as the InIT's first take at speaker diarization. The motivation to tackle this problem mainly comes from the recent success in speaker clustering.

2.2 Datasets and Corpora

Previously the InIT used the TIMIT corpus for their speaker clustering tasks. To be able to compare the system developed in this project with others (ICSI [3], LIUM [5]), an evaluation on the RT-09 corpus will be needed. The RT-09 corpus was provided during the RT-09 challenge [2].

2.2.1 TIMIT

The TIMIT corpus consists of various speakers speaking different dialects. The corpus prompts the speakers to speak various standardized sentences. Thus, resulting in the same sentences spoken by different speakers. However, not every speaker speaks all the sentences. All sentences are spoken in the English language.

Dialect Region	# Male	# Female	Total
DR1 (New England)	31 (63%)	18 (27%)	49 (8%)
DR2 (Northern)	71 (70%)	31 (30%)	102 (16%)
DR3 (North Midland)	79 (67%)	23 (23%)	102 (16%)
DR4 (South Midland)	69 (69%)	31 (31%)	100 (16%)
DR5 (Southern)	62 (63%)	36 (37%)	98 (16%)
DR6 (New York City)	30 (65%)	16 (35%)	46 (7%)
DR7 (Western)	74 (74%)	26 (26%)	100 (16%)
DR8 (Army Brat - moved around)	22 (67%)	11 (33%)	33 (5%)
Total	438 (70%)	192 (30%)	630 (100%)

Table 2.1: TIMIT speaker distribution of dialects and genders [9]

The TIMIT corpus is not evenly distributed to male and female speakers, as it can easily be seen in table 2.1. It is also a rather small corpus regarding speakers, but it is providing many data per speaker. The TIMIT dataset comes in handy when the need arises to create test data during the development cycle as it consists out of clean or almost noiseless data.

2.2.2 VoxCeleb

The VoxCeleb corpus [10] is another dataset used during the development cycle. VoxCeleb is a dataset collected from YouTube videos. Nagrani et al. [10] created a pipeline which can produce a large dataset consisting of various celebrity voices. The dataset also provides ground truth for samples.

Statistic	Total
Speaker count	1'251
Utterances	145'379

Table 2.2: VoxCeleb Statistics [10] (simplified)

Due to the nature of this dataset, it does contain low-quality and noised audio samples. This way, during development the TIMIT-dataset can be used and then the results are validated on the VoxCeleb-dataset.
As of April 2018, a newer version of the VoxCeleb dataset called VoxCeleb2 [11] has been released by Chung et al. The VoxCeleb2 corpus is almost five times the size of the VoxCeleb corpus. The initial VoxCeleb-dataset suffices the needs for this project, which is why the VoxCeleb2 dataset is not used.

2.2.3 RT-09

The RT-09 (or RT09) corpus has been provided during the NIST RT-09 (or RT09) challenge. In contrast to the TIMIT corpus, the RT-09 corpus is made of 7 meeting recordings which were recorded at the Edinburgh University (EDI, 2 recordings), the IDIAP Research Institute (IDI, 2 recordings) and the National Institute of Standards and Technology (NIST, 3 recordings). All the recordings together yield roughly 180-minutes of meeting excerpts. The RT-09 challenge incorporated the three tasks:

Speech-To-Text (STT) - convert spoken words into streams of text.

Speaker Diarization (SPKR) - find the segments of time within a meeting in which each meeting participant is talking.

Speaker Attributed Speech-To-Text (SASTT) - convert spoken words into streams of text with the speaker indicated for each word.

The speaker diarization task provides two audio input conditions, also known as evaluation conditions. An audio input condition specifies from which microphone setting the audio originates.

Multi distant microphones (MDM) - this evaluation condition is made of various microphone recordings. All of the microphones are placed in the middle of the meeting table in between the participants. There are at least three omnidirectional microphones per meeting recording.

Single distant microphones (SDM) - this evaluation condition is made of a single microphone recording. This single omnidirectional microphone is placed in the middle of the meeting table in between the participants. This recording is also included in the MDM evaluation condition.

So-called evaluation maps delimit the various tasks. An evaluation map for a task (and input condition) specifies which segment of a meeting recording is relevant for the scoring. In contrast to the previously described datasets, the RT-09 corpus is highly noised, as it was recorded inside regular meeting rooms. Not only does it contain a lot of ambient noise, but also does it contain a lot of overlapping speech segments.

2.3 Speaker Clustering

Speaker clustering (SC) is a crucial task in speaker diarization. Since speech samples should be attributed to speakers, grouping (cluster) those samples together is needed. In the past years there have been many research projects by the InIT in speaker clustering. Most of those newly developed approaches are based on deep neural networks.
There was an effort in a recent project in 2017 to unify all those different approaches and make them usable in a single suite (Niederer et al. [12]). This developed suite is being used in this project, which is why it makes sense to harmonize the naming convention for the various sub-projects and follow the proposed naming scheme by Niederer et al.
Niederer et al. also unified the metrics used in the various approaches to make these systems comparable. Based on the results by Niederer et al., it was no problem to select one of the four approaches to supplement the speaker diarization system. All of the following systems used the TIMIT corpus as training material.

luvo - **Lukic et al. 2016 [7]** was the first approach by the InIT in speaker clustering, which yielded great success using a deep convolutional neural network. The network was able to produce embeddings, which were used in clustering the speakers. As clustering method agglomerative hierarchical clustering was used.

pairwise_kldiv - **Lukic et al. 2017 [13]** was the second take by the InIT in speaker clustering. It mainly sets itself apart from the "luvo" network by using weakly labeled data. The *pairwise_kldiv* network used a pairwise approach by comparing two audio samples. Then it determined whether they belong to the same speaker or not.

flow_me - **Gygax et al. [14]** was another approach developed during a bachelor thesis. Its architecture is comparable to the "luvo" network.

pairwise_blstm - **Gerber et al. [8]** this system differs from the previously seen networks. Gerber et al. used a bidirectional LSTM network (LSTM for **l**ong **s**hort **t**erm **m**emory). According to Niederer et al. this system outperformed the others. Therefore this neural network is used in this project.

2.4 Embeddings

Embeddings are mappings of an object to a vector representation with real numbers. A practical example for an embedding could be the color representation in the RGB color model. For example, the embedding for the color **RED** would be RGB(255,0,0). Similarly to the color information, audio information can also be represented in a vector. A popular vector representation for speech processing

applications, like speaker diarization, are MFCC (Mel Frequency Cepstral Coefficients) [15]. A disadvantage of such a representation is the number and determination of the relevant features, which have to be evaluated empirically.
Another increasingly used approach is to train and learn these representations. It is common to use deep neural networks as learning algorithms. The embeddings are the output of one of the hidden layers. The recent work of the InIT shows that embeddings are better suited for the speaker clustering task than handcrafted features, because they are able to represent the speakers identity [8] [14].
Since the *pairwise_blstm* network is used, the term embeddings refers to the *pairwise_blstm*-embeddings. The *pairwise_blstm* provides a 512-dimensional vector.

2.5 Audio Quality

Audio quality is an omnipresent topic in speech recognition tasks. It is without a doubt a topic which should not be neglected. For instance, a system which is performing really well on studio recordings might not yield the same results on the same task using cellphone recordings. Studio recordings usually are recorded using expensive equipment, inside noise reduced rooms, and may have post-processing applied, whereas cellphone recordings mostly are recorded on the streets with a rather cheap microphone and much ambient noise. This fact leads to the following problem. A system which performs well under lab conditions but not in the real world (does not generalize), will not be considered good.
The RT-09 corpus is used for this project, therefore, meeting recordings of mediocre quality has to be processed. The recordings were made using carefully positioned microphones, but they contain a lot of meeting room noise and moreover the various voices are sometimes overlapping. This is why the quality should be considered mediocre. The RT-09 provides meeting room schemes for the three meeting room setups which can be found in the appendix chapter A.4.3.

2.6 Spectrogram

A spectrogram is one way of representing an audio signal, another popular one is the waveform. To get a better understanding of what a spectrogram actually is and how it distinguishes to a waveform, one should look at an example recording. For this example, the phrase "Hello, I am an audio spectrogram" was recorded.
The waveform (see figure 2.1) is a simple representation of the amplitude over time. This representation does not give any insight on the audio recording, apart from the "loudness" of the speakers voice at a given time.

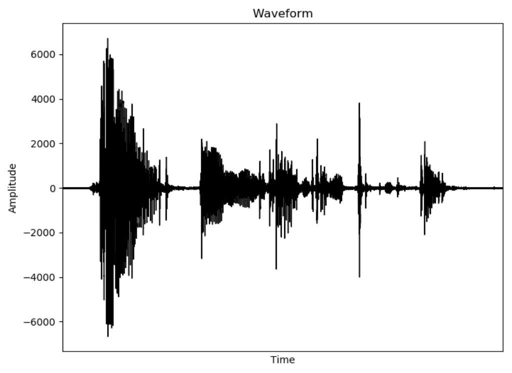


Figure 2.1: Waveform of "Hello, I am an audio spectrogram."

The spectrogram (figure 2.2) represents information in a 3-dimensional way and is an extension of the waveform. A spectrogram represents the loudness (z-axis) of a frequency (y-axis) over time (x-axis). A spectrogram is generated by applying the Fourier transformation on an audio signal.

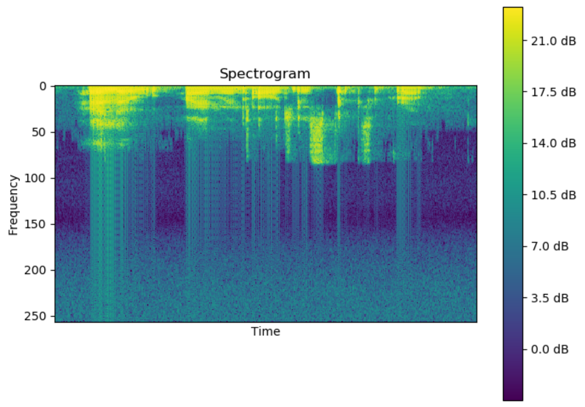


Figure 2.2: Spectrogram of "Hello, I am an audio spectrogram."

As it can be easily seen in the lower area of the plot in figure 2.2, there is a lot of noise and "useless" data. The mel-spectrogram (see figure 2.3), a special form of the spectrogram, is usually used in speech recognition tasks [15]. The mel-spectrogram is a "reduced" and special representation of a normal spectrogram. Reduced means it better represents audio in a way a human-being can distinguish the audio. The reduction is achieved by applying the mel-scale [16] on a regular spectrogram.

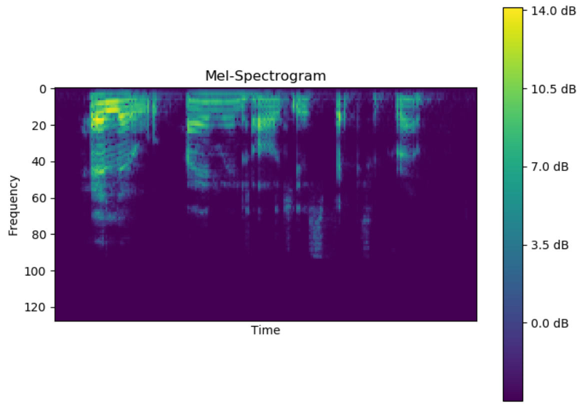


Figure 2.3: Mel-Spectrogram of "Hello, I am an audio spectrogram."

The *pairwise_lstm* network by Gerber et al. depends on mel-spectrogram representation of audio files.

2.7 Bidirectional LSTM Recurrent Neural Networks

A "Long-Short-Term-Memory" (LSTM) network is a special kind of a recurrent neural network (RNN). The idea of an LSTM was introduced in the paper "Long Short Term Memory" by Hochreiter and Schmidhuber, 1997 [17]. RNN can persist information through an inner loop. Thus, they can learn temporal dependencies like in audio signals or structural dependencies like in natural language. According to Christopher Olah RNNs do not differ much from a normal neural network: "A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor." [18] Another view of an RNN is a chain of modules, where a module corresponds to the repeating network.

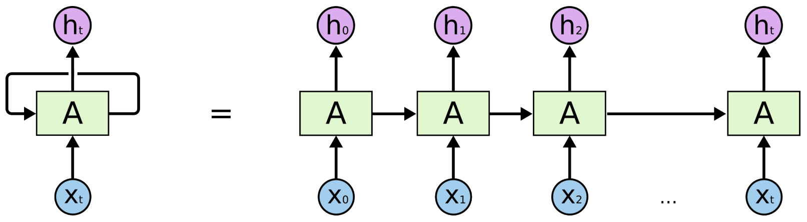


Figure 2.4: An unrolled recurrent neural network [18]

The disadvantages of RNNs is the missing capability to store long term dependencies, which is primarily caused by the vanish and exploding gradient problem detailed in Bengio et al. [19]. To solve this problems, Hochreiter et al. [17] introduced the LSTM architecture. Instead of every repeating network having one layer, a LSTM module or also called "cell" has four layers. Each of these cells yields two outputs the cell-state C_t and the actual output h_t.

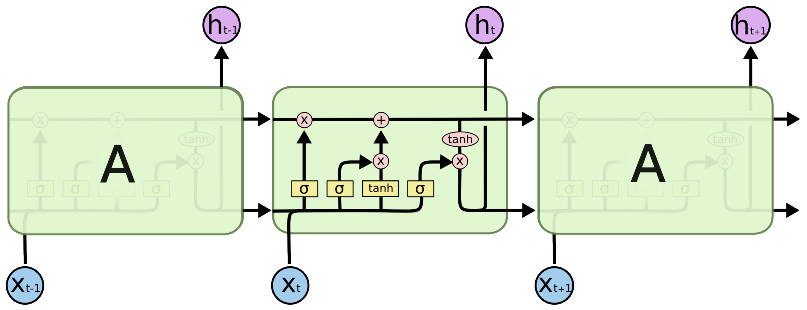


Figure 2.5: LSTM cell [18]

The key to solve the problem with long term dependencies is the cell state. The cell state flows straight through the chain with some minor interactions, namely the forget f_t and update u_t functions [18]. The forget function f_t defines which parts of the cell state to forget. The update function u_t defines which new information to add.
To calculate the new output h_t, the last σ layer is used. This layer filters the current cell state C_t and therefore defines which information to output.
A bidirectional LSTM network (BLSTM) processes the input from the beginning $x_0, \ldots, x_t$ and simultaneously from the end $x_t, \ldots, x_0$. It can be described as two independent networks, where each process the input in a different order [20]. The final output y_t results by the concatenation of the two outputs. Bidirectional layers can be used by all RNN architectures, where a bidirectional LSTM network uses the LSTM architecture.

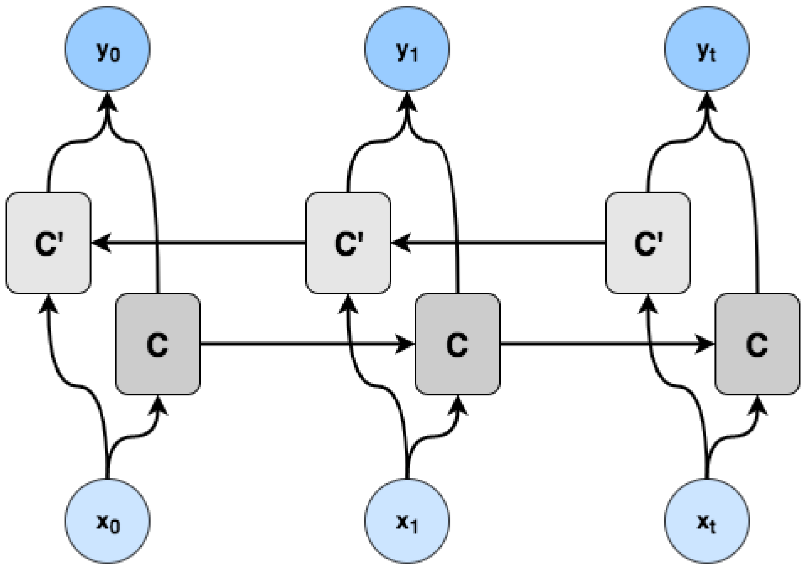


Figure 2.6: Bidirectional RNN [8]

The advantage of a bidirectional RNN is the knowledge of the past and the future at each input x_t, according to Alex Graves et al. [20].

2.8 Dimensionality Reduction

When operating on high dimensional data, one must somehow reduce the number of dimensions in order to be able to plot information.

2.8.1 Principal Component Analysis

Principal component analysis (PCA) is a procedure developed by Karl Pearson in 1901 [21]. Basically, *PCA* tries to reduce the number of dimensions of a dataset by selecting those dimensions which explain the most variation in the data. *PCA* is a pure mathematical method, therefore no machine learning is required. Hence, *PCA* is a great and easy to use tool for dimensionality reduction of a high dimensional dataset.

2.8.2 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (*t-SNE*) is a modern technique used to visualize high dimensional data. It was developed by Maaten et al. in 2008 [22]. *t-SNE* gained a lot of attention in the past as it appeared to work great in machine learning projects[1]. *t-SNE* is a probabilistic method based on gradient descent. It searches a lower dimensional representation of the high dimensional data with the goal to preserve structural information. *t-SNE* and *PCA* can be combined, whereas *PCA* is used to initially break down the dimensions before applying *t-SNE*.

[1]Based on the citation count of the paper.

2.9 Dominant Set Clustering

When referring to the term *dominant set clustering* (or *dominant set*) in this thesis, the newly developed speaker clustering (SC) algorithm by Felix Hibraj et al. [23] is meant. This algorithm outperforms other state of the art clustering techniques like spectral, hierarchical or k-means clustering in the SC domain. Similar to spectral clustering, dominant set clustering is a graph-based method, but with the aim to find dominant sets in the graph. According to Pavan et al. [24] analogies exist between a dominant set and a cluster.
The graphs have to be modeled as undirected acyclic weighted graphs $G = (V,E,w)$, where the vertexes V correspond to the items of the dataset (usually feature vectors). The edges $E \subseteq V \times V$ represent the pairwise relation between those items. Finally the weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ calculates the similarity of two items. A symmetric $n \times n$ adjacency matrix $A = (a_{ij})$ summarizes the graph G, where $n = |V|$ [23]:

$$a_{ij} = \begin{cases} w(i,j) & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

To find the dominant sets, the algorithm applies a concept of evolutionary game theory known as replicator dynamics (RD). *RD* operates a iterative selection process, where at each iteration some components emerge, while others get extinct. Extinct components do not belong to the current cluster, while emerged ones do. This selection process converges, if the adjacency matrix A satisfies the following property: A is symmetric and non-negative [23]. As converge criteria a threshold value ϵ is used. If the difference between the iterations i and $(i-1)$, is bellow or equal to ϵ, a new dominant set (cluster) is found. ϵ is therefore the first hyperparameter.
Theoretically an extinct component c has the value $x_c = 0$, where x_c defines the participation to the current cluster. In practice this is not the case. Therefore Hibraj et al. [23] introduced a cutoff value θ, the second hyperparameter of the algorithm. Instead of using an absolute cutoff value, a relative value $\theta \in [0,1)$ is applied. According to Vascon et al. [25], the participation of an component x_c is relative to the participation of the centroid, where θ defines the degree of this relationship. The selection process repeats until all components belong to a cluster, where at each iteration the current cluster is removed from the graph [23].
A distance metric like *euclidean* or *cosine* can be used as similarity metric. This is needed by the weight function w. Since the *cosine* metric showed good performance in *SC* [10], this is the default metric in the algorithm.
The big advantage of the dominant set algorithm is that no prior knowledge about the cluster number k is needed. This property is ideal to use dominant set as clustering algorithm in a speaker diarization system, because in speaker diarization the number of speakers is usually unknown.

3 Methods

This chapter provides an overview over the whole system developed during this project. It should also outline each component of the system on a theoretical level. This gives an insight on how the system is developed and how the development was planned.

3.1 Overview

For the sake of simplicity, the system developed in this project gets codenamed "SPDR" (originates from **sp**eaker **d**ia**r**ization). This should make it easier to reference the system in the following sections.

3.1.1 Idea

The main idea behind the SPDR system boils down to the following:

1. Load the dataset.
2. Extract embeddings from the audio files using the *pairwise_blstm* neural network by Gerber et al. [8].
3. Cluster the embeddings using an unsupervised clustering approach, preferably the "Dominant Sets Clustering" [23] by Hibraj et al.
4. Use a voice activity detector to single out clusters consisting of silence.
5. Calculate the diarization error rate.

Since an audio stream is a linear sequence of segments, it is not desired to randomly cluster segments together.
Lets assume an audio sequence with multiple 500ms segments is given as in figure 3.1. Given the characteristics of speech, it is highly unlikely that speakers speak in turn of 500ms windows. Hence, with a certain probability consecutive segments will belong to the same speaker. This is why some sort of pre-clustering is needed to gain insights over the audio to properly initialize the clustering algorithm.

Segments	1	2	3	4	5	6	7	8	9	10	11	12

Figure 3.1: Hypothetical audio sequence

One way of gaining this knowledge is to iterate over all segments and compare the next segment with the ones already processed. The idea is to measure the spatial distance of the previously seen segments and the next one. If the distance remains under a certain threshold, the segments will then be grouped to a sequence. Afterwards, this information of sequences will be fed to the clustering algorithm for initialization.

3.1.2 Pipeline & Components

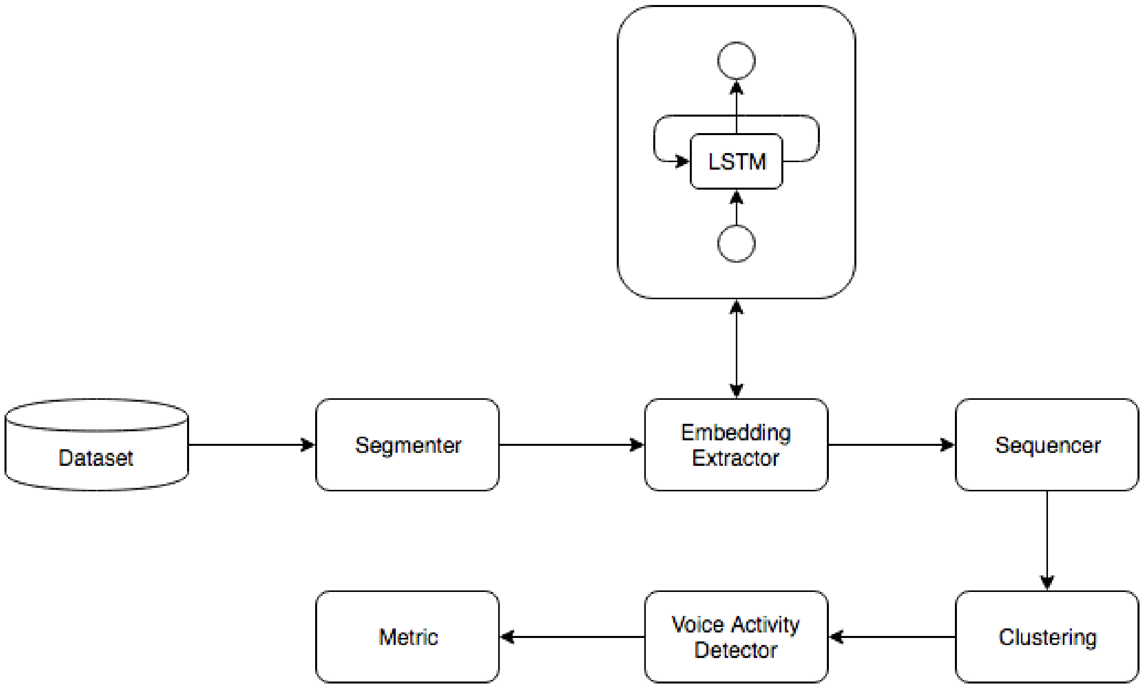


Figure 3.2: SPDR Pipeline

To implement the idea, a pipeline for the system as in figure 3.2 is needed. In chapter 3.2 an in-depth look at each component will be given.

3.1.3 Sanity Checks

In a system consisting of many components like the one shown, it is crucial to make sure each of them performs as expected. A series of experiments is introduced to ensure quality and isolated test each component. These experiments should pinpoint flaws in the implementation and reveal weak spots in the idea. Furthermore, the experiments should also give an idea on how to improve the system.

No.	Component	Problem	Approach
1.1	Embedding Extractor	Do different (regarding different phrases) samples of the same speaker generate similarly located embeddings in the Euclidean plane (using *t-SNE*)?	Use small mock datasets, with the length of 1 sample, of different phrases of the same speaker.
1.2	Embedding Extractor	Does the same (regarding the same phrase) sample of different speakers generate scattered embeddings in the Euclidean plane?	Use small mock datasets, with the length of 1 sample, of the same phrase of different speakers.
1.3	Embedding Extractor	Do different samples from different speaker generate scattered embeddings and locally cluster (per speaker) in the Euclidean plane?	Use small mock datasets, with the length of 1 sample, with different phrases of different speakers.

Continued on next page

Table 3.1: List of experiments to ensure quality results of components

Continued from previous page

No.	Component	Problem	Approach
2.1	Sequencer	Find a good cutoff value.	Build random pairs of segments, calculate the distance and see if "same speaker" and "not same speaker" segments are significantly apart.
3.1	Clustering	Is the clustering algorithm able to cluster same speaker segments to one cluster?	Use a small mock dataset consisting of multiple samples of the same speaker.
3.2	Clustering	Is the clustering algorithm able to cluster speaker segments from different speakers, each originating from different recording conditions, to multiple clusters?	Use a small mock dataset consisting of multiple samples of multiple speakers, with various recording conditions.
3.3	Clustering	Is the clustering algorithm able to cluster speaker segments from different speakers, each originating from the same recording environment, to multiple clusters?	Use a small mock dataset consisting of multiple samples of multiple speakers. Each of the speaker has to be recorded with the same recording conditions.
3.4	Clustering	How do the hyperparameters influence the performance of the clustering algorithm?	Perform a grid search on a small mock dataset, where the embeddings represent a speakers identity.
4.1	All	How does random perform?	Make multiple runs with different random seeds and calculate average.

Table 3.1: List of experiments to ensure quality results of components

3.1.4 Comparison to other systems

Other systems, such as the LIA-EURECOM system, the ICSI system, and the LIUM system, have similar architectures to the system described in this chapter. Since these systems do not depend on a neural network for feature extraction, they need to do their feature engineering in a pre-processing step. All of the other systems take great care of input sanitation (and enhancement) before feature extraction. Speech or voice activity detection is usually done before the clustering, which is probably the biggest difference to the SPDR system.
Other systems may also support multiple RT-09 evaluation conditions as described in chapter 2.2.3. The SPDR system only supports the *SDM* condition.

3.2 Components

This section provides an in-depth look at each component as shown in figure 3.2. The experiments described in table 3.1 are used to improve and test each of the following components.

3.2.1 Handler

The handler is responsible for the preprocessing of a given RT-09 dataset. It parses the ground truth such that, an evaluation of the system's performance is possible. After parsing the ground truth, the input audio files are converted to the needed format for further processing. The audio data is converted to mono-channel wave files (.WAV) because this format is used by Gerber et al. [8].

3.2.2 BLSTM - Neuronal Network

As mentioned in the theory chapter 2.3, the *pairwise_blstm* network developed by Gerber et al. [8] will be used. Thanks to Niederer et al. [12] this network is available in the *ZHAW Deep Voice Suite*[2]. The network works by feeding it with Mel-spectrograms of 500ms in length. A duration of 500ms, the *pairwise_blstm* showed the best performance, according to Gerber et al. [8]. For most of the development tasks, the best-trained model by Niederer et al. is used for the *pairwise_blstm* network. This model was trained on 100 speakers from the TIMIT corpus. Along with this project two new models of the *pairwise_blstm* were trained. These models were created to evaluate the system's performance using different embeddings. One model is trained on the RT-09 corpus, which contains 19 speakers. The other model is trained on a VoxCeleb subset containing 100 speakers. The three models are referred as follows:

- TIMIT_BLSTM
- VOXCELEB_BLSTM
- RT09_BLSTM

3.2.3 Segmenter

The segmenter is one of three components (the other two are the embedding extractor and the metric component) in the system, which do not need to be tuned in respect of a given data set. The segmenters task is to cut any given audio file in fixed-length segments (hence the name). All of the segments will be written to the disk and will not be kept in memory to keep the systems hardware resources footprint small.

3.2.4 Embedding Extractor

Once the Segmenter has written the segments to disk, the embedding extractor will generate a Mel-spectrogram for each segment. Since the *pairwise_blstm* expects a four-dimensional tensor, the Mel-spectrogram are stored in this tensor. The tensor has the following structure:

$$X[Segment, Channel, Frequency, Time]$$

The dimension of $Segment$ corresponds to the number of segments, $Channel$ has one, $Frequency$ has 128, and $Time$ has 50 dimensions. In the $Time$-dimension 100 units represent one second. The $Channel$ is set to one dimension because the audio files have one channel (mono) and the network was trained on mono audio files. The same applies to the $Frequency$ dimension. The generated Mel-spectrogram are stored in the dimension $Frequency$ and $Time$.
After processing this tensor, the neural network provides the embeddings as $n \times m$ matrix, where n is equal to the number of segments, and m is 512, the vector size of the embeddings. The embeddings extractor provides this matrix for further processing by the SPDR system.

3.2.5 Sequencer

The sequencer is a pre-clustering step, which yields some further influence on the clustering algorithm, apart from its hyperparameters. The sequencers job is to detect speaker change points in a very conservative way. It does so by looking at each embedding (segment) and calculating the distance to the previous ones if the distance falls under a threshold (referred as *cutoff value* throughout the following sections) the segment is considered as a prolongation of the current speaker's speech. The goal is to detect possible speaker change points. In return, it is desired to be able to detect consecutive same speaker segments with very high precision. This approach yields too many speaker change points as the cut off value is going to be set very conservative, but the clustering algorithm should then figure out which of those many sub-clusters belong together.

[2]https://github.com/stdm/ZHAW_deep_voice

Speaker change point detection

The sequencer will generate a simple vector containing 0 (for **not same** speaker as previous segment) and 1 (for **same** speaker as previous segment). Lets assume two speakers, A and B, are speaking in a sequence. The sequence consists of multiple segments. The sequence could look as shown in figure 3.3.

A	A	A	A	B	B	B	A	A	B	B	A

Figure 3.3: Example sequence between speaker A & B

After running the sequence of figure 3.3 through the sequencer, it will create the vector as in figure 3.4.

0	1	1	1	0	1	1	0	1	0	1	0

Figure 3.4: Generated vector for example sequence

A speaker change detection is then easily possible by iterating over the vector and detecting any falling edges (changes from 1 to 0).

Linkage method

When building a speaker sub-sequence for any new segment the question arise: "to which previously seen segment should the distance be calculated?". When selecting an element to calculate the distance to, one chooses the linkage method. Deciding which linkage method to use is as crucial as carefully selecting a good cutoff value.

Previous linkage means to calculate the distance to the previously seen segment.

Single linkage means to calculate the distance to the closest one of the previously seen same speaker elements. This method can also be expressed by

$$L(x,Y) = \min_{y \in Y} d(x,y)$$

where Y is the set of already seen same speaker segments, d is the distance function and x the current segment.

Complete linkage means to calculate the distance to the farthest one of the previously seen same speaker elements. This method can also be expressed by

$$L(x,Y) = \max_{y \in Y} d(x,y)$$

where Y is the set of already seen same speaker segments, d is the distance function and x the current segment.

Other linkage methods are for instance "average", "centroid" or "ward". All the methods except "single" and "complete" are quite computation-intensive because they cannot be calculated by simply iterating once over the already seen segments. Hence, the focus is set solely on the ones mentioned above.

3.2.6 Clustering

The clustering component is the heart of the diarization system. The clustering component utilizes the *dominant set clustering* algorithm as described in chapter 2.9. Clustering works on embedding level to cluster all elements. Ideally, the resulting labels correspond to the speaker's identity. Theoretically, any unsupervised clustering algorithm can be used, which also means that, only clustering methods without the prior knowledge of expected clusters are suitable.
During the development process good experience with clustering algorithms such as *Affinity Propagation* [26] and the *DBSCAN* [27] family has been made. If the *dominant set clustering* method appears not to be suitable for the job, it is recommended (solely based on the experience gathered during development of this system) to have a look at *HDBSCAN* [28] which is a refined version of *DBSCAN*.

Initialization

To improve clustering performance, it might be a good idea to initialize the clustering algorithm with the previously collected potential speaker change points. Because *dominant set* is a graph-based clustering method, these change points have to be defined inside the graph. To modify the graph, the adjacency matrix needs to be adjusted. There are basically three approaches to this problem with various peculiarities.

Flag change point One approach is to flag the change points. This can be achieved by setting the pairwise similarity of the change point-segment and the prior segment to zero. The zero value indicates a pairwise dissimilarity. This however, might be a dangerous approach: If there are many potential speaker change points detected, flagging all of them as distinct might upset the clustering algorithm and prevent it from actually clustering same speaker sub-clusters.

Flag same speaker Another approach is to flag segments within a sequence as equal by setting the pairwise similarity to 1.0. According to S. Vascon [29], the problem with this value is that the algorithm may yield a cluster for each equal pair. Hence, he recommends to use the max similarity value within the graph. Therefore, the max similarity value is used to flag equal segments. This approach ensures that sequences are not further divided. However, it is possible that change points are missed.

Flag both This approach is the combination of both approaches mentioned before. The change points and the sequences are defined in the adjacency matrix. It unifies the advantages (and disadvantages) of both methods.

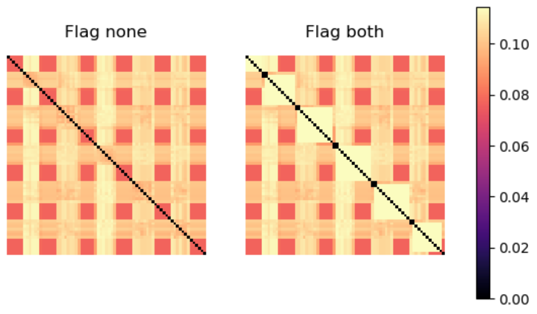


Figure 3.5: Adjacency matrix before initialization (left) and after (right) using the "Flag both" method

Figure 3.5 shows two heatmaps. The left heat map illustrates the adjacency matrix before the initialization procedure is applied, where the right one visualizes the adjacency matrix afterward. The black color represents a dissimilarity and the yellow color equality. The change points and the corresponding sequences are visible on the right matrix. The yellow squares along the diagonally are the sequences, and the smaller black squares between those indicate the change points.

3.2.7 Voice Activity Detector

The SPDR system implements a voice activity detector (VAD). The main purpose of this VAD is to filter out clusters which consist of non-speech segments.
The VAD can be placed at various positions in a diarization system pipeline. The table 3.2 should give an idea of the benefits for the VADs current location in our pipeline.

Location	Pros	Cons
Before Embedding Extractor	Less load on the whole system.	Nothing is known yet about the rest of the audio file. Difficult* to keep track of flagged non-speech segments throughout the whole system.
Before Clustering	Less load on the clustering algorithm (faster clustering).	Difficult** to keep track of flagged non-speech segments throughout the clustering algorithm.
Before Metric	Easy to implement because ideally all non-speech segments are clustered to one cluster.	Most load on the whole system because it will let the system do the work which then will be partially discarded.

Table 3.2: VAD pipeline location pros and cons

*Because as soon as the audio files are passed into the LSTM network, work is solely done on the embeddings. This means a separate list is needed in order to keep track of the non-speech segments.
**Because the clustering algorithm needs to be initialized carefully. When initializing the clustering algorithm with a distance matrix, all non-speech segments have to be marked inside that matrix.

Technically speaking, the VAD is a trained Gaussian Mixture Model. In its current configuration, the VAD loops through all clusters and classifies each segment either as speech or non-speech. The goal is to find clusters where the non-speech segments are prevalent ($> 50\%$). This default behavior can be reconfigured and thus, the aggressiveness adjusted. Depending on the dataset (noisy or clean) it can make sense to tune it more aggressive (non-speech to speech ratio of 25% or more) or more conservative (non-speech to speech ratio of 75% or more). For the sake of simplicity and engineering reasons the VAD is placed after the clustering component. When looking at other diarization systems, it might make sense to put the VAD at a different position in the pipeline.

3.2.8 Metric

Benchmarking a speaker diarization system requires various factors to be considered. Almost every speaker diarization challenge provides a different tool to benchmark a developed system and each tool requires the data to be in a different format. This is why a robust and easy to use system which provides standardized metrics is needed. Luckily there is an open-source python library which solves this problem. For the metric component the *pyannote.metrics* [30] library by Bredin 2017 is used.

Reference and Hypothesis

To calculate any metric value for speaker diarization with any given tool, two things are needed.

Reference is the ground truth of a given audio file, which includes the speaker ID, the time when the speaker began to speak and also the end time or duration.

Hypothesis is the counterpart to the reference. It consists of the information gained through inference. The hypothesis is generated by the system. Obviously, the speaker IDs do not match the reference IDs.

Diarization Error Rate

As already identified in the paper [30] of *pyannote.metrics* and by manually comparing various speaker diarization papers, the Diarization Error Rate (DER) is the standard metric to compare those systems. *pyannote.metrics* implements the NIST standard of the DER. The DER is calculated using the following formula:

$$\text{DER} = \frac{\text{false alarm} + \text{missed detection} + \text{confusion}}{\text{total}}$$

False alarm is the amount of time misclassified as speech where it should have been non-speech.

Missed detection is the amount of time misclassified as non-speech where it should have been speech.

Confusion is the amount of time the wrong speaker ID has been assigned to a spoken segment.

Total is the total amount of time considered as speech by the reference.

Purity & Coverage

Purity & Coverage is a dual evaluation metric, comparable (regarding duality) to Precision & Recall, although they do not represent the same idea. Purity & Coverage give insight into the quality of the DER. According to the paper "Segmentation, classification and clustering of an Italian broadcast news corpus" [31] by Cettolo 2000, Purity is a metric to judge the quality of a cluster.

The metrics, Purity & Coverage are calculated using the following formula [30]:

$$\text{purity} = \frac{\sum \max |\text{reference speaker} \cap \text{hypothesized speaker}|}{\sum |\text{cluster}|}$$

$$\text{coverage} = \frac{\sum \max |\text{hypothesized speaker} \cap \text{reference speaker}|}{\sum |\text{hypothesized speaker}|}$$

|**hypothesized speaker**| is the total time a hypothesized speaker speaks

|**reference speaker**| is the total time a reference speaker speaks

$\max$ |**reference speaker** ∩ **hypothesized speaker**| is the maximum time a reference speaker aligns with each hypothesized speaker

$\max$ |**hypothesized speaker** ∩ **reference speaker**| is the maximum time a hypothesized speaker aligns with each reference speaker

The above mentioned formula can give the following insights:

- If the system predicts too many different speakers, the purity score tends to be high while the coverage score tends to be low.
- If the system predicts too few speakers, the purity score tends to be low while the coverage score tends to be high.

F-Measure for Purity & Coverage

As with precision and recall, one can also calculate an F-measure (in this case called F1-score) for purity and coverage. This F1-score is being calculated using the following formula:

$$\text{PC-F-Score} = 2 \cdot \frac{\text{purity} \cdot \text{coverage}}{\text{purity} + \text{coverage}}$$

When speaking about the F-Score in this project, the mentioned *PC-F-Score* is meant.

Examples for Calculating Purity & Coverage

To understand how purity and coverage are calculated, an example will reveal the functioning, as the formulas are quite complex. Given an example where the purity score is 0.75 and the coverage score is 0.85 respectively.

Purity To calculate the purity for the scenario as in figure 3.6. One has to find the maximum time a reference speaker aligns with each hypothesized speaker. This will yield the $\max|$reference speaker$\cap$ hypothesized speaker$|$. The equation for this example looks like this:

$$\text{purity} = \frac{6+2+7}{20} = 0.75$$

Note: If there are multiple reference speakers aligning with a hypothesized speaker, the one with the maximum spoken time prevails.

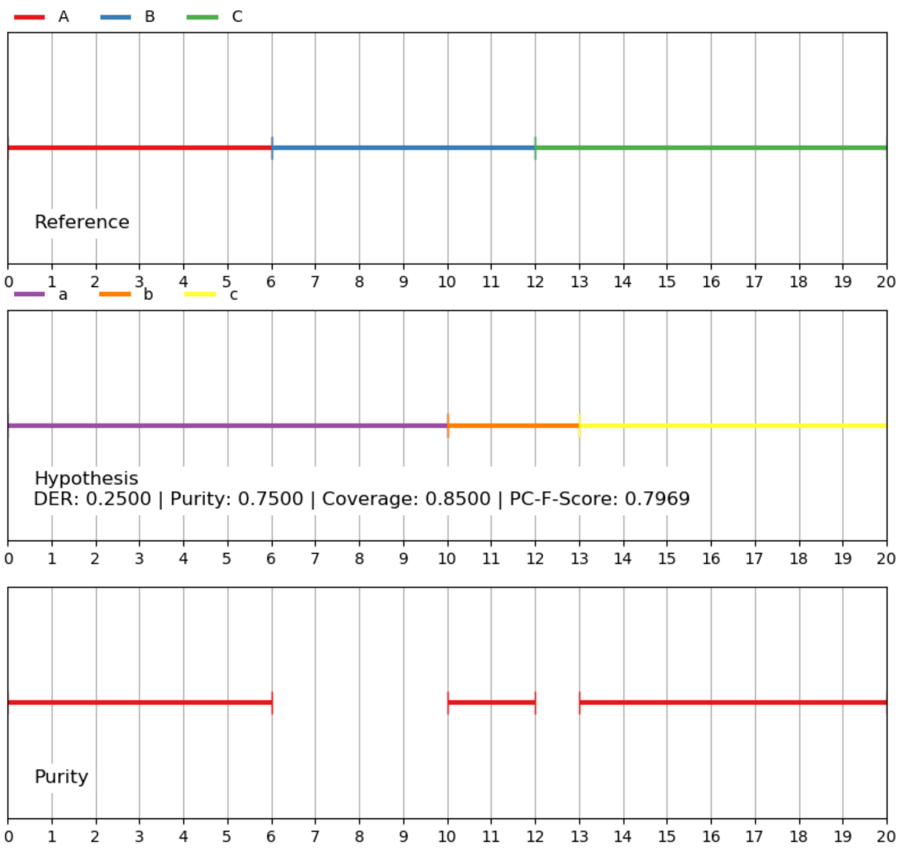


Figure 3.6: Example for calculating purity

Coverage To calculate the coverage for the scenario as in figure 3.7. One has to find the maximum time a hypothesized speaker aligns with each reference speaker. This will yield $\max |\text{hypothesized speaker} \cap \text{reference speaker}|$. The equation for this example looks like this:

$$\text{coverage} = \frac{6+4+7}{20} = 0.85$$

Note: If there are multiple hypothesized speakers aligning with a reference speaker, the one with the maximum spoken time prevails.

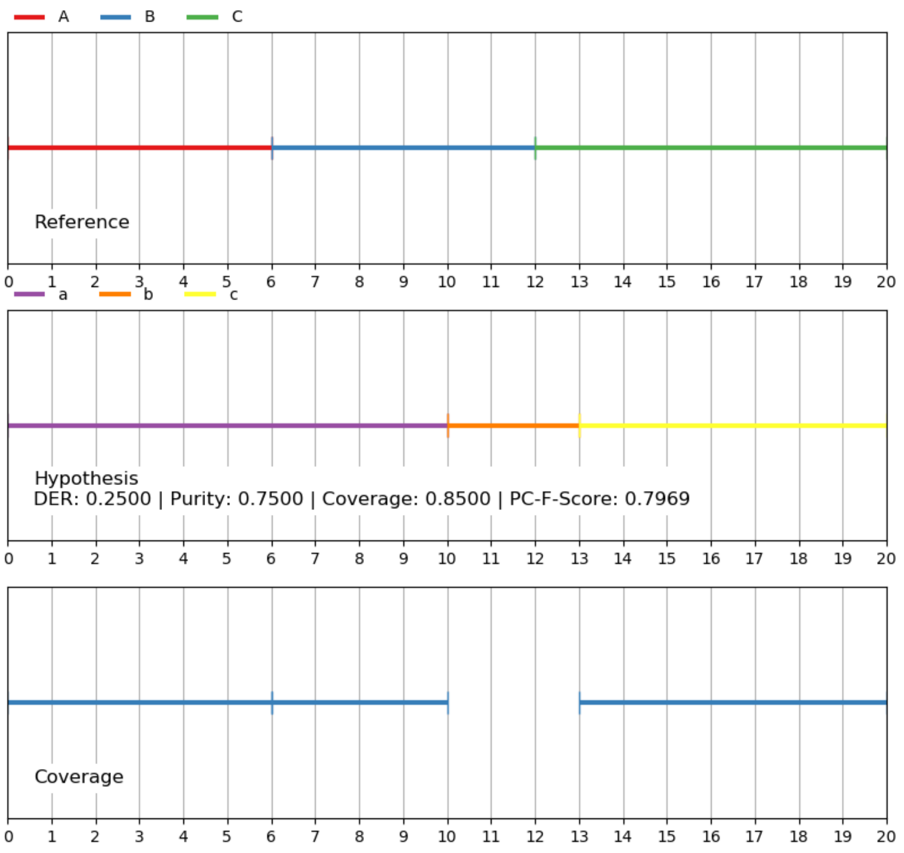


Figure 3.7: Example for calculating coverage

Examples for Over- and Under-Clustering

In the following examples, the impact of various clustering behaviors on the Purity & Coverage values will be visualized. For the sake of simplicity, it is assumed that the system does perfectly detect silence. Given is the fictional reference as in figure 3.8.

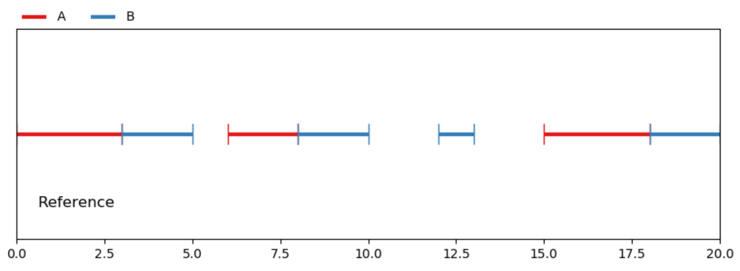


Figure 3.8: Example Reference

Perfect diarization Ideally it is desired to achieve perfect diarization (DER 0.0). Note: The detected clusters are intentionally named differently because unless previously initialized, a speaker diarization system cannot guess the names of speakers.

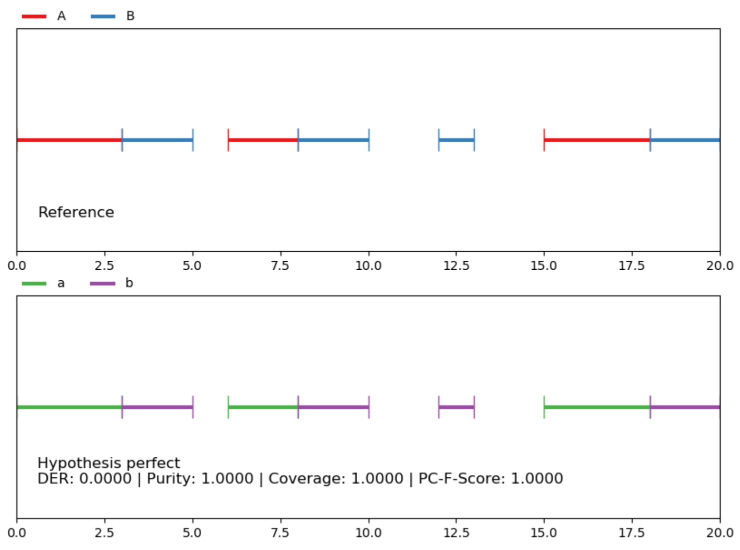


Figure 3.9: Reference and hypothesis with perfect diarization

Over-clustering As already mentioned in the Purity & Coverage section, over-clustering (yielding too many speakers and too many change points) will yield a high purity and low coverage value.

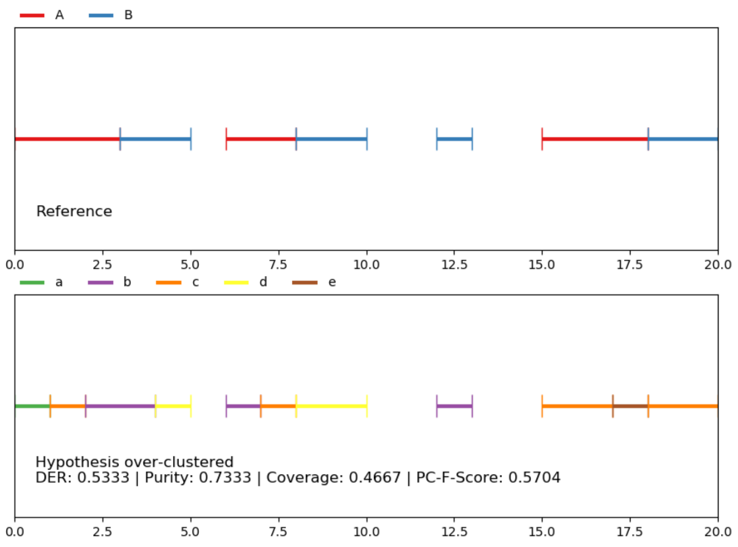


Figure 3.10: Reference and hypothesis with high purity and low coverage

Under-clustering Vice versa, under-clustering (cluster too many different speakers as one) will yield a low purity and high coverage value.

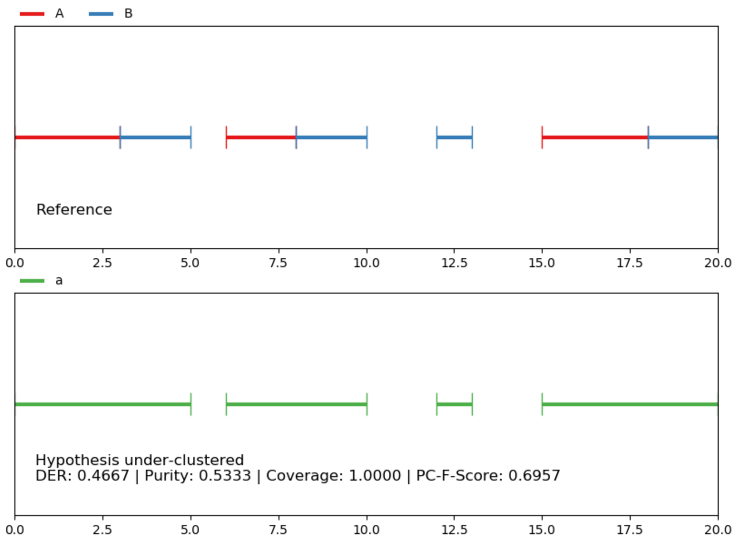


Figure 3.11: Reference and hypothesis with low purity and high coverage

4 Results

In this chapter, the results of each experiment as described in chapter 3.1.3 are presented. After having a detailed look at each experiment, the performance of the SPDR system is demonstrated on the RT09 dataset.

4.1 State Of The Art

In table 4.1 various results on the RT-09 corpus are presented. This should give a rough idea on what can be expected from a speaker diarization system.

Condition	LIA-EURECOM [6]	ICSI [3]	Hanwu et al. [32]
MDM	23.5%/18.5%	39.27%	-
SDM	26.0%/21.5%	44.61%	17.34%/12.10%

Table 4.1: Performance of various speaker diarization systems on RT-09

If there is more than one value in the table, then the first value is the result of "with overlapping audio" and the second value is the result of "without overlapping audio". The RT09 evaluation campaign allowed to be evaluated while ignoring parts where multiple speakers (overlapping speech) are speaking at once. As there are various meetings in the RT-09 dataset, the presented DER is the overall score.

4.2 Experiments

As already mentioned in the chapter 3.1.3, multiple experiments serve to ensure the quality and performance of the system. Some are intended to give a clear indicator of what to expect, and others should give insights why some things do not work whereas others do. Listed in this section are the detailed experiments and their interpretations.

4.2.1 Experiment #1.1

Goal

This experiment aims to verify if embeddings of one speaker are similarly located in the Euclidean plane.

Expectation

Assuming the embeddings represent a speaker's identity, it is expected that the embeddings form one (dense) cluster if they are from the same speaker.

Setup

Mock datasets are used in order to compare the results of this experiment on various conditions and different constraints. The mock dataset for this experiment is a small dataset consisting of only one speaker but multiple spoken phrases by this speaker.
As source, multiple subsets originating from the following dataset are used:

- TIMIT (studio recordings, no overlapping speech, no noise)
- VoxCeleb (YouTube videos, various recording conditions, overlapping speech, noise present)
- RT09 (meeting recordings, various recording conditions, overlapping speech, noise present)

A mock dataset is defined by the following parameters:

- Source dataset
- Number of speakers
- Number of utterances per speaker
- Number of segments per utterance

The total number of samples (segments) n is therefore:

$$n = \text{no. of speakers} \times \text{no. of utterances} \times \text{no. of segments}$$

A utterance is segmented with a duration of 500ms, whereby the parameter "No. of segments" defines the maximum. For each segment, the script yields a tuple with following structure: (Mel-spectrogram, speaker-label). The speakers and the utterances are selected randomly to ensure a variation of the data. For each Mel-spectrogram embeddings are generated using each of the following trained models:

- TIMIT_BLSTM (trained on the TIMIT dataset)
- VOXCELEB_BLSTM (trained on the VoxCeleb dataset)
- RT09_BLSTM (trained on the RT-09 dataset)

Scatter plots are used to analyze the embeddings, which visualize the location of the data in the Euclidean plane. In order to plot the embeddings, a dimensionality reduction is performed using *t-SNE*. Each data-point is colorized according to the speaker label. In this experiment a dataset is used with five utterances of one speaker to ensure a variation in the spoken word. Each utterance is split into four segments.

Results

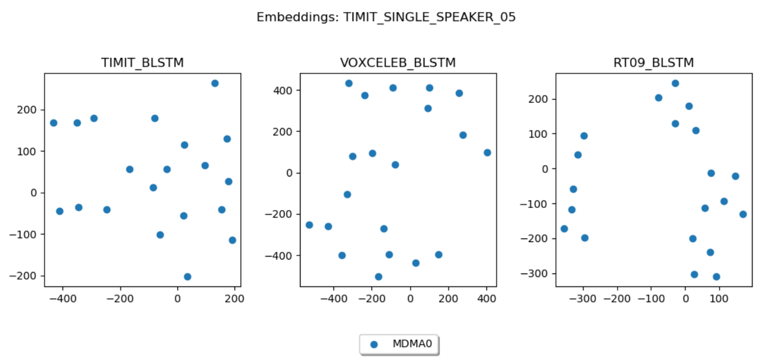


Figure 4.1: Single speaker with multiple utterances of the TIMIT-dataset

The result shows that none of the embeddings are arranged as one cluster. However, the distances between the embeddings of the TIMIT-network are of equal size and also quite dense, whereas the distances of the other embeddings have a wider variance. This indicates that the *pairwise_blstm* does not generalize well enough on other datasets.

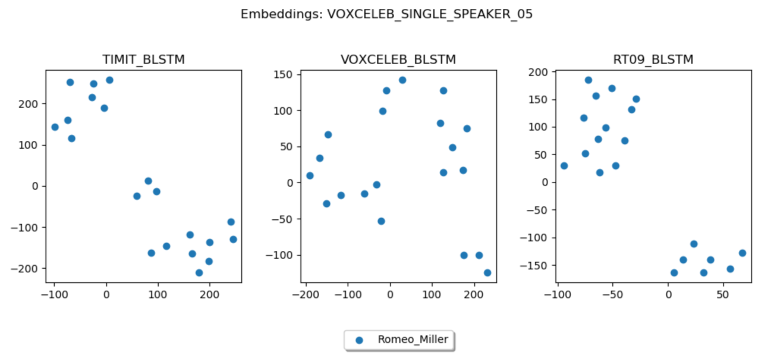


Figure 4.2: Single speaker with multiple utterances of the VoxCeleb-dataset

On the VoxCeleb dataset all three plots look similar. The TIMIT- and RT09-embeddings are arranged in three clusters, whereas the VoxCeleb-embeddings form five clusters.

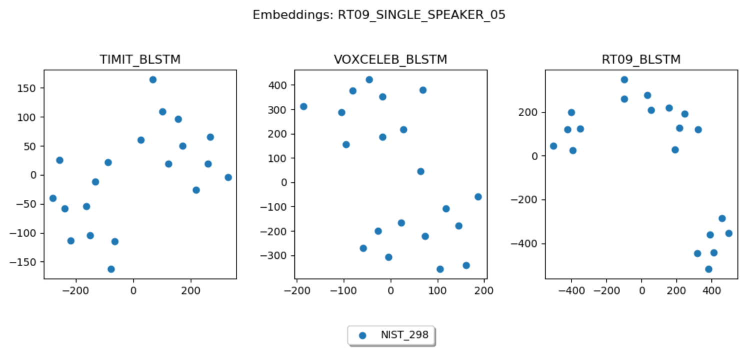


Figure 4.3: Single speaker with multiple utterances of the RT-09-dataset

On the RT-09 dataset the performance of the networks is equally bad as on the VoxCeleb dataset. None of the embeddings are arranged as one cluster.

Interpretation

This experiment demonstrates that the embeddings do not represent a speaker's identity. However, compared to the other datasets, the run on the TIMIT-dataset shows the best performance. The embeddings have a denser distribution. After all, this experiment indicates that the embeddings have a bigger distribution for each speaker than expected.

4.2.2 Experiment #1.2

Goal

This experiment aims to verify if embeddings of multiple speakers using only one utterance are significantly distributed (clustered per speaker) in the Euclidean plane.

Expectation

It is expected that the embeddings form one cluster per speaker.

Setup

Datasets are generated as in experiment 1.1 to run this experiment on various conditions. Instead of one speaker, five speakers are used, each with one utterance. Four segments are used for each utterance. The neural networks *TIMIT_BLSTM*, *VOXCELEB_BLSTM* and *RT09_BLSTM* are used as embedding-generators.

Results

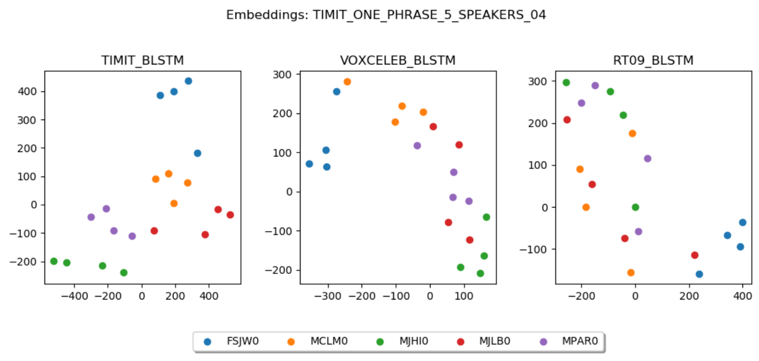


Figure 4.4: Five speakers with one utterance of the TIMIT-dataset

The TIMIT-embeddings represent one cluster per speaker, except for two outliers. The VoxCeleb-embeddings show two clearly distinct clusters and one cluster with multiple speakers, whereas the RT09-embeddings solely represent mixed clusters.

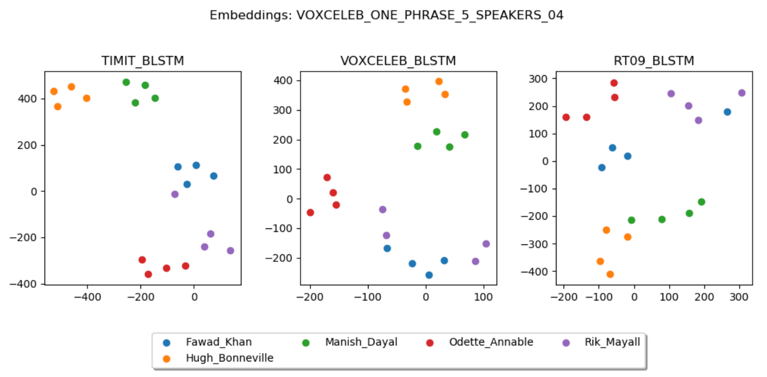


Figure 4.5: Five speakers with one utterance of the VoxCeleb-dataset

All embeddings form one cluster per speaker on the VoxCeleb-dataset, with one outlier in the RT09-embeddings.

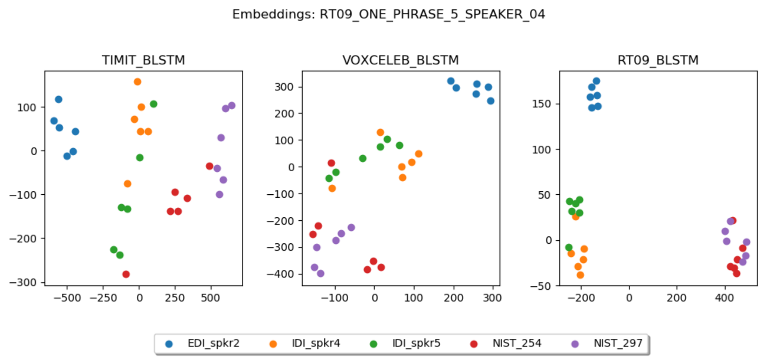


Figure 4.6: Five speakers with one utterance of the RT09-dataset

The VoxCeleb- and RT09-network show three and the TIMIT-network four clusters. The number of clusters by the VoxCeleb- and RT09-embeddings corresponds to the number of meetings. The speaker label identifies the meeting (EDI, IDI, and NIST) to which a speaker belongs.

Interpretation

The results of this experiment meet the expectation on the VoxCeleb-dataset and by the TIMIT-network on the TIMIT-dataset. On the RT09-dataset all embeddings form three clusters, which correspond to the number of meetings and the speakers originating from those meetings. This experiment shows that the *pairwise_blstm* can distinguish speakers if each utterance has different audio characteristics. The VoxCeleb-dataset meets this condition since these recordings have different recording qualities. It appears as the TIMIT-network performs well if the recording environment does not differ like in the TIMIT-dataset, This characteristic also applies to a meeting of the RT09-dataset, but compared to TIMIT, overlapping speech is present. Which is probably why the TIMIT-network performance is worse on the RT09-dataset than on the TIMIT-dataset.

4.2.3 Experiment #1.3

This experiment is a combination of experiment 1.1 and 1.2. It uses ten speakers, whereas eight utterances support each speaker. Since this experiment is a combination of the two experiments, it is a closer approximation of the actual application.

Goal

This experiment aims to verify that embeddings of multiple speakers using multiple utterances are significantly distributed (clustered per speaker) in the Euclidean plane.

Expectation

Since the TIMIT-network showed a good performance in experiment 1.1 and 1.2., the TIMIT-network is expected to perform well on the TIMIT-dataset. The VoxCeleb-dataset is expected to perform rather well. According to the results of experiment 1.2, the RT09-dataset is expected to be clustered per meeting instead of per speaker.

Setup

Datasets are generated as in experiment 1.1 and 1.2 to run this experiment on various conditions. Although, more variation in the data is needed to simulate the real use case. One utterance contains four segments as in the previous experiments.

Results

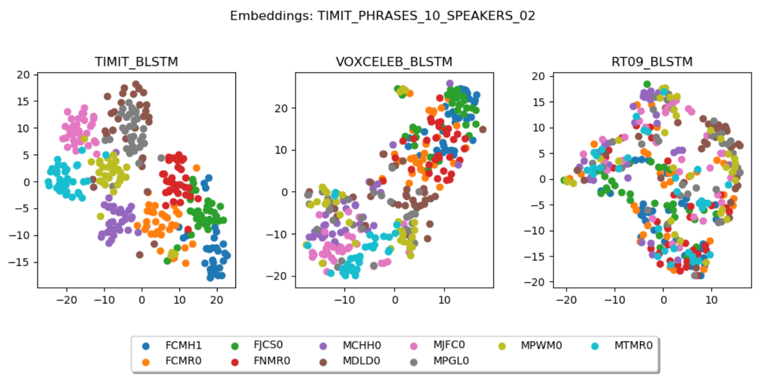


Figure 4.7: Ten speakers with multiple utterances of the TIMIT-dataset

The TIMIT-embeddings form clusters per speaker. It does not perfectly cluster the utterances, for instance the speakers MPGL0 (gray) and MDLD0 (brown). Besides that, some outliers are present. The VoxCeleb-embeddings do not form distinct clusters. The plot of the RT09-embeddings just contains mixed clusters, which makes the result rather useless.

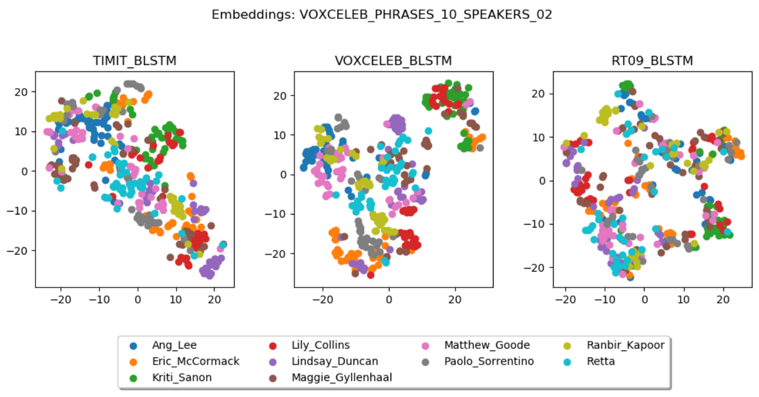


Figure 4.8: Ten speakers with multiple utterances of the VoxCeleb-dataset

None of the embeddings form clusters. The cluster-like piles do not represent speakers.

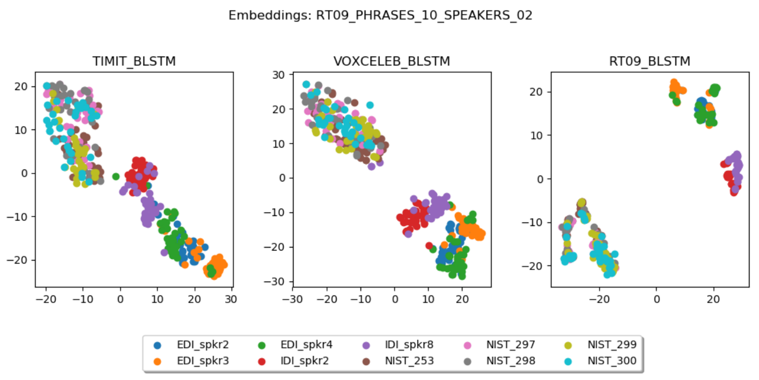


Figure 4.9: Ten speakers with multiple utterances of the RT09-dataset

All plots look similar. Each of them visualizes three rough clusters.

Interpretation

This experiment meets the expectations. The TIMIT-network shows the best performance on the TIMIT-dataset. On the RT09- and VoxCeleb-dataset no neural network outperforms the others. As expected, on the RT09-dataset there are roughly three clusters present. The results lead to the conclusion that the embeddings represent the different meetings instead of the different speakers. Each meeting shows different recording qualities, which the neural networks detect. Within a meeting, the networks are not able to differentiate between the speakers.

General interpretation This experiment indicates that the *pairwise_blstm* does not just learn characteristics of speech, it mostly learns characteristics of the environment. If this characteristics vary (e.g. more or less noise in the background), the *pairwise_blstm* uses this to distinguish speakers. If these characteristics do not vary, the neural network is forced to use speech characteristics (or something else) for the distinction. This behavior is emphasized during the runs with the TIMIT-network. On the TIMIT-dataset, which does not contain a wide variation in the recording quality, it is able to distinguish speakers. On datasets with varying recording conditions, like RT09 or VoxCeleb, it uses these variations as differentiation criteria.

Another issue is overlapping speech. The recordings of the RT09 dataset contains a lot of utterances with overlapping speech, especially the NIST-meeting, which explains the wide distribution within the NIST-meeting cluster.

4.2.4 Experiment #2.1

This experiment serves to proof whether a certain pre-clustering is possible using a cutoff value inside the sequencer component. The idea behind the pre-clustering is to group consecutive segments if they are spatially dense and initialize the clustering component with this knowledge. If the distance of consecutive segments surpasses the cutoff value, a new group will be created. The cutoff value ought to be chosen quite conservative as we would like to group same speaker segments with high precision. As the cutoff value is used in conjunction with the spatial distance of two given embeddings, this experiment shall be re-run for each dataset/neural network model combination.

Goal

Demonstrate the possibility to pre-cluster using a cutoff value.

Expectation

Assuming the embedding extractor delivers quality embeddings in terms of speaker identification*, finding an appropriate cutoff value should be fairly easy. Easy means it should be obvious where the cutoff value should be.

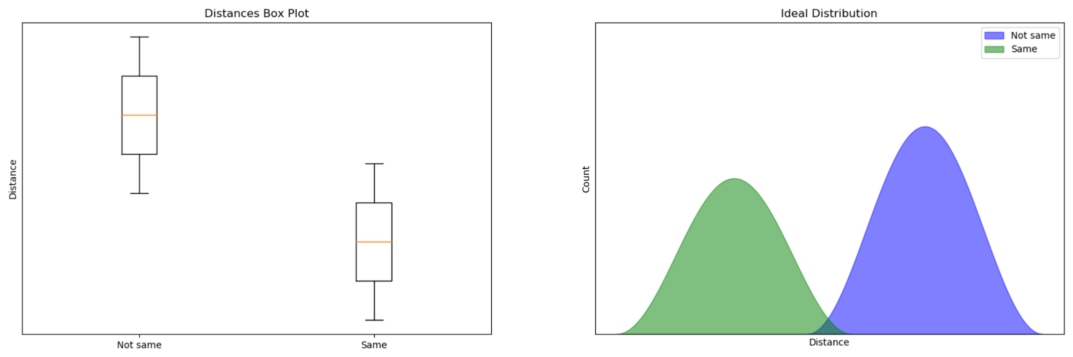


Figure 4.10: Ideal distance distribution

Ideally, the experiment yields plots as shown in figure 4.10. The plots show the spatial distances of a multitude of segment pairs. Some segment pairs are from the same speaker, and some segment pairs are from distinct speakers. A good cutoff value would be somewhere close to the point where the two curves intersect (or slightly to the left of it), as almost all "same speaker" segments would get hit using this cutoff value and almost no "not same speaker" segments.

* In theory, same speaker segments should be spatially dense whereas distinct speakers should be significantly apart. If the neural network properly learned how to identify speakers, then this should be no problem. However, if it learned to distinguish the reverberation of the room (or something similar), then this will be difficult or rather impossible.

Setup

Since finding the cutoff value is crucial and the various datasets may differ in the result of this experiment, this experiment has to be done manually.
A dataset consisting of embeddings power the experiment. During this experiment pairs of segments are created.

- Each embedding is annotated by the identity of its speaker.
- Half of the embeddings are fully joined with the other half to create pairs ($\sim 40'000 - 50'000$ pairs).
- Each pair is annotated whether it consists of same speaker segments or not.
- The in-pair spatial distance is calculated.

Results

During development this experiment has been run multiple times with different combinations as shown in table 4.2. Note: The dataset mentioned in table 4.2 does not refer to the whole corpus, it is just a small subset of 500ms segments.

Run No.	Dataset	Network
1	TIMIT Random	TIMIT-trained
2	RT09 EDI Meetings	TIMIT-trained
3	RT09 EDI Meetings	VoxCeleb-trained

Table 4.2: Various experiment setup combinations

Keeping in mind, an ideal distribution (as in figure 4.10) on the plots for these runs is expected

Run 1 - TIMIT on TIMIT This run is expected to perform the best result according to the setup condition.

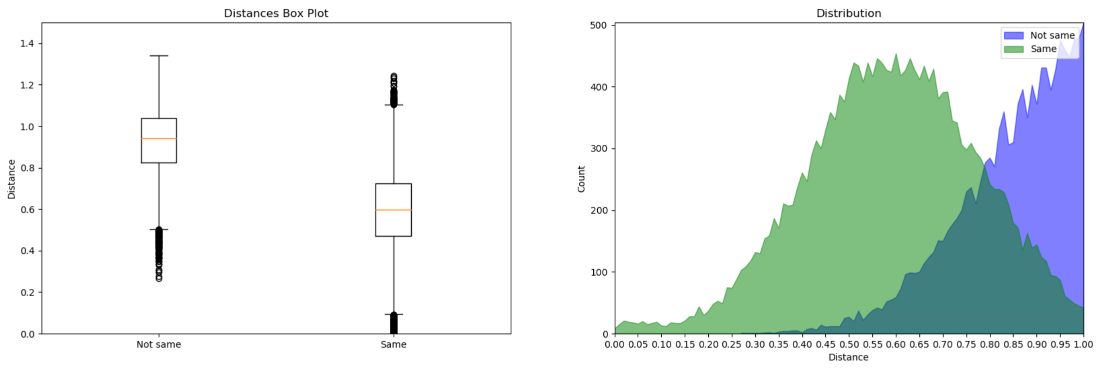


Figure 4.11: TIMIT on TIMIT: distance distribution

Run 2 - RT09 on TIMIT This run is expected to perform the worst result because of the experience with previous experiments and results during development.

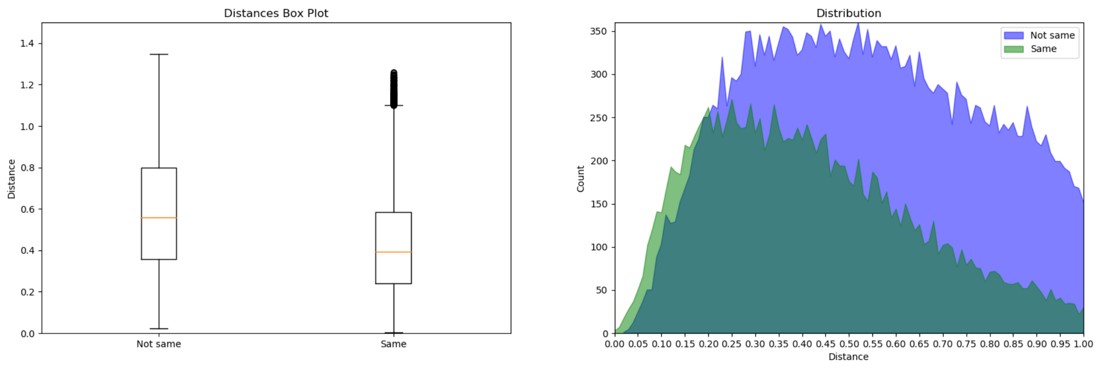


Figure 4.12: RT09 EDI on TIMIT: distance distribution

Run 3 - RT09 on VoxCeleb This run is expected to perform equally as the second run, since the VoxCeleb trained network model showed the same performance on the RT09 dataset as the TIMIT trained network.

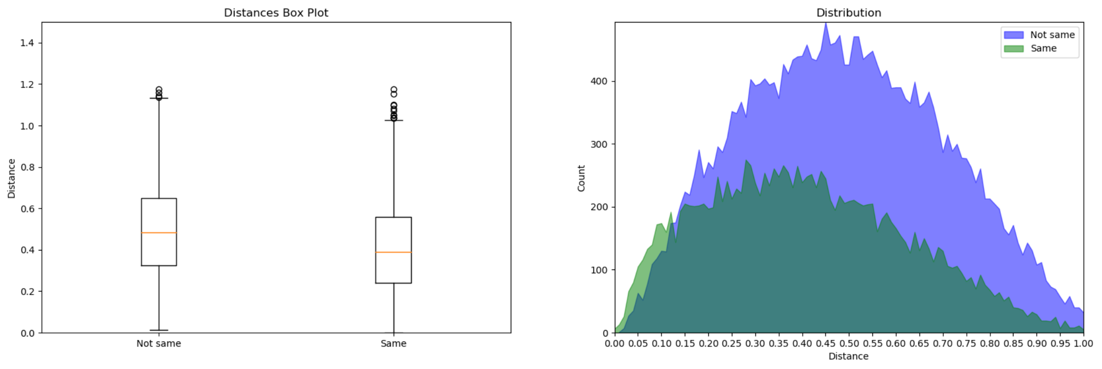


Figure 4.13: RT09 EDI on VoxCeleb: distance distribution

Interpretation

This experiment sheds some light on various things.

Interpretation of figure 4.11 Even on a well-trained model (TIMIT) with almost noiseless data (TIMIT), it is hard to find a clear cutoff value. The two curves are partially overlapping but can still be separated to some degree. It also shows that the idea theoretically works on good data.

Interpretation of figure 4.12 and 4.13 As expected, these runs (2 and 3) are much worse than run 1. It is almost impossible to select a good cutoff value, which means one would need to resort to a cutoff value close to 0.0. This also means in return more stress is put on the clustering algorithm. Seeing these plots, good performance of the system is highly unlikely.
These plots not only show that the quality of the embeddings is not ideal, but it also gives the notion that the neural network cannot clearly identify speakers but rather outputs embeddings for room conditions. This notion is derived from the fact that the curves are heavily overlapping.

General interpretation Generally speaking, finding a good cutoff value is difficult after all. However, having good quality embeddings which are strongly tied to the speaker's identity, would possibly have great impact on the performance of the system.

4.2.5 Experiment #3.1

This experiment serves to test the clustering component in the pipeline. *Dominant set* is used as clustering algorithm. The clustering component is essential for speaker diarization and therefore for this system, since it is responsible to determine the correct number of speakers and the assignment of segments to a speaker. Therefore, it is indispensable to test the performance of the clustering component. As the embeddings are clustered, the performance strongly depends on the quality of these. In this experiment embeddings of only one speaker are used.

Goal

The aim of this experiment is to determine whether embeddings originating from the same speaker are clustered to one cluster.

Expectation

Assuming the embeddings represent a speaker's identity, it is expected that they are clustered to one cluster.

Setup

Ground truth is needed to evaluate the performance. Since embeddings are clustered, each of them has to be labeled with the corresponding speaker. This condition is met by the generated datasets used in the embedding experiments (chapter 4.2.1). The generated datasets, which fulfill the needed setup of this experiment, are the "single speaker"-datasets of experiment 1.1. They contain five utterances of one speaker. According to experiment 1.1, the embeddings of the TIMIT-network on the TIMIT-dataset meet our expectations of the embeddings to perform speaker clustering. Therefore, the generated TIMIT-dataset is used in this experiment. Since the dataset contains the Mel-spectrograms, the embeddings have to be extracted. The TIMIT-BLSTM network is used as extractor.
These embeddings are clustered with the *dominant set* algorithm with the hyperparameters set to $\epsilon = 1e-6$, $\theta = 0.15$ and metric = *cosine*. These setup is the best parameter choice according to Felix Hibraj et al.[23] Two scatter plots are used to evaluate the clustering performance. One visualizes the embeddings colorized by the speaker labels (ground-truth) and the other one shows the embeddings colorized by the cluster assignment (prediction). Dimensionality reduction is performed using *t-SNE* to be able to plot the embeddings.

Results

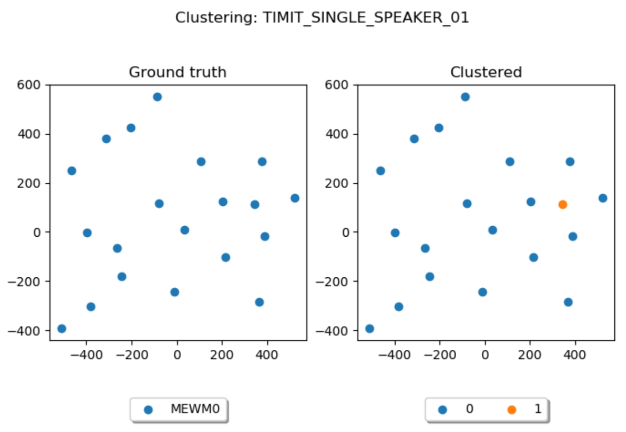


Figure 4.14: Clustering of one speaker

This plot shows that the embeddings, except for one outlier, are assigned to one cluster.

Interpretation

This experiment indicates that *dominant set* can cluster embeddings of one speaker to one cluster. It shows that *dominant set* is suitable for speaker clustering.

4.2.6 Experiment #3.2

This experiment tests *dominant set* on a dataset consisting of various speakers. The dataset contains speakers, which have been recorded under different conditions. This variation enables an evaluation of the clustering performance. Since variation in the data is needed, ten speakers are used.

Goal

The goal of this experiment is to check whether *dominant set* can detect several clusters when using multiple speakers.

Expectation

As shown in the embedding experiments, the embeddings represent rather the meeting room conditions than the speaker's identity. *Dominant set* is therefore expected to cluster the embeddings to three clusters (one for each meeting).

Setup

As in experiment 3.1, a generated dataset from the embedding experiments is used. The RT09-dataset contains three meetings, and each one has a different recording setup. Therefore, a generated RT09-dataset is used, which contains ten speakers, each with eight utterances. The TIMIT-network is used to extract embeddings. For *dominant set* the hyperparameter are set as in experiment 3.1 ($\epsilon = 1e-6$, $\theta = 0.15$ and metric = *cosine*). The evaluation is performed as in experiment 3.1 with two scatter plots, where one shows the ground truth and the other the clustered (predicted) result.

Results

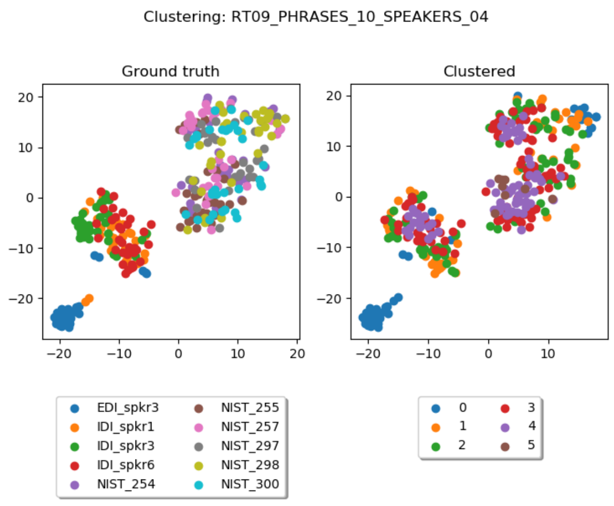


Figure 4.15: Clustering of ten speakers in various recording conditions

This plot shows that *dominant set* finds six clusters. Contrary to the expectation, the clusters are spread over the different meetings (visible clusters). As an example, the cluster "0" (blue), which corresponds to speaker "EDI_spkr3", does have utterances in a NIST-meeting.

Interpretation

This experiment does not meet the expectations. Instead of a cluster for each meeting, there are six clusters present. These clusters are not restricted to a meeting, neither are they tied to the speakers, which shows that *dominant set* connects the embedding in a different way.

4.2.7 Experiment #3.3

This experiment tests *dominant set* with multiple speakers, where all are recorded in the same environment. The recording characteristics will not vary since the samples originate from the same source. Ten speakers are used to evaluate the performance.

Goal

The aim of this experiment is to observe the behavior of *dominant set* on a stable recording environment.

Expectation

If the embeddings represent a speaker's identity, *dominant set* is expected to be able to cluster the segments to the correct speaker.

Setup

The TIMIT dataset is used since a stable environment and embeddings which properly represent a speaker are needed. The TIMIT-network extracts the embeddings because according to the embedding experiments, the embeddings extracted by this experiment represents best a speaker. As only a subset of TIMIT is needed, a generated dataset of the embedding experiments is used. This dataset contains ten speakers, each with eight utterances. The hyperparameters of *dominant set* are set to $\epsilon = 1e-6$, $\theta = 0.15$ and metric = *cosine*. Scatter plots are used to evaluate the performance, where one visualizes the ground truth and the other the clustered (predicted) result.

Results

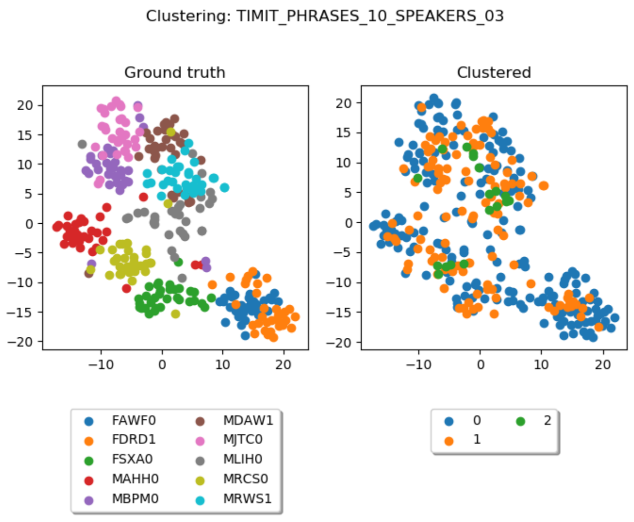


Figure 4.16: Clustering of ten speakers in similar recording conditions

The ground truth shows a cluster for each speaker, whereas in the result only three clusters are present.

Interpretation

This experiment indicates that *dominant set* is not able to attribute the embeddings to the speakers. Even in a stable environment where the embeddings represent a speaker, the clustering is not working as expected.

General interpretation These experiments show that *dominant set* does not cluster the same way as *t-SNE* does visually. Interestingly enough, neither does it perform well on the TIMIT-dataset, where Felix Hibraj et al. reported a misclassification rate of 0.0 [23]. The key difference to their system is the neural network, which provides the necessary embeddings. Instead of an LSTM-network, they used convolutional neural networks (CNN), namely the *luvo* neural network [7] and the VGGVox-CNN [10]. Both networks provide higher dimensional embeddings than the embeddings generated using the *pairwise_blstm*.

4.2.8 Experiment #3.4

In this experiment *dominant set* is tested with several hyperparameters to demonstrate the effects they induce on the clustering performance.

Goal

The goal of this experiment is to execute *dominant set* with different settings (grid search) to identify possible flaws and to observe the effect of these hyperparameters.

Expectation

According to Felix Hibraj et al. [23] *dominant set* has a large area with possible hyperparameters where the performance of the algorithm is stable. Hence, this behavior is expected in this experiment too. The stable performance of *dominant set* indicates that the cluster performance will not increase significantly without an optimal setting. This depends primarily on the used feature vectors (embeddings). Therefore an improvement of the clustering performance is not expected in almost any of the tested hyperparameter configuration.

Setup

A grid search is performed to test *dominant set* with different settings. The search uses combinations of the hyperparameter ϵ and θ to evaluate the performance of *dominant set*. For the parameters the following values are used: $\epsilon \in \{1e-11, 1e-10, \ldots, 1e-2\}$ and $\theta \in \{0.0, 0.0005, \ldots, 0.9995\}$. This search space was also used in the sensitivity analysis of *dominant set*, executed by Felix Hibraj et al. [23] In order to evaluate the clustering performance, the misclassification rate (MR) is applied [12]. A mock TIMIT-dataset is used to perform the clustering. The dataset consists of ten speakers in conjunction with eight utterances each. The result is a heat map which visualizes the results of the grid search.

Results

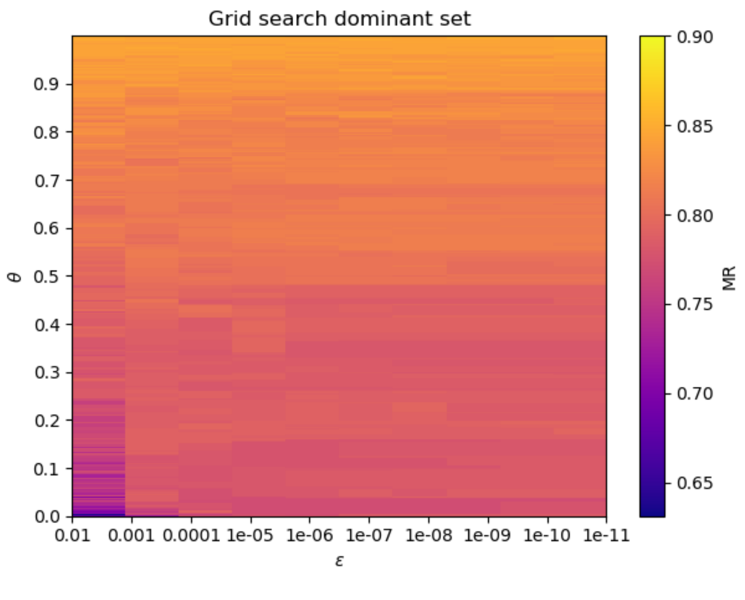


Figure 4.17: Grid search results of *dominant set*

Figure 4.17 illustrates that the clustering performance is stable for almost all hyperparameter combinations. The best results yield the following setup: $\epsilon = 1e-2$ and $\theta = 0.0005$ with a MR of 0.63 (lower left corner of figure).

Interpretation

The results show that *dominant set* has a large area where it produces consistent results, which means the hyper-parameters do not have a big impact on the performance. In general the performance is not great, since the MR is between 0.8 and 0.85 for most combinations. This leads to the conclusion that *dominant set* strongly depends on the feature vectors. Therefore, to improve the clustering performance, the feature vectors have to be improved for the clustering task.

4.2.9 Experiment #4.1

Description

This experiment serves to assess the baseline for the diarization error rate. The baseline should represent the performance of "guessing" as well as possible.

Goal

For each available RT-09 meeting recording the baseline should be evaluated.

Expectation

During the development process, many runs were performed on the RT-09 dataset to evaluate the progress. Based on these runs one could expect to have a baseline for each meeting recording around 0.85.

Setup

The experiment was run for each meeting ten times, each with a different random seed (the seeds have been generated randomly using RANDOM.org[3]).
The pseudocode listing 4.1 is an illustration of how the experiment is set up.

```
DATASETS = ['EDI_20071128-1000', 'EDI_20071128-1500', 'IDI_20090128
    -1600', 'IDI_20090129-1000', 'NIST_20080201-1405', 'NIST_20080227
    -1501', 'NIST_20080307-0955']
SEEDS = [30938, 29772, 24801, 31810, 44803, 46861, 42649, 33132,
    25087, 18062]

for seed in SEEDS:
    np.random.seed(seed)
    for dataset in DATASETS:
        load_embeddings()
        define_cluster_count()
        randomly_assign_clusters_to_embeddings()
        randomly_pick_one_cluster_as_silence()
        calculate_metrics()
```

Listing 4.1: Pseudocode Random Experiment

[3]https://www.random.org/

With such a complex system as the diarization system, one cannot simply tell a function to act random instead of normal. Therefore, the experiment's purpose is to recreate the functionality of the system. The following criteria serve to outline the conditions after which the "random system" obeys. These criteria also act as the blueprint for the baseline evaluation process:

- Randomly decide how many clusters (speakers) will be in the audio recording (random pick from [2..20])
- Random cluster assignment to each segment
- Randomly select one cluster as "non-speech"

Results

The table 4.3 shows the summarized performance of guessing (random) according to the previously described experiment.

Meeting	Mean DER	Mean Purity	Mean Coverage	Mean F-Score
EDI_20071128-1000	0.7578	0.3967	0.5223	0.4366
EDI_20071128-1500	0.8014	0.4172	0.6348	0.4944
IDI_20090128-1600	0.7654	0.3771	0.5199	0.4214
IDI_20090129-1000	0.7881	0.3627	0.6023	0.4296
NIST_20080201-1405	0.7556	0.3736	0.6356	0.4449
NIST_20080227-1501	0.7782	0.3774	0.7842	0.4906
NIST_20080307-0955	0.7608	0.3497	0.5156	0.3931
Overall Mean	0.7724	0.3792	0.6021	0.4444

Table 4.3: Random Performance on RT-09

Detailed results of this experiment can be found in the appendix chapter A.3.1.

Interpretation

The results of this experiment are quite interesting. They give a rough understanding of how well the system should perform at least. Nevertheless, these results should be considered as pseudo-random since there are some constraints on the algorithm. For instance, there is a limitation on the possible amount of clusters (speakers) and the assumption that only one cluster is considered as silence. In real diarization, a voice activity detection component could probably flag multiple clusters as silence. A really random experiment (without constraints) may perform much worse. Nonetheless, these results are still realistic, since a meeting usually does not involve an indefinite amount of people. Most of the time a meeting consists of 2 to 20 participants.

4.3 Overall Performance

In this section a closer look is taken at the system's performance from A to Z. As already shown in the embedding experiments, the results highly depend on the input, which is why multiple runs with different configurations are needed to examine the performance.
Handcrafted TIMIT "dummy datasets" are used which are fairly easy to work with to get an understanding how good the system could perform. The "dummy datasets" consist of various utterances. There are two different datasets being used[4]:

1. TIMIT / TIMIT_DUMMY-01: This is a handcrafted dataset with 5 different speakers and one phrase (roughly 3s) each. It consists of handpicked speakers which sound very distinct. This dataset can be regarded as the easiest for speaker diarization. The dataset has a total length of 30s of evaluable time as it **always** contains silence after each phrase.
2. TIMIT_RND / TIMIT_DUMMY: This dataset is similar to the other TIMIT dummy. However, this dataset consists of 10 speakers which were randomly picked from the whole TIMIT dataset. There can be multiple spoken phrases originating from the same speaker. The dataset has roughly 75s of evaluable length as it **can** also contain silence after a single phrase.

4.3.1 Evaluation

Benchmarking the system is done by performing multiple runs with various different configurations.

Run	Dataset	Model
1	TIMIT / TIMIT_DUMMY-01	TIMIT_BLSTM
2	TIMIT / TIMIT_DUMMY-01	VOXCELEB_BLSTM
3	TIMIT_RND / TIMIT_DUMMY	TIMIT_BLSTM
4	TIMIT_RND / TIMIT_DUMMY	VOXCELEB_BLSTM
5	RT09 / EDI_20071128-1000	TIMIT_BLSTM
6	RT09 / EDI_20071128-1000	VOXCELEB_BLSTM
7	RT09 / EDI_20071128-1500	TIMIT_BLSTM
8	RT09 / EDI_20071128-1500	VOXCELEB_BLSTM
9	RT09 / IDI_20090128-1600	TIMIT_BLSTM
10	RT09 / IDI_20090128-1600	VOXCELEB_BLSTM
11	RT09 / IDI_20090129-1000	TIMIT_BLSTM
12	RT09 / IDI_20090129-1000	VOXCELEB_BLSTM
13	RT09 / NIST_20080201-1405	TIMIT_BLSTM
14	RT09 / NIST_20080201-1405	VOXCELEB_BLSTM
15	RT09 / NIST_20080227-1501	TIMIT_BLSTM
16	RT09 / NIST_20080227-1501	VOXCELEB_BLSTM
17	RT09 / NIST_20080307-0955	TIMIT_BLSTM
18	RT09 / NIST_20080307-0955	VOXCELEB_BLSTM

Table 4.4: Description of the various runs

As shown in table 4.4, the runs are all possible combinations of available datasets and the trained models for the *pairwise_blstm* network. For the overall performance experiment, these runs were performed multiple times, each with a different cutoff value for the sequencer. The first four runs are sanity runs on the TIMIT "dummy datasets". Since the experiment in chapter 4.2.4 has shown that "not-same speaker" and "same speaker" embeddings are not easily be told apart, multiple conservative cutoff values were chosen (0.0, 0.05, 0.1).
As clustering configuration, the parameters are set to $\epsilon = 1e-6, \theta = 0.15$ and metric=*cosine*, using the initialization method "flag both". The detailed results can be found in the appendix chapter A.3.2.

[4]More details in Appendix Chapter A.4.1

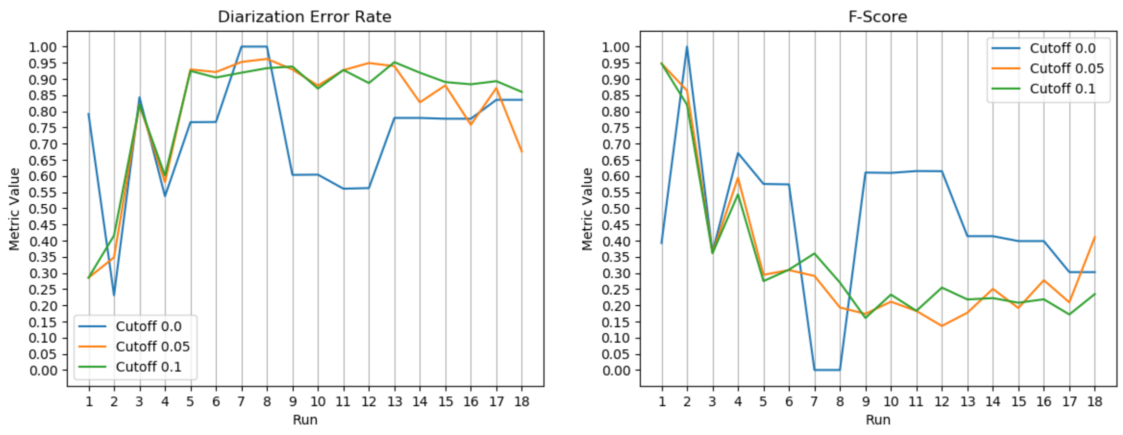


Figure 4.18: Overall Performance DER & F-Score

In figure 4.18 the diarization error rate (left) and the F-score (right) are depicted. All runs as specified in table 4.4 are listed on the x-axis of the figures 4.18 and 4.19.

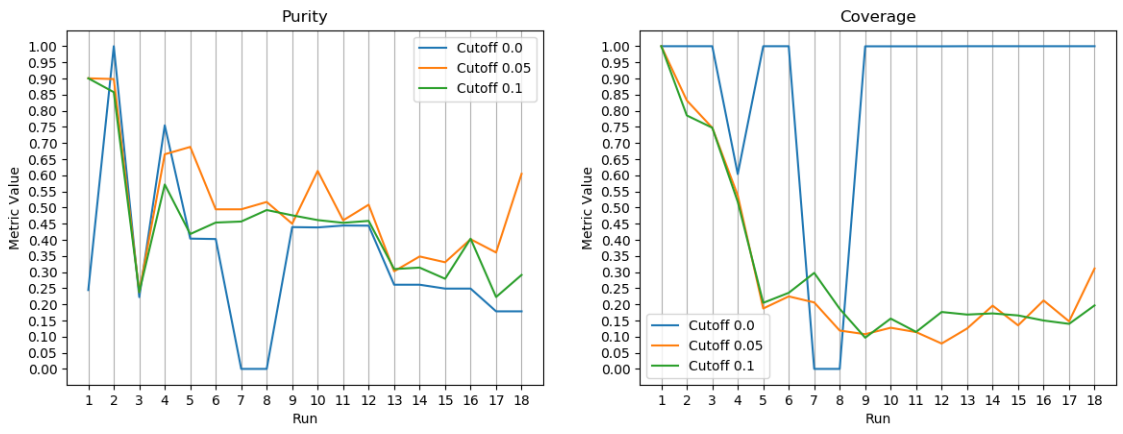


Figure 4.19: Overall Performance Purity & Coverage

In the figure 4.19 the purity (left) and the coverage (right) scores are shown. The plots show the behavior of over-clustering as described in the chapter 3.2.8.

Meeting	Ground Truth	TIMIT Model			VoxCeleb Model		
		0.0	0.05	0.1	0.0	0.05	0.1
TIMIT / TIMIT_DUMMY-01	5	1	4	4	4	5	5
TIMIT_RND / TIMIT_DUMMY-01	10	1	2	2	11	11	10
EDI_20071128-1000	4	1	23	8	1	13	6
EDI_20071128-1500	4	0	22	8	0	24	8
IDI_20090128-1600	5	2	26	39	2	46	14
IDI_20090129-1000	4	2	25	20	2	46	17
NIST_20080201-1405	5	1	29	34	1	19	20
NIST_20080227-1501	6	1	26	15	1	18	29
NIST_20080307-0955	11	1	31	18	1	30	16

Table 4.5: Overall speaker clustering results (No. of predicted speakers)

The table 4.5 shows the amount of speakers detected by the two different models. When using a cutoff value greater than 0.0, the models clearly over-cluster which correlates with the purity and coverage plots in figure 4.19.

4.3.2 Interpretation

General The results as shown in figures 4.18 and 4.19 are almost as expected. The runs 1 to 4 are obviously performing really well as they are performed on the TIMIT "dummy datasets" (low DER and high F-score).
The runs on the RT09 dataset are somewhat expected, since the previous experiments showed, that the components of the system cannot produce great output. The system seems to over-cluster (high purity and low coverage) quite a lot on the RT09 dataset, which is also shown in table 4.5. The only exceptions are the runs performed with a cutoff value of 0.0, where the exact opposite is the case. The performance for the runs 9 to 12 (IDI meetings) using a cutoff value of 0.0 are thus surprising. Obviously, a DER in the range of 0.5 and 0.6 is not great compared to the state of the art systems, but the performance is really good in comparison to the other runs.

Clustering The over-clustering runs (cutoff value 0.05 and 0.1) occurred probably due to the fact that the clustering algorithm is initialized with a modified distance matrix. Initializing the clustering algorithm does more harm than good in this case. Since a cutoff value of 0.0 does the exact opposite to the clustering result, it can be assumed that without proper and careful initialization of the clustering algorithm, no good results can be expected. Obviously, the runs with a cutoff value of 0.0 are looking good DER-wise, but this is rather by accident than by intention. As shown in table 4.5 most of the meetings do have 4-6 participants, which are fairly close to the predicted cluster count. However, since the clustering component clusters almost all runs to 1 or 2 clusters, this can only be assumed as a mistake.

5 Discussion

This chapter takes a step back on the presented work and reviews the results. Possible follow up project ideas are listed at the end of this chapter.

5.1 Project Scope & Goals

This project aimed to match the performance of current state-of-the-art systems. Looking at the results described in chapter 4.3 this is not the case. Nevertheless, the goal as described in chapter 1.2 can be considered as reached. The following reasons contribute to this statement:

- A system was built that can be benchmarked using the NIST RT-09 evaluation campaign, which makes it comparable to other state-of-the-art systems.
- Problems in the pipeline are outlined in the experiment's chapter 4.2, which allow further improvements of the system.
- All of this was developed in a relatively short timeframe compared to the years of research used in the state-of-the-art systems, which shows how well these new approaches (deep learning) work for this task.

5.2 Learnings

During the course of this project many learnings became clear. Some were related to machine learning and audio processing and some others are more interdisciplinary and more abstract. These learnings should serve as good takeaway messages for future projects.

- It really helps to get your hands early on the datasets and get a good understanding of how it is structured. Ideally, get it ready before the development starts, otherwise you will be refactoring a lot.
- Build the pipeline quickly and then refine it. Do not work component-wise, but build it as a whole and then flesh out the components. Employ mocking of components where necessary.
- Do not get sidetracked when fleshing out the components. Stick to the plan. Too much time was spent trying to improve the clustering algorithm as a whole with various machine learning techniques without implementing the original plan in the first place.

5.3 Future Work

There are a lot of potential projects and tasks for the future where one could extend the system. Based on the experience collected during this project, the following next steps are recommended.

ZHAW Deep Voice Suite The suite itself needs some extensions. Currently it is only possible to pass fixed length segments into the neural network. This should be made dynamic to make it trainable with various datasets. Also a fixed length means the system can only be as accurate in diarization as the model's segment length during training. The suite is quite tailored to the TIMIT corpus. The TIMIT corpus is made out of single speaker recordings in studio condition, whereas other datasets (i.e. RT-09) are multi-speaker recordings (conversations). Converting the RT-09 dataset into a TIMIT like dataset is possible, but not ideal. For this system to perform well, the model needs to be trained on various datasets. Maybe even new networks have to be implemented.

Embedding Extractor The embedding extractor needs to deliver better embeddings. Some of the experiments in chapter 4.2 have shown that the retrieved embeddings are not really identifying the speakers, but the room itself. One way to achieve better embeddings would probably be to normalize the training data before passing it to the neural network.

Sequencer As shown in the experiments, the idea of pre-clustering works but it needs some refinement. Not only better embeddings are required but also the algorithm of pre-clustering should be made more robust. Apart from just relying on a cutoff value in conjunction with a linkage method on previously seen segments, a lookahead method should be considered to detect outliers. A speaker change detection could also be interpreted as a binary classification problem. Another way of solving the speaker change detection problem could be to use a LSTM neural network. There has been research in this direction for example by Yin et al. [33].

Clustering The dominant set clustering algorithm has the capability to perform very well for this task. However, as the experiments have shown, it does not perform well with the *pairswise_blstm* embeddings. Therefore, experiments with another neural network should be performed. For example the CNN-network *luvo*, developed by Lukic et al. [7], would be a good choice, because *dominant set* shows a good performance using those embeddings [23].
Another thing which definitely should be experimented with is the implementation of different clustering algorithms such as *Affinity Propagation* and *(H)DBSCAN*. These algorithms have shown good performances during development while testing various architectures.
All clustering algorithms depend on some sort of metric to evaluate whether an element belongs to a cluster or not. According to a newly published article by Meier et al. proper clustering can also be learned using a neural network. [34].
Optimizing the dominant set clustering algorithm and comparing it to different clustering algorithm could possibly be the scope for a follow up bachelor thesis.

Pipeline The pipeline currently does not incorporate any input sanitization. This means the incoming audio is not being cleaned and no other preprocessing is done with the audio. One could try to clean the audio to get better Mel-spectrograms. When comparing the pipeline with other state of the art speaker diarization software, these kind of mechanisms are clearly lacking.
Furthermore, the pipeline currently only supports the SDM condition of the RT-09 evaluation campaign. One should implement the MDM condition to have a competitive system. This would require to work with various microphones. The additional condition and the research on how to deal with multiple microphones could probably be fit in a complete project on its own.

6 Listings

Bibliography

[1] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, O. Friedland, and O. Vinyals, “Speaker diarization : A review of recent research,” *"IEEE Transactions On Audio, Speech, and Language Processing" (TASLP), special issue on "New Frontiers in Rich Transcription", February 2012, Volume 20, N°2, ISSN: 1558-7916*, 05 2011.

[2] NIST, “The 2009 (rt-09) rich transcription meeting recognition evaluation plan.” `https://web.archive.org/web/20100308103107/http://www.itl.nist.gov/iad/mig/tests/rt/2009/index.html`, 2009. [Online; accessed through webarchive 2nd of April 2018].

[3] G. Friedland, A. Janin, D. Imseng, X. Anguera, L. Gottlieb, M. Huijbregts, M. T. Knox, and O. Vinyals, “The icsi rt-09 speaker diarization system,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 371–381, Feb 2012.

[4] S. Galliano, G. Gravier, and L. Chaubard, “The ester 2 evaluation campaign for the rich transcription of french radio broadcasts,” in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[5] Rouvier, Mickael, G. Dupuy, Gay, Paul, Khoury, Elie, Merlin, Teva, Meignier, and Sylvain, “An open-source state-of-the-art toolbox for broadcast news diarization,” in *Interspeech*, 2013.

[6] C. Fredouille, S. Bozonnet, and N. Evans, “The lia-eurecom rt’09 speaker diarization system,” 01 2009.

[7] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, “Speaker identification and clustering using convolutional neural networks,” in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop*, pp. 1–6, IEEE, 2016.

[8] P. Gerber and S. Glinski, “Machine learning for speaker clustering.” Bachelor Thesis, 2017.

[9] J. S. Garofolo, “The darpa timit acoustic-phonetic continuous speech corpus (timit) - training and test data.” README.doc, 1990.

[10] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: a large-scale speaker identification dataset,” in *INTERSPEECH*, 2017.

[11] J. S. Chung, A. Nagrani, and A. Zisserman, “Voxceleb2: Deep speaker recognition,” in *arXiv*, 2018.

[12] S. Niederer and B. Heusser, “Deep learning für speaker clustering.” ZHAW project thesis, 2017.

[13] Y. X. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, “Learning embeddings for speaker clustering based on voice equality,” in *Machine Learning for Signal Processing (MLSP), 2017 IEEE 27th International Workshop on*, pp. 1–6, IEEE, 2017.

[14] T. Gygax and J. Egli, “Speaker clustering mit metric embeddings.” Bachelor Thesis, 2017.

[15] T. Ganchev, N. Fakotakis, and G. Kokkinakis, “Comparative evaluation of various mfcc implementations on the speaker verification task,” in *Proceedings of the SPECOM*, vol. 1, pp. 191–194, 2005.

[16] S. S. Stevens, J. Volkmann, and E. B. Newman, “A scale for the measurement of the psychological magnitude pitch,” *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] C. Olah, "Understanding lstm networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 08 2015. [Online; accessed 9th of May 2018].

[19] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE International Conference on Neural Networks*, pp. 1183–1188 vol.3, 1993.

[20] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602 – 610, 2005. IJCNN 2005.

[21] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[22] 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.

[23] F. Hibraj, S. Vascon, T. Stadelmann, and M. Pelillo, "Speaker clustering using dominant sets," *arXiv preprint arXiv:1805.08641*, 2018.

[24] M. Pavan and M. Pelillo, "Dominant sets and pairwise clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 167–172, Jan 2007.

[25] S. Vascon, M. Cristani, M. Pelillo, and V. Murino, "Using dominant sets for k-nn prototype selection," in *Image Analysis and Processing – ICIAP 2013* (A. Petrosino, ed.), (Berlin, Heidelberg), pp. 131–140, Springer Berlin Heidelberg, 2013.

[26] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[27] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[28] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 1, p. 5, 2015.

[29] S. Vascon, "Private communication." May 2018.

[30] 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.

[31] M. Cettolo, "Segmentation, classification and clustering of an italian broadcast news corpus," in *Content-Based Multimedia Information Access-Volume 1*, pp. 372–381, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2000.

[32] H. Sun, B. Ma, S. Z. K. Khine, and H. Li, "Speaker diarization system for rt07 and rt09 meeting room audio," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4982–4985, March 2010.

[33] R. Yin, H. Bredin, and C. Barras, "Interspeech 2017, 18th Annual Conference of the International Speech Communication Association," (Stockholm, Sweden), August 2017.

[34] B. B. Meier, T. Stadelmann, and O. Dürr, "Learning to cluster," 2018.

List of Figures

List of Tables

Glossary

This chapter explains the terms (if not already explained in a previous chapter) and abbreviations used in this document.

Affinity Propagation An unsupervised clustering algorithm.

BLSTM Bidirectional Long Short Term Memory, see chapter 2.7.

CNN Convolutional Neural Network, a neural network architecture.

Complete Linkage A linkage method, see chapter 3.2.5.

Cosine A distance metric used to calculate spatial distance.

Coverage A metric which can be calculated using the metric component, see chapter 3.2.8.

Cutoff value A configurable hyperparameter used in the sequencer for pre-clustering.

DBSCAN An unsupervised clustering algorithm.

DER Diarization error rate, see chapter 3.2.8.

Dominant Sets Clustering An unsupervised clustering algorithm, see chapter 2.9.

DS / DOS Dominant Set, see chapter 2.9.

EDI Edinburgh University

Epsilon ϵ A hyperparameter for *dominant set*. See chapter 2.9.

ESTER A rich transcription evaluation campaign.

Euclidean A distance metric used to calculate spatial distance.

F-Score A metric for accuracy, see chapter 3.2.8.

Flow_me A neural network developed by students at the InIT.

GMM Gaussian Mixture Models

Grid search A technique of finding optimal hyperparameters.

HDBSCAN An improved version of DBSCAN. An unsupervised clustering algorithm.

ICSI International Computer Science Institute (USA)

IDI IDIAP Research Institute (Switzerland)

InIT Institute of Applied Information Technology (Switzerland)

LIA-EURECOM A joint venture of LIA and EURECOM to build speech recognition software.

LIUM Research Institute at University Lemans.

LSTM Long Short Term Memory, see chapter 2.7.

Luvo A neural network developed by students at the InIT.

MDM An evaluation condition for the RT09 challenge, see chapter 2.2.3.

Mel-Spectrogram A special form of a spectrogram, see chapter 2.6.

MFCC Mel Frequency Cepstral Coefficient

MR Misclassification Rate

NIST National Institute of Standards and Technology

Pairwise_blstm A neural network developed by students at the InIT.

Pairwise_kldiv A neural network developed by students at the InIT.

PC-F-Score The F-Score calculated using purity and coverage, see chapter 3.2.8.

PCA Principal Component Analysis, used as a dimensionality reduction technique in this project.

Previous Linkage A linkage method, see chapter 3.2.5.

Purity A metric which can be calculated using the metric component, see chapter 3.2.8.

RD Replicator dynamics, see chapter 2.9.

RGB A color information representation model.

RNN Recurrent Neural Network, a neural network architecture.

RT Rich Transcription, a style of transcribing audio (or video) which includes speaker markers.

RT09 A rich transcription evaluation campaign by NIST.

RT09_BLSTM The *pairwise_blstm* network model trained using RT09 data.

SASTT Speaker Attributed Speech-To-Text, a task from the RT09 evaluation campaign. See chapter 2.2.3.

SC Speaker clustering, see chapter 2.3.

SDM An evaluation condition for the RT09 challenge, see chapter 2.2.3.

Single Linkage A linkage method, see chapter 3.2.5.

SPDR Speaker Diarization, a term coined during this project.

Spectrogram A representation of audio, see chapter 2.6.

SPKR Speaker Diarization, a term coined by NIST. Also a task from the RT09 evaluation campaign. See chapter 2.2.3.

STT Speech-To-Text, a task from the RT09 evaluation campaign. See chapter 2.2.3.

t-SNE t-Distributed Stochastic Neighbor Embedding, used as a dimensionality reduction technique in this project.

Theta θ A hyperparameter for *dominant set*. See chapter 2.9.

TIMIT A speech corpus used to train neural networks. See chapter 2.2.1.

TIMIT_BLSTM The *pairwise_blstm* network model trained using TIMIT data.

VAD Voice Activity Detector, see chapter 3.2.7

VoxCeleb A speech corpus used to train neural networks. See chapter 2.2.2.

VOXCELEB_BLSTM The *pairwise_blstm* network model trained using VoxCeleb data.

WAV Waveform Audio File Format

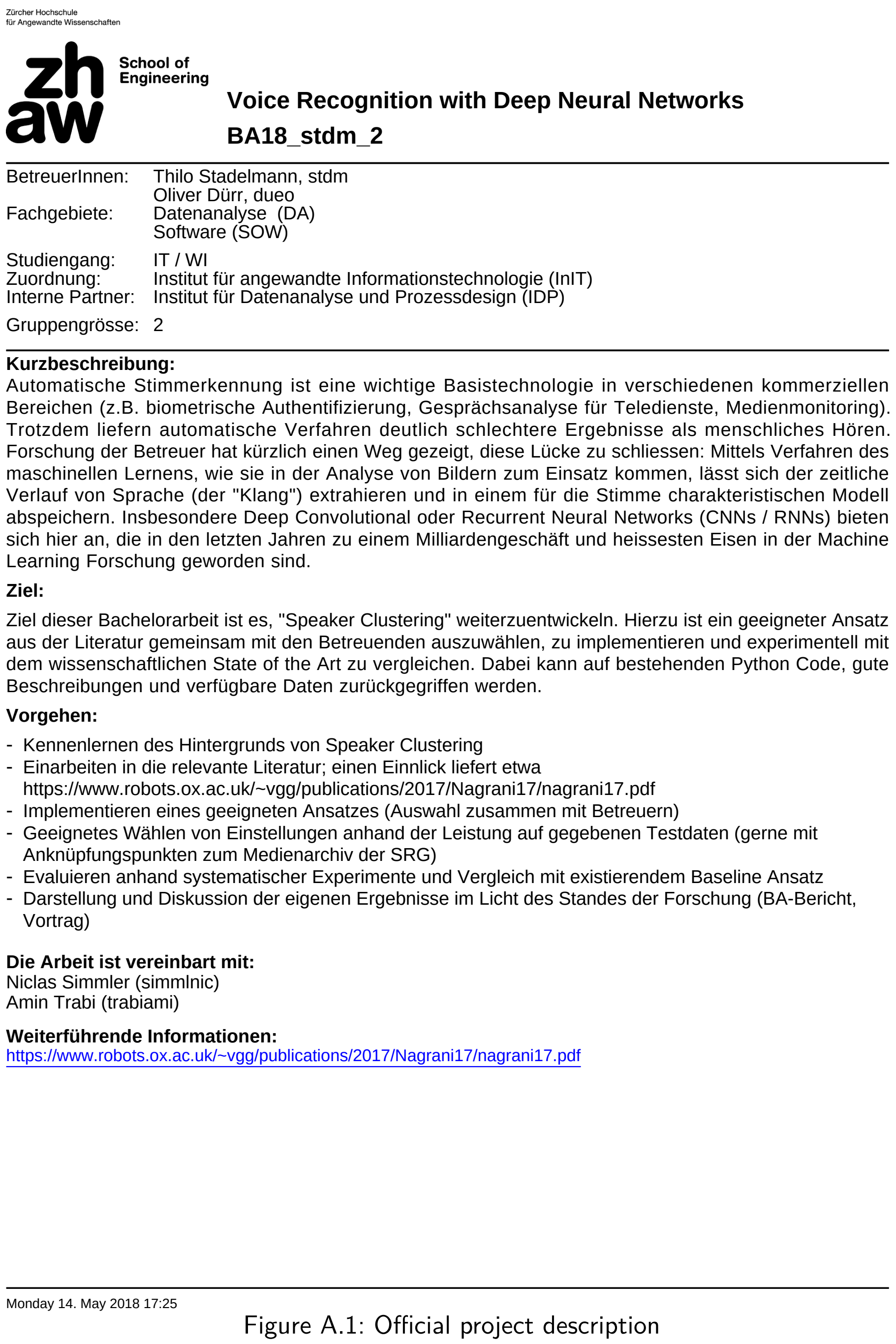
YAML / yml YAML Ain't Markup Language, a language commonly used to write configuration files. See chapter A.2.3.

ZHAW Zurich University of Applied Sciences

A Appendix

A.1 Project Management

A.1.1 Official Project Description



Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

Voice Recognition with Deep Neural Networks
BA18_stdm_2

BetreuerInnen:	Thilo Stadelmann, stdm Oliver Dürr, dueo
Fachgebiete:	Datenanalyse (DA) Software (SOW)
Studiengang:	IT / WI
Zuordnung:	Institut für angewandte Informationstechnologie (InIT)
Interne Partner:	Institut für Datenanalyse und Prozessdesign (IDP)
Gruppengrösse:	2

Kurzbeschreibung:
Automatische Stimmerkennung ist eine wichtige Basistechnologie in verschiedenen kommerziellen Bereichen (z.B. biometrische Authentifizierung, Gesprächsanalyse für Teledienste, Medienmonitoring). Trotzdem liefern automatische Verfahren deutlich schlechtere Ergebnisse als menschliches Hören. Forschung der Betreuer hat kürzlich einen Weg gezeigt, diese Lücke zu schliessen: Mittels Verfahren des maschinellen Lernens, wie sie in der Analyse von Bildern zum Einsatz kommen, lässt sich der zeitliche Verlauf von Sprache (der "Klang") extrahieren und in einem für die Stimme charakteristischen Modell abspeichern. Insbesondere Deep Convolutional oder Recurrent Neural Networks (CNNs / RNNs) bieten sich hier an, die in den letzten Jahren zu einem Milliardengeschäft und heissesten Eisen in der Machine Learning Forschung geworden sind.

Ziel:

Ziel dieser Bachelorarbeit ist es, "Speaker Clustering" weiterzuentwickeln. Hierzu ist ein geeigneter Ansatz aus der Literatur gemeinsam mit den Betreuenden auszuwählen, zu implementieren und experimentell mit dem wissenschaftlichen State of the Art zu vergleichen. Dabei kann auf bestehenden Python Code, gute Beschreibungen und verfügbare Daten zurückgegriffen werden.

Vorgehen:

- Kennenlernen des Hintergrunds von Speaker Clustering
- Einarbeiten in die relevante Literatur; einen Einnlick liefert etwa https://www.robots.ox.ac.uk/~vgg/publications/2017/Nagrani17/nagrani17.pdf
- Implementieren eines geeigneten Ansatzes (Auswahl zusammen mit Betreuern)
- Geeignetes Wählen von Einstellungen anhand der Leistung auf gegebenen Testdaten (gerne mit Anknüpfungspunkten zum Medienarchiv der SRG)
- Evaluieren anhand systematischer Experimente und Vergleich mit existierendem Baseline Ansatz
- Darstellung und Diskussion der eigenen Ergebnisse im Licht des Standes der Forschung (BA-Bericht, Vortrag)

Die Arbeit ist vereinbart mit:
Niclas Simmler (simmlnic)
Amin Trabi (trabiami)

Weiterführende Informationen:
https://www.robots.ox.ac.uk/~vgg/publications/2017/Nagrani17/nagrani17.pdf

Monday 14. May 2018 17:25

Figure A.1: Official project description

A.1.2 Workload Distribution

Since this project is intended to be completed by a team of two, the workload needs to be organized. There are topics and components which are mainly developed by one team member. The following list should give an idea who got the lead or is the expert on which part of this system:

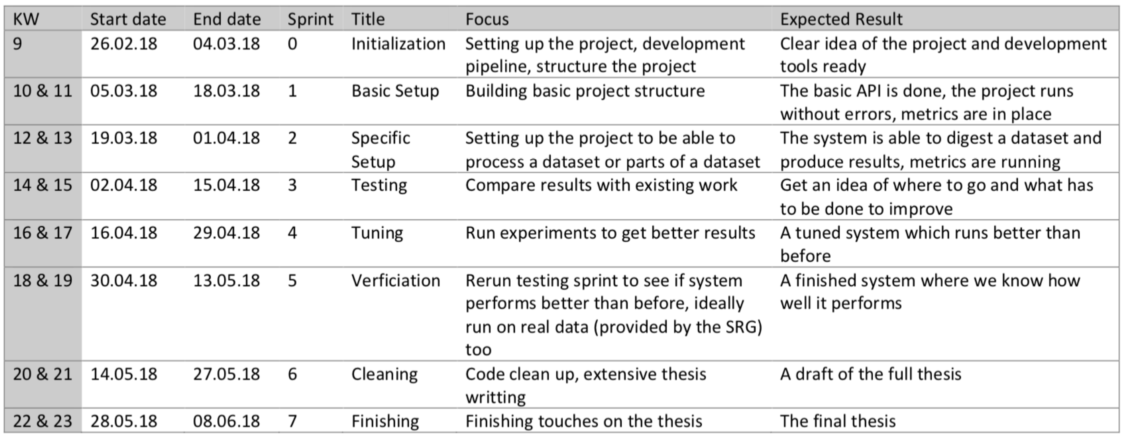
Amin Trabi Segmenter, Embedding Extractor and ZHAW Deep Voice Suite Setup, Dominant Set Clustering Algorithm, Voice Activity Detector

Niclas Simmler Pipeline Setup, Dataset Procurement and Handling, Sequencer (Preclustering), Metrics

Note: Both were working on all parts of the system. However, the list shows who had the lead on each part.

A.1.3 Time Schedule

This thesis is due to be handed in by the 8th of June 2018. The work is organized using sprints. Sprints, in this case, are thematic groupings of tasks and last two weeks.



KW	Start date	End date	Sprint	Title	Focus	Expected Result
9	26.02.18	04.03.18	0	Initialization	Setting up the project, development pipeline, structure the project	Clear idea of the project and development tools ready
10 & 11	05.03.18	18.03.18	1	Basic Setup	Building basic project structure	The basic API is done, the project runs without errors, metrics are in place
12 & 13	19.03.18	01.04.18	2	Specific Setup	Setting up the project to be able to process a dataset or parts of a dataset	The system is able to digest a dataset and produce results, metrics are running
14 & 15	02.04.18	15.04.18	3	Testing	Compare results with existing work	Get an idea of where to go and what has to be done to improve
16 & 17	16.04.18	29.04.18	4	Tuning	Run experiments to get better results	A tuned system which runs better than before
18 & 19	30.04.18	13.05.18	5	Verficiation	Rerun testing sprint to see if system performs better than before, ideally run on real data (provided by the SRG) too	A finished system where we know how well it performs
20 & 21	14.05.18	27.05.18	6	Cleaning	Code clean up, extensive thesis writting	A draft of the full thesis
22 & 23	28.05.18	08.06.18	7	Finishing	Finishing touches on the thesis	The final thesis

Figure A.2: Project Schedule

A.2 Speaker Diarization Suite

A.2.1 Structure

```
/
├─ data
│  ├─ in
│  ├─ out
│  ├─ pickles
│  ├─ plots
│  ├─ runs
│  └─ vad_models
├─ experiments
├─ spdr
├─ tests_and_examples
├─ ZHAW_deep_voice
├─ controller.py
├─ setup_conda.sh
└─ runner.py
```

The structure of the speaker diarization folder is quite simple.

- One of the most important folders is the *data* folder.
  - In the *in* subfolder the datasets can be found.
  - In the *out* subfolder the fixed length segments used by the embedding extractor will be stored.
  - In the *pickles* subfolder the embeddings for quicker re-runs will be dumped.
  - In the *plots* subfolder any kind of plots produced by the system or experiments are archived.
  - In the *runs* subfolder the results of the remotely executed runs by the *runner.py* file are stored.
  - In the *vad_models* the trained GMM models used inside the voice activity detection component should be placed
- Code which was used for creating the plots or running the experiments as described in the experiments chapter can be found in the folder *experiments*.
- The whole (productive) code can be found inside the *spdr* folder.
- During development there was the need for isolated testing. In the *tests_and_examples* folder many scripts are available, which show example usages or test some behavior.
- Inside the *ZHAW_deep_voice* folder resides the ZHAW_deep_voice suite (including the trained models).
- The *controller.py* file is the main entry point for the suite.
- Using the *setup_conda.sh* file one can set up a dedicated python environment with all the required libraries preinstalled.
- The *runner.py* file can be used to remotely trigger multiple runs by specifying multiple run conditions for the *controller.py* file.

A.2.2 Setup

The suite can be installed using the following steps.

1. One needs to install the python environment using the *setup_conda.sh* script. After running the script a python environment called *spdr* will be present, with all the required packages installed.
2. The required dataset needs to be placed inside the *./data/in/* folder.
3. The *ZHAW_deep_voice* suite needs to be set up. One needs to either train or place a fully trained model in the correct place. As of the time of writing this thesis, it would be under *./ZHAW_deep_voice/common/data/experiments/nets/*. For further information consult the documentation of Niederer et al.[12]

A.2.3 Config.yml - Suite Configuration

The system can be configured using a *config.yml* file. The *SPDR_Utils* class offers an easy to use handle (n-dimensional-array) to access the configuration. Since the configuration file is a YAML[5] file, the configuration can easily be modified. For the sake of readability the configuration parameters are shown in dot-notation instead of a YAML representation. When using the *SPDR_Utils* class for accessing the configuration an (multidimensional) array notation has to be used.

Configuration	Possible Values	Description
clustering.algorithm	dominantset	The clustering algorithm to be used in the clustering component. Only "dominantset" is implemented yet.
clustering.metric	cosine, euclidean	The metric to be used during clustering. "cosine" is recommended as it has shown the best results during the experiments.
clustering.dominantset.cutoff	Float 0..1	A hyper-parameter for the dominant set algorithm
clustering.dominantset.epsilon	Float 0..1	A hyper-parameter for the dominant set algorithm.
clustering.dominantset.apply_similarity_mode	similar, different, full	A hyper-parameter for the dominant set algorithm. Defines how the distance matrix is modified for the pre-clustered results generated using the sequencer.
data.dataset_original	String	A substring used to replace paths when parsing a dataset. Used for the RT09 dataset. Not a real config parameter. For the RT09 this should be "./audio/eval09/english/confmtg/"
data.datasubset_to_use	String	For the RT09 this would be the meeting.
data.in	String of a path	The location of the dataset. Normally this path would be a subfolder located in the *data/in* folder.
data.out	String of a path	The location of a temporary out folder. Normally this path would be the *data/out* folder.
data.pickle	String of a path	The location of the pickle folder. Normally this path would be the *data/pickles* folder.

Continued on next page

Table A.1: Config.yml Configuration Parameters

[5] http://yaml.org/

Continued from previous page

Configuration	Possible Values	Description
data.use_normalized_audio	Boolean flag	A flag to use the normalized audio. The script: *./normalize_audio.sh* have to be executed to normalize the audio.
embeddings.network	String	The filename of the trained model to be used for the ZHAW_Deep_Voice.
hypothesis.scaledown	Boolean flag	If set to true, the system converts *ms* to *s* when parsing a dataset. Can be handy when ground truth is using *ms*.
segment.cap	Number	The maximum amount of segments the segmenter will create. It is not recommended to go below *length of file in ms/segment.size*.
segment.size	Number	The size of segments to be created. This corresponds to the segment size for the neural network. 500ms is recommended for current trained models.
sequence.cutoff	Float 0..1	The cutoff value used for pre-clustering.
sequence.linkage	single, complete, last	The linkage method used for pre-clustering.
sequence.metric	cosine, euclidean	The metric used for pre-clustering.
spkr.condition	SDM	The RT09 condition on which to evaluate. MDM is not supported, but could be implemented in the future.
vad.aggressiveness	low, normal, high	Defines the aggressiveness of the VAD. When set to high a cluster only has to contain 25% of silence to be classified as silence. Normal requires 50% and low requires 75% of silence.

Table A.1: Config.yml Configuration Parameters

A.3 Experiments

The following chapter lists the detailed results collected while running some of the experiments.

A.3.1 Experiment 4.1

These results were collected during the run of experiment 4.1 (the "random" experiment).

Meeting	Run	DER	Purity	Coverage	F-Score
EDI_20071128-1000	1	0.7427	0.4916	0.6814	0.5711
	2	0.8092	0.3974	0.4588	0.4259
	3	0.7222	0.4088	0.4716	0.4379
	4	0.7976	0.4602	0.3757	0.4137
	5	0.8179	0.2943	0.5879	0.3922
	6	0.7531	0.2742	0.6365	0.3833
	7	0.7744	0.2745	0.5977	0.3762
	8	0.6492	0.5053	0.4700	0.4870
	9	0.8390	0.3983	0.5824	0.4731
	10	0.6726	0.4620	0.3614	0.4056
EDI_20071128-1500	1	0.7701	0.4444	0.5664	0.4981
	2	0.7519	0.2809	0.4181	0.3360
	3	0.7065	0.3208	0.5474	0.4045
	4	0.8402	0.2878	0.7630	0.4180
	5	1.0000	1.0000	1.0000	1.0000
	6	0.8646	0.4270	0.6446	0.5137
	7	0.7174	0.4088	0.6540	0.5031
	8	0.8019	0.4139	0.6399	0.5026
	9	0.8546	0.2672	0.5668	0.3632
	10	0.7065	0.3208	0.5474	0.4045
IDI_20090128-1600	1	0.7744	0.2745	0.5977	0.3762
	2	0.6492	0.5053	0.4700	0.4870
	3	0.8556	0.3983	0.3424	0.3682
	4	0.6726	0.4620	0.3614	0.4056
	5	0.6828	0.4602	0.5420	0.4977
	6	0.8546	0.2672	0.5668	0.3632
	7	0.7531	0.2742	0.6365	0.3833
	8	0.7744	0.2745	0.5977	0.3762
	9	0.8259	0.4568	0.5851	0.5130
	10	0.8110	0.3983	0.4998	0.4433
IDI_20090129-1000	1	0.6794	0.4485	0.4928	0.4696
	2	0.7394	0.4602	0.4388	0.4493
	3	0.8312	0.2792	0.5801	0.3770
	4	0.7065	0.3208	0.5474	0.4045
	5	0.8402	0.2878	0.7630	0.4180
	6	0.7241	0.3951	0.8800	0.5454
	7	0.8385	0.4066	0.3786	0.3921
	8	0.8692	0.3014	1.0000	0.4632
	9	0.7976	0.4602	0.3757	0.4137
	10	0.8546	0.2672	0.5668	0.3632
NIST_20080201-1405	1	0.7065	0.3208	0.5474	0.4045
	2	0.8498	0.2951	0.7591	0.4250
	3	0.6492	0.5053	0.4700	0.4870

Continued on next page

Table A.2: Results of random performance experiments

Continued from previous page

Meeting	Run	DER	Purity	Coverage	F-Score
	4	0.8652	0.3983	0.3455	0.3700
	5	0.6726	0.4620	0.3614	0.4056
	6	0.6677	0.4539	0.8403	0.5894
	7	0.8562	0.2647	1.0000	0.4185
	8	0.7700	0.3072	0.6573	0.4187
	9	0.8402	0.2878	0.7630	0.4180
	10	0.6785	0.4407	0.6124	0.5125
NIST_20080227-1501	1	0.9083	0.3682	0.8826	0.5196
	2	0.7959	0.4671	1.0000	0.6368
	3	0.6757	0.4444	0.8637	0.5869
	4	0.8423	0.2631	1.0000	0.4165
	5	0.7416	0.2857	0.6966	0.4052
	6	0.8298	0.2191	0.7639	0.3405
	7	0.7084	0.4108	0.7104	0.5205
	8	0.9070	0.3929	1.0000	0.5642
	9	0.5752	0.4620	0.5495	0.5020
	10	0.7976	0.4602	0.3757	0.4137
NIST_20080307-0955	1	0.7519	0.2809	0.4181	0.3360
	2	0.7941	0.2781	1.0000	0.4352
	3	0.7744	0.2745	0.5977	0.3762
	4	0.7192	0.4501	0.4973	0.4725
	5	0.8652	0.4153	0.3424	0.3753
	6	0.6726	0.4620	0.3614	0.4056
	7	0.7976	0.4602	0.3757	0.4137
	8	0.7519	0.2809	0.4181	0.3360
	9	0.7065	0.3208	0.5474	0.4045
	10	0.7744	0.2745	0.5977	0.3762

Table A.2: Results of random performance experiments

A.3.2 Overall Performance

The table A.3 shows the results of all the experiments we performed to measure the overall performance. The name of a run is defined by its DATASET (TIMIT or RT09), the SUBSET (for RT09 this would be the different meetings) and the used MODEL for the *pairwise_blstm* network. For the sake of completeness, we also included the results for the performance on our dummy datasets.

#	Name	Metric	Cutoff		
			0.0	0.05	0.1
1	DATASET_timit. SUBSET_timit_dummy-01. MODEL_pairwise_lstm_100_best	DER	0.7915	0.2853	0.2853
		Purity	0.2444	0.9005	0.9005
		Coverage	1	1	1
		F-Score	0.3927	0.9476	0.9476
2	DATASET_timit. SUBSET_timit_dummy-01. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.2309	0.3482	0.4159
		Purity	1	0.8982	0.8576
		Coverage	1	0.8321	0.7855
		F-Score	1	0.8639	0.82

Continued on next page

Table A.3: Results of overall performance experiments

Continued from previous page

#	Name	Metric	Cutoff		
			0.0	0.05	0.1
3	DATASET_timit_rnd. SUBSET_timit_dummy. MODEL_pairwise_lstm_100_best	DER	0.8435	0.8208	0.8208
		Purity	0.2225	0.2381	0.2381
		Coverage	1	0.7474	0.7474
		F-Score	0.364	0.3611	0.3611
4	DATASET_timit_rnd. SUBSET_timit_dummy. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.5373	0.5809	0.6012
		Purity	0.7548	0.6651	0.5715
		Coverage	0.6037	0.5376	0.5178
		F-Score	0.6708	0.5946	0.5433
5	DATASET_rt09. SUBSET_edi_20071128-1000. MODEL_pairwise_lstm_100_best	DER	0.7663	0.9294	0.9248
		Purity	0.4038	0.6879	0.4179
		Coverage	1	0.1872	0.2047
		F-Score	0.5753	0.2943	0.2748
6	DATASET_rt09. SUBSET_edi_20071128-1000. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.7669	0.9213	0.9043
		Purity	0.4025	0.4945	0.4535
		Coverage	1	0.2247	0.2358
		F-Score	0.5739	0.309	0.3103
7	DATASET_rt09. SUBSET_edi_20071128-1500. MODEL_pairwise_lstm_100_best	DER	1	0.9525	0.9193
		Purity	0	0.4945	0.4567
		Coverage	0	0.206	0.2975
		F-Score	0	0.2909	0.3603
8	DATASET_rt09. SUBSET_edi_20071128-1500. MODEL_pairwise_lstm_1251_voxceleb_best	DER	1	0.9619	0.9333
		Purity	0	0.5173	0.4923
		Coverage	0	0.1188	0.1862
		F-Score	0	0.1932	0.2703
9	DATASET_rt09. SUBSET_idi_20090128-1600. MODEL_pairwise_lstm_100_best	DER	0.6034	0.9293	0.9384
		Purity	0.4394	0.4499	0.476
		Coverage	0.9997	0.1076	0.0967
		F-Score	0.6105	0.1737	0.1608
10	DATASET_rt09. SUBSET_idi_20090128-1600. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.6041	0.8793	0.8701
		Purity	0.4384	0.6135	0.4612
		Coverage	0.9997	0.1275	0.1559
		F-Score	0.6095	0.2112	0.2331
11	DATASET_rt09. SUBSET_idi_20090129-1000. MODEL_pairwise_lstm_100_best	DER	0.5606	0.9272	0.9277
		Purity	0.4443	0.4604	0.4527
		Coverage	0.9997	0.114	0.1146
		F-Score	0.6152	0.1827	0.1829
12	DATASET_rt09. SUBSET_idi_20090129-1000. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.5625	0.9492	0.8872
		Purity	0.4439	0.5088	0.4587
		Coverage	0.9997	0.0786	0.1764
		F-Score	0.6148	0.1362	0.2548
13	DATASET_rt09. SUBSET_nist_20080201-1405. MODEL_pairwise_lstm_100_best	DER	0.7794	0.9395	0.9518
		Purity	0.2609	0.3026	0.3095
		Coverage	1	0.1253	0.1684
		F-Score	0.4138	0.1772	0.2182
14	DATASET_rt09. SUBSET_nist_20080201-1405. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.7794	0.8276	0.9193
		Purity	0.2609	0.3484	0.3139
		Coverage	1	0.196	0.172
		F-Score	0.4138	0.2508	0.2223

Continued on next page

Table A.3: Results of overall performance experiments

Continued from previous page

#	Name	Metric	Cutoff		
			0.0	0.05	0.1
15	DATASET_rt09. SUBSET_nist_20080227-1501. MODEL_pairwise_lstm_100_best	DER	0.7769	0.8801	0.8903
		Purity	0.2489	0.3305	0.2793
		Coverage	1	0.135	0.1657
		F-Score	0.3986	0.1917	0.208
16	DATASET_rt09. SUBSET_nist_20080227-1501. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.7769	0.7583	0.8834
		Purity	0.2489	0.4019	0.4036
		Coverage	1	0.2121	0.15
		F-Score	0.3986	0.2777	0.2187
17	DATASET_rt09. SUBSET_nist_20080307-0955. MODEL_pairwise_lstm_100_best	DER	0.8354	0.8719	0.8931
		Purity	0.1784	0.3607	0.2228
		Coverage	1	0.1473	0.1396
		F-Score	0.3027	0.2092	0.1717
18	DATASET_rt09. SUBSET_nist_20080307-0955. MODEL_pairwise_lstm_1251_voxceleb_best	DER	0.8354	0.6755	0.8598
		Purity	0.1784	0.6046	0.2912
		Coverage	1	0.3111	0.1968
		F-Score	0.3027	0.4108	0.2349

Table A.3: Results of overall performance experiments

A.4 Various

A.4.1 Reference TIMIT Dummy Datasets

As mentioned in chapter 4.3 two dummy datasets consisting of snippets from the TIMIT corpus were created.

TIMIT / TIMIT_DUMMY-01

Dataset name	TIMIT
Subset name	TIMIT_DUMMY-01
Speaker count	5
Total length	33.3s
Segments per speaker	1
Segments seperated by silence	Yes
Remarks	Speakers are handpicked to sound distinct
Diarization Difficulty	Very easy

Table A.4: TIMIT / TIMIT_DUMMY-01 Characteristics

```
3000.000 6027.000 S5
9027.000 13315.000 S3
16315.000 19873.000 S1
22873.000 26988.000 S4
29988.000 33399.000 S2
```

Listing A.1: Ground truth of TIMIT / TIMIT_DUMMY-01

TIMIT_RND / TIMIT_DUMMY

Dataset name	TIMIT_RND
Subset name	TIMIT_DUMMY
Speaker count	10
Total length	∼75s
Segments per speaker	multiple (every distinct segment once)
Segments seperated by silence	Yes
Remarks	Speakers are randomly picked from the whole TIMIT corpus
Diarization Difficulty	Easy to Moderate

Table A.5: TIMIT_RND / TIMIT_DUMMY Characteristics

```
1500.000 4956.000 FKDE0
6456.000 9528.000 MDJM0
11028.000 13844.000 MBML0
15344.000 18186.000 FCJF0
19686.000 22240.000 MDJM0
23740.000 26537.000 FSMA0
28037.000 31058.000 MTBC0
32558.000 37275.000 MTBC0
38775.000 40727.000 MMGK0
42227.000 45805.000 FSMA0
47305.000 49967.000 MJEE0
51467.000 55077.000 MTBC0
56577.000 59534.000 MJBG0
61034.000 63645.000 MJBG0
65145.000 69964.000 FCDR1
71464.000 75822.000 FCDR1
```

Listing A.2: Ground truth of TIMIT_RND / TIMIT_DUMMY

A.4.2 Additional Tools

Various tools were developed during this project. The most heavily used ones are open-sourced and can be found on the GitHub[6].
The open-source project space can be found using the following link: https://github.com/ZHAW-SPDR.
The most important repositories found on this project space are described below:

SPDR is the repository for the code of the system developed during this thesis.

TIMIT-stitcher is the repository for a tool which can be used to create datasets as in chapter A.4.1. These datasets can then be used in the SPDR system (placeable in the *./data/in/* folder).

ZHAW_deep_voice_dataset_converter is the repository for a tool which can be used to create datasets to train the networks in the *ZHAW_deep_voice* suite.

[6]https://www.github.com/

A.4.3 Meeting Room Setup

The RT-09 corpus consists of recordings from three different meetings. Those meetings were held on-site at the Edinburgh University (EDI), the IDIAP Research Institute (IDI), and the National Institute of Standards and Technology (NIST). The following schemes should give an idea of the meeting room looked like and how the microphones were spaced out. These schemes were provided by NIST inside the RT-09 dataset and now listed as is. Not all marked microphones are relevant for the speaker diarization task (and the MDM/SDM evaluation condition).

EDI - this is the meeting room scheme for the room at the Edinburgh University

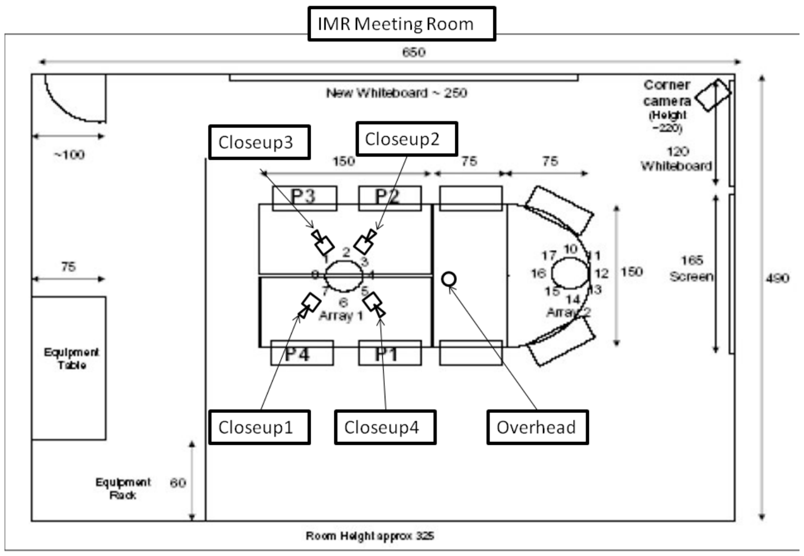


Figure A.3: EDI Meeting Room Scheme[2]

IDI - this is the meeting room scheme for the room at the IDIAP Research Institute

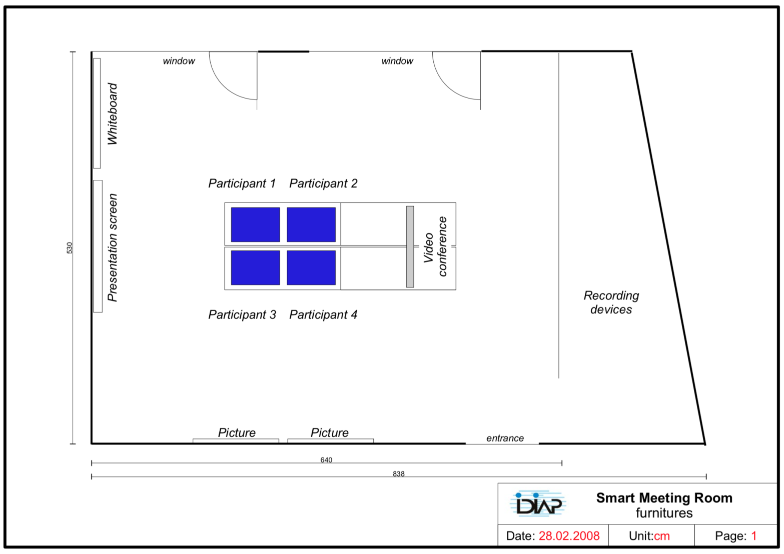


Figure A.4: IDI Meeting Room Scheme[2]

NIST - this is the meeting room scheme for the room at the National Institute of Standards and Technology.

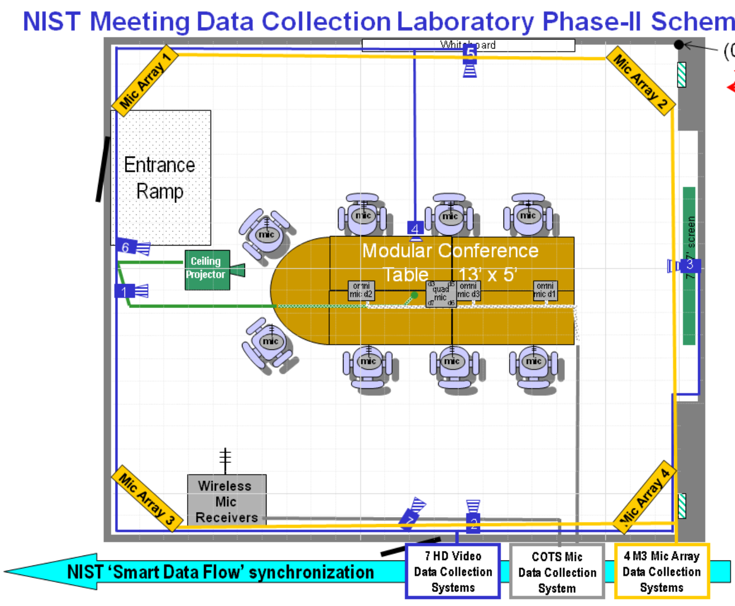

NIST Meeting Data Collection Laboratory Phase-II Schematic
Whiteboard
(0,0,0)
x
y
Mic Array 1
Mic Array 2
Mic Array 3
Mic Array 4
Entrance Ramp
Ceiling Projector
Modular Conference Table 13' x 5'
mic
omni mic d2
quad mic
omni mic d3
omni mic d1
screen
Wireless Mic Receivers
NIST 'Smart Data Flow' synchronization
7 HD Video Data Collection Systems
COTS Mic Data Collection System
4 M3 Mic Array Data Collection Systems

Figure A.5: NIST Meeting Room Scheme[2]