Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Runar Ask Johannessen

Aggregation of Speaker Embeddings for Speaker Diarization

Master's thesis in Computer Science
Supervisor: Rune Sætre, Björn Gambäck
June 2020

NTNU
Norwegian University of
Science and Technology

Runar Ask Johannessen

Aggregation of Speaker Embeddings for Speaker Diarization

Master's thesis in Computer Science
Supervisor: Rune Sætre, Björn Gambäck
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Runar Ask Johannessen

Aggregation of Speaker Embeddings for Speaker Diarization

Master's Thesis in Computer Science, Spring 2020

Data and Artificial Intelligence Group
Department of Computer Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Abstract

Speaker diarization answers the question of “who spoke when?” by splitting an audio stream into single speaker segments. This information is often used to enrich the content of automatic transcriptions. Such transcriptions generally only contain “what was said” without designating “who said what.” Speaker diarization can add this extra information by distinguishing between speakers.

Laerdal Medical is looking to apply speaker recognition technology to simulated calls for training emergency centre operatives. The operatives are trained in a simulated setting, where a trainer pretends to have an emergency and calls the emergency number. The final part of the training process is a post-analysis where the session recording and its transcript are important tools to properly evaluate the training. Laerdal Medical’s goal and the motivation for this Thesis, is to investigate the use of speaker diarization to support automatic transcription and analysis of the simulated calls.

To distinguish between speakers, a system is dependent on *speaker embeddings* that capture speaker specific characteristics. Traditionally, speaker embeddings are defined for an audio segment as the average of smaller frame embeddings. This work investigates the impact of aggregation methods, other than the average, on diarization performance. In particular, a moving average filter, dimensionality reduction, and a median operation were tested using an experimental research approach.

The results of the experiments showed that the system’s error rate decreased from 18.79% to 13.72% when using an alternative aggregation method. The results contribute to the little amount of research that has been done on the effects of different speaker embedding aggregation methods.

Sammendrag

Talerindeksering (“speaker diarization”) brukes til å svare på spørsmålet “hvem snakket når?” ved å dele lydfiler inn i segmenter med tale fra én enkelt person. Dette brukes ofte som tilleggsinformasjon til automatiske transkripsjoner. Slike transkriberingsprosesser forteller som regel bare “hva som ble sagt” uten å si noe om “hvem som sa hva.” Talerindeksering kan tilføye denne informasjonen ved å skille mellom talere.

Denne oppgaven ble motivert av Laerdal Medical som ønsker å anvende talegjenkjenning på simulerte nødsamtaler til opplæring av nødpersonell. Personellet blir trent til å takle nødsituasjoner gjennom simulerte anrop hvor en trener later som hen er vitne til en ulykke. Treningsøkter avsluttes med en evaluering hvor en skriftlig transkripsjon er et viktig verktøy. Målet til Laerdal Medical var å undersøke bruken av talerindeksering til å støtte automatisk transkribering og analyse av treningssamtalene.

For å kunne skille mellom talere er en avhengig av representasjoner som fanger personlige kjennetegn. Tradisjonelt blir taler-innbakinger (“speaker embeddings”) definert for et lydsegment som gjennomsnittet av mindre del representasjoner. Denne oppgaven utforsket effekten av forskjellige aggregeringsmetoder på talerindekseringssystemer. Spesifikt ble et glidende gjennomsnittsfilter, dimensjonsreduksjon og median testet ved eksperimentelle metoder.

Resultatene fra eksperimentene viste at feilraten gikk ned fra 18.79% til 13.72% ved å bruke de alternative aggregeringsmetodene. Disse funnene bidrar til forskningen på virkningen av forskjellige aggregeringsmetoder på taler-innbakinger.

Preface

This Master's Thesis was conducted at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway as the final part of a Master's degree in Computer Science. The task and motivation were provided by Laerdal Medical through Kaare Petersen and the work was supervised by Rune Sætre and Björn Gambäck.

I would like to thank both my supervisor Rune Sætre and assisting supervisor Björn Gambäck for their help and for keeping me on track throughout the semester. Thanks also to Torbjørn Karl Svendsen for helping me gain access to the *2000 NIST Speaker Recognition Evaluation* dataset. I also extend my gratitude to Kaare Petersen and Laerdal Medical for giving me the opportunity to take part in one of their projects and gain some insight into their work.

Finally, the experiments in this Thesis would not have been possible to perform without access to the NTNU IDUN/EPIC computing cluster (Själander et al., 2019).

Runar Ask Johannessen
Trondheim, 16th June 2020

Contents

List of Figures

List of Tables

1 Introduction

Speaker diarization seeks to answer the question of "who spoke when?" in a given audio stream. By itself, speaker diarization does not produce meaningful information. Instead, it adds value when solving other speech recognition tasks. It can be used to add metadata when creating automatic speech transcripts by labeling "who said what" in a multi-person recording, resulting in a *rich transcription*. In its most basic form, a speaker diarization system does not know the identity of the speakers but assigns them arbitrary labels during processing.

1.1 Background and Motivation

Emergency call takers have a critical role in life and death situations. To be able to handle such situations, they require proper training. Such training is often done with simulated calls between a call taker and a trainer, where the calls are recorded and analysed to be used for feedback and improvement. Laerdal Medical was looking to apply speaker diarization on recorded calls to make such analysis easier.

1.2 Goals and Research Questions

This section describes the goals set for this Thesis. Two goals were formulated and further divided into sub-goals in the form of four research questions.

Goal 1 *Investigate the performance of state-of-the-art speaker diarization systems using different speaker embedding methods.*

A key part of speaker diarization systems is how the audio is represented. In speaker diarization, the representations are often called speaker embeddings because they are made to represent (or embed) speaker specific characteristics. This goal was to look at how different methods for creating speaker embeddings would affect the performance of state-of-the-art speaker diarization systems. Achieving this goal would provide useful knowledge for working on Goal 2.

Goal 2 *Determine whether an unsupervised or a fully supervised speaker diarization system performs best on simulated emergency calls.*

The motivation was to apply speaker diarization on simulated emergency calls for training call centre personnel. Using the knowledge about speaker embeddings learned by completing Goal 1, the second goal was to compare two state-of-the-art systems on a set of simulated emergency calls provided by Laerdal Medical.

Research question 1a *How are state-of-the-art Speaker Diarization systems built?*

In order to achieve Goal 1, state-of-the-art speaker diarization systems had to be implemented. Research question 1a needs to be answered as part of this process.

Research question 1b *What features do state-of-the-art Speaker Diarization systems use?*

This research question relates to Goal 1. Knowledge of what embeddings are used in other systems is required to investigate the impact of other embedding methods.

Research question 2 *How do different embedding aggregation methods impact system performance?*

Answering this question directly achieves Goal 1.

Research question 3 *How does a supervised diarization system trained on off-domain data perform compared to an unsupervised method on simulated emergency calls?*

Traditional speaker diarization systems were unsupervised and did not require any training to produce diarization results. Later systems started to experiment with supervised learning techniques to produce better diarization results. Both types of systems, unsupervised and supervised, were able to produce state-of-the-art diarization results. With their strengths and weaknesses, the two system types should be compared on the simulated emergency calls. The supervised system was to be trained on off-domain data because the number of simulated emergency calls provided by Laerdal Medical was not expected to be sufficient to train a complex supervised machine learning algorithm.

1.3 Research Method

The work was done in the form of both exploratory (or secondary) and experimental research. The first part covered existing literature and datasets to acquire an understanding of the field. The purpose of this was to create a basis that would allow the design of relevant experiments for the final part. The experimental part of the work consisted of implementing and testing the performance of a speaker diarization system on a standard dataset.

1.4 Contributions

The work contributes to the speaker diarization field by adding to the research on speaker embeddings used in speaker diarization systems. In particular, using a new aggregation method for combining frame-level speaker embeddings were found to significantly improve overall diarization performance. With the new method, proposed by Dimitriadis (2019), the system achieved an error rate of 13.72%. This was an absolute improvement of 5.07% compared to when the standard aggregation method was used. The results could be used for benchmarks and ideas for later research on speaker embeddings in speaker diarization systems. Other fields where separable speaker embeddings are desired could also benefit from the results.

1.5 Thesis Structure

The rest of this Thesis consists of seven chapters. Chapter 2 provides the theoretical background necessary for the uninitiated reader to understand the subsequent chapters. It also introduces terminology, such that important terms can be used precisely in later chapters. Chapter 4 describes the different datasets that were used. Its purpose is to give an understanding of what specific data was processed, as well as making the work more reproducible. In chapter 3, previous relevant work is presented. The chapter is meant to give an understanding of the speaker diarization field as a whole. Chapter 5 covers the system architecture that was implemented and used for the experiments. Chapter 6 describes the planning, setup, and results of the experiments that were performed to answer the research questions. In chapter 7, the results of the experiments are evaluated and discussed. Finally, chapter 8 evaluates whether the research questions were answered and discusses future work.

2 Background Theory

The goal of this chapter is to introduce the theoretical elements and terminology required to sufficiently understand the contents of the following chapters. Section 2.1 introduces some basic speaker diarization concepts and the principal components of a speaker diarization system. Section 2.2 covers some relevant machine learning methods. Most notably, a specific class of neural networks known as *recurrent neural networks* is introduced together with two variations, *long short-term memory*, and the *gated recurrent unit*. Lastly, section 2.3 covers basic concepts related to signal representations used in speaker diarization systems.

2.1 Speaker Diarization

Speaker diarization seeks to answer the question of "who spoke when?" in a given audio stream by partitioning it into speaker homogeneous segments and grouping them based on speaker identities. That is, segments that contain speech from exactly one speaker. Figure 2.1 shows a general speaker diarization architecture. The exact implementation and interaction between the modules vary between systems, but the figure captures the core of the diarization process. The next part of this section gives a detailed explanation of the four main diarization components illustrated in figure 2.1. Following is a brief discussion on overlapping speech and its position in the speaker diarization field. Finally, two common error metrics used to evaluate diarization systems are introduced.

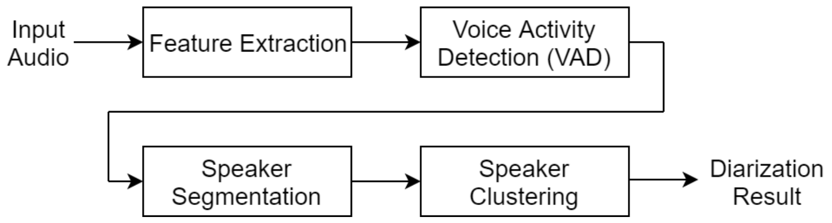


Figure 2.1: General speaker diarization architecture. Recreated from Basu et al. (2016) with permission.

2.1.1 Feature Extraction

The first step in many speaker diarization systems is to extract a set of features to be used by the subsequent processes. The input to diarization systems is audio files containing conversational speech between at least two unique speakers. Conversational speech contains a lot of information. The purpose of feature extraction is to produce a set of features that represents the conversational data without any irrelevant information for diarization purposes. Relevant information, in this case, is anything that allows a system to distinguish between speakers. That is, speech from two different speakers should result in feature values that are more different than speech from only one speaker. A specific set of feature values is often referred to as a **speaker embedding**.

A traditional approach to feature extraction is based on capturing frequency information using mel-frequency cepstral coefficients (see section 2.3.5). A more modern option is to use abstract features, in the sense that they do not directly correspond to any intrinsic property of the audio (see section 2.3.6).

2.1.2 Voice Activity Detection

Non-speech such as silence, noise, and music is not relevant for speaker diarization and is usually removed from the audio using a *voice activity detector* (VAD), also called *speech activity detector* (SAD). A VAD is essentially a binary classifier that determines whether audio contains human speech or not. A standard way to implement a VAD is by measuring the energy levels of the audio and classifying it as speech if the values are above a certain threshold. The problem with this approach is that it does not account for all types of non-speech. Music is an example of non-speech that would be expected to have similar energy levels as speech, but should not be classified as such. An alternative that is better suited for separating speech from music is model-based VADs. As the name suggests, a model-based VAD creates models that are used to represent the two classes, speech and non-speech. Classification is performed by evaluating which of the two models best describes the audio. For speaker diarization, both types of VADs are valid options and the decision mostly comes down to a trade-off between simplicity and accuracy.

2.1.3 Segmentation

In the context of speaker diarization, segmentation is the process of partitioning a given audio signal into speaker homogeneous segments. Single-speaker segments are usually determined by locating speaker changes and dividing the audio at these points. If all points of speaker change are found the resulting segments contain exactly one speaker. The segmentation task, therefore, comes down to successfully detecting every time the speaker switches. Speaker changes can be found by running two adjacent, possibly overlapping, sliding windows over the audio. For each step, the windows are compared to see if they contain speech from different speakers. If they do, the boundary between the windows is marked as a speaker change.

2.1.4 Clustering

Clustering is the grouping of data into "clusters" where similar data points are put together. In speaker diarization, clustering is used to group the single-speaker segments from the segmentation phase into speaker specific clusters. Ideally, this phase produces a cluster for each unique speaker and each cluster contains speech from exactly one speaker. There are many different approaches to clustering and this section introduces two of the more popular algorithms for speaker diarization, agglomerative hierarchical clustering, and spectral clustering.

Agglomerative hierarchical clustering (AHC) is a "bottom-up" clustering algorithm. The algorithm is initialized with many small clusters that are merged iteratively until some stopping criterion is met. In general, the two most similar clusters are merged until there is a clear difference between all clusters or until a specified number of clusters are left. Spectral clustering is not a specific clustering algorithm, but a technique that can be used in combination with other clustering methods. In short, it reduces the dimensionality of the input data using the eigenvectors and eigenvalues of the similarity matrix representing the relationship between the data points. After the dimensionality reduction, the reduced feature vectors are clustered using another clustering algorithm such as AHC or K-means. A requirement for both AHC and spectral clustering to work is that all data samples need to be available before starting. AHC needs the information to know which clusters to merge and spectral approaches need it for creating the similarity matrix. Algorithms that require all data points to start are said to work in an *offline* fashion. This is in contrast to *online* algorithms that can handle data one at a time. For speaker diarization, an online approach would be able to produce diarization results of a live conversation. An offline algorithm would have to wait for the conversation to be finished before processing.

2.1.5 Chinese Restaurant Process

The experiments were based on a supervised diarization system developed by Zhang et al. (2019), the *unbounded interleaved-state recurrent neural network* (UIS-RNN, see section 3.2). In the UIS-RNN, clustering was based on the *distance dependent Chinese restaurant process*, a variation of the *Chinese restaurant process*. This section covers these two processes, starting with the basic Chinese Restaurant Process followed by the distance dependent variation.

The Chinese restaurant process (CRP) is a stochastic process used to cluster a collection of observations. The process has its name after the metaphor used to explain it, where the observations are analogous to customers to be seated at an infinitely large Chinese restaurant. In the metaphor, clusters are represented by tables and cluster members as customers sitting at the tables. Customers choose tables one at a time and either sit at an already occupied table or a new table. Naturally, the first customer can only go to an empty table. The driving assumption behind the table selection is that customers prefer to sit at popular tables, but always have a non-zero probability of choosing to sit at an empty table. The probability of customer number N+1 sitting at table k is defined as:

$$P(z_{N+1} = k \mid n, \alpha) = \frac{n_k}{N + \alpha} \tag{2.1}$$

where n_k is the number of customers already at table k, N is the total number of customers seated and α is a *dispersion parameter.* A separate equation defines the probability of customer N+1 choosing an empty table:

$$P(z_{N+1} = K + 1 \mid n, \alpha) = \frac{\alpha}{N + \alpha} \tag{2.2}$$

where K is the number of tables already having at least one customer. As shown by this equation, α is directly proportional to the probability of the customer sitting at a new table. In other words, it is proportional to instantiating a new cluster. With these definitions, the CRP as able to model an unspecified number of clusters. It does not require the number of clusters (or groups) to be specified before the clustering, as opposed to other popular clustering methods such as K-means.

The standard CRP is exchangeable, that is, the order in which customers are seated does not affect the probabilities of the clustering results. This property makes the CRP unsuited for time series data or other observations where the order matters. The *distance dependent Chinese restaurant process* (ddCRP) is an alternative version of the CRP that can handle ordered data. The ddCRP utilizes a distance metric to bias the clustering such that similar data points are more likely to be in the same cluster. Using the Chinese restaurant metaphor, the ddCRP changes the view from customers sitting at tables, to customers sitting with other customers. That is, new customers are seated with other customers rather than specific tables. The probabilities are therefore shifted from a customer sitting at table X to the customer sitting with customer A or B. These new probabilities are dependent on the distance between the customers according to some metric, for example, time (customers arriving at similar times are more likely to sit together) or age (customers of similar age are more likely to sit together). The modified probability function is defined as:

$$p(c_i = j \mid D, f, \alpha) \propto \begin{cases} f(d_{ij}), & \text{if } i \neq j \\ \alpha, & \text{if } i = j \end{cases} \tag{2.3}$$

where d_{ij} is the distance between a new customer i and an already seated customer j. D represents the set of all distances between customers and f is a decay function that translates the distance between two customers to a probability of them sitting together. For example, if similar customers should sit together and d_{ij} is large, then $f(d_{ij})$ should be small. $c_i = j$ denotes that customer i sits with j. α is the same as before, the probability of the new customer i creating a new group (sitting at an empty table).

2.1.6 Overlapping Speech

Overlap occurs when two or more people talk at the same time. In speaker diarization, both detecting and performing inference on overlapped speech is a tough challenge. At the time of writing, most of the literature that does not explicitly focus on overlapped speech excludes it from evaluation entirely. A reason for this is that many systems assume, directly or indirectly, that no overlapped speech is present in the data. These systems are therefore incapable of handling overlapped speech segments.

For the experiments in chapter 6, overlap was not considered following the work of Zhang et al. (2019) where they assumed single speaker segments.

2.1.7 Diarization Error Rate

The main performance metric used to evaluate speaker diarization systems is the *diarization error rate* (DER). The DER is calculated by comparing the ground truth labels of a segment with the predicted labels produced using the following equation:

$$DER = \frac{\text{false alarm} + \text{missed detection} + \text{confusion}}{\text{total}} \tag{2.4}$$

False alarm is the sum of all the time that the system's hypothesis predicts speech, while there is no speech at all. A *missed detection* is the opposite of a false alarm, where the system incorrectly predicts that there is no speech. *Confusion* is made up of all points where the system predicts the incorrect speaker. Finally, the *total* is the total duration of the ground truth (reference) segments. Figure 2.2 illustrates the three error types.

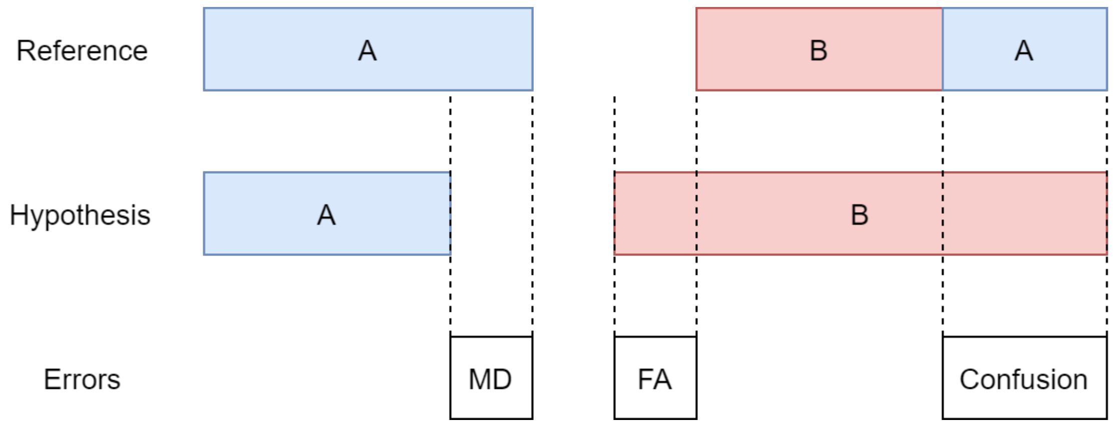


Figure 2.2: Examples of False Alarm (FA), Missed Detection (MD) and Confusion errors.

In practice, it is common to allow for errors less than 250ms between segments and also not to consider overlapped speech during evaluation. False alarms and missed detections are usually excluded from reported DERs because they are dependent on the specific VAD implementation and most papers primarily focus on the clustering module. These conventions were adopted when reporting the results of the experiments in section 6.3.

2.1.8 Equal Error Rate

Equal error rate (EER) is an error metric often used for speaker verification systems. It is defined as the error when the *false acceptance rate* (FAR) equals the *false rejection rate* (FRR). In speaker verification, the goal is to determine whether two speech samples belong to the same speaker or not. The FAR is then the percentage of times where the samples are incorrectly classified as originating from the same speaker and the FER is the percentage of times where the samples are incorrectly classified as different speakers.

The EER was used to evaluate the speaker encoder in experiment 1, described in section 6.3.1. The EER was selected as an appropriate performance measure because the encoder was built and trained as a speaker verification system.

2.2 Machine Learning Methods

Machine learning is a field about making computers able to handle tasks without explicitly programming how they should solve them. While there are multiple types of machine learning algorithms, in general, most systems are *trained* using example input data. One type of algorithm called **supervised learning**, accompanies each example input with a corresponding desired output. That is, the system is provided multiple input-output data pairs in order for it to *learn* how to map a given input to the desired output. **Unsupervised learning** algorithms differ from supervised approaches in that they are not provided with sample outputs, only inputs. A class of unsupervised learning algorithms important to speaker diarization is cluster analysis. This was described previously in section 2.1.4, where the system groups similar data points together.

The rest of this section is dedicated to machine learning concepts relevant to this Thesis. In particular, one of the most popular supervised machine learning algorithms, *artificial neural networks* is presented. Further, introductions are provided for a specific class of artificial neural networks, *recurrent neural networks* (RNNs), and two specific types of RNNs, *long-short term memory*, and *gated recurrent unit.* The chapter is finished with a section about *principal component analysis.*

2.2.1 Artificial Neural Networks

An artificial neural network (ANN) is a system inspired by the design of biological brains. The basic component of an ANN is the *artificial neuron*, a simple computational unit inspired by biological neurons. An artificial neuron, illustrated in figure 2.3, takes a set of inputs X, applies a function F, and returns the result as its output. When discussing neural networks it is common to refer to artificial neurons as either nodes or simply neurons, with the context implying that they are artificial. Neural networks are built up of many neurons, often in multiple layers. Figure 2.4 is an example of a *feedforward* network where every neuron is connected to all neurons in its adjacent layers. The network is *fully connected.* In general, the first layer (leftmost in the figure) is the input layer of the network and the last layer is the output layer. All layers between the input and output layers are referred to as hidden layers. The network in the figure therefore has a single hidden layer. However, it is common for ANNs to have multiple hidden layers in which they often are called *deep neural networks* (DNNs).

Each edge between a pair of nodes is assigned a weight, representing the strength of the connection between the two nodes. Training ANNs consists of updating these weights such that the network produces the desired outputs. In the case of supervised learning, the training data is a set of example inputs with corresponding outputs. For each input sample, the network's actual output is compared with the desired output measuring their

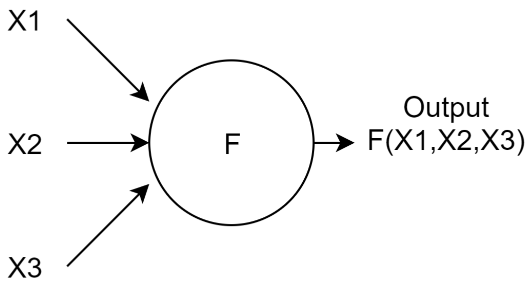


Figure 2.3: An artificial neuron with three inputs. Figure reused from Johannessen (2019), a previous project by the author.

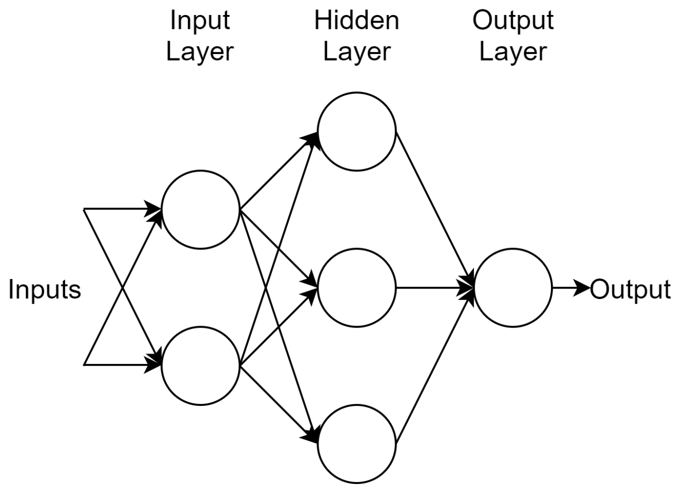


Figure 2.4: Neural network with one hidden layer. Figure reused from Johannessen (2019), a previous project by the author.

difference using some loss function. The network weights are then updated in a way that reduces the loss, that is, the difference between the actual output and the desired output is reduced.

2.2.2 Recurrent Neural Network

As the name suggests, feedforward ANNs only send information in one direction, forward from the input layer towards the output. *Recurrent neural networks* (RNNs) is a class of neural networks that include feedback loops, allowing the network to "remember" previous inputs. This property is especially useful for sequences of inputs where the order matters, such as time series or speech data. RNNs keep a hidden state that represents the context created by all seen inputs and updates the state each time a new input is provided. Figure 2.5 illustrates a simple RNN where the network's output is "looped" back to be processed together with the next input.

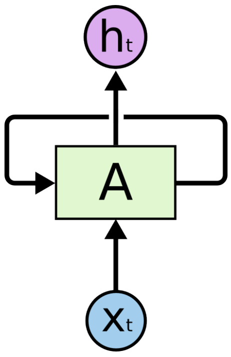


Figure 2.5: Basic RNN. Illustration from Olah (2015), used with permission.

Due to this temporal aspect, RNNs can be visualized by "rolling them out" over time as shown in figure 2.6. The figure illustrates that previous outputs impact future outputs. A weakness of the RNN architecture is that it struggles to remember information over longer time intervals. This becomes apparent when observing the way in which information is passed forward in time. The hidden state from the previous time step is processed together with the current output which means that over time, early hidden states can be "watered down" to the point where they are practically forgotten. Naturally, this is a bigger problem for longer input sequences than shorter ones. Due to this property of the RNN, it is common to say that it suffers from short-time memory. The RNN architecture famously also suffers from the *vanishing gradient problem* and can, therefore, be troublesome to train correctly. The two following subsections introduce variations of the standard RNN that, for the most part, solves the problem of vanishing gradients, namely the *long short-term memory* and the *gated recurrent unit*. Additionally, these two architectures do not have the same short-term memory problem as the standard RNN.

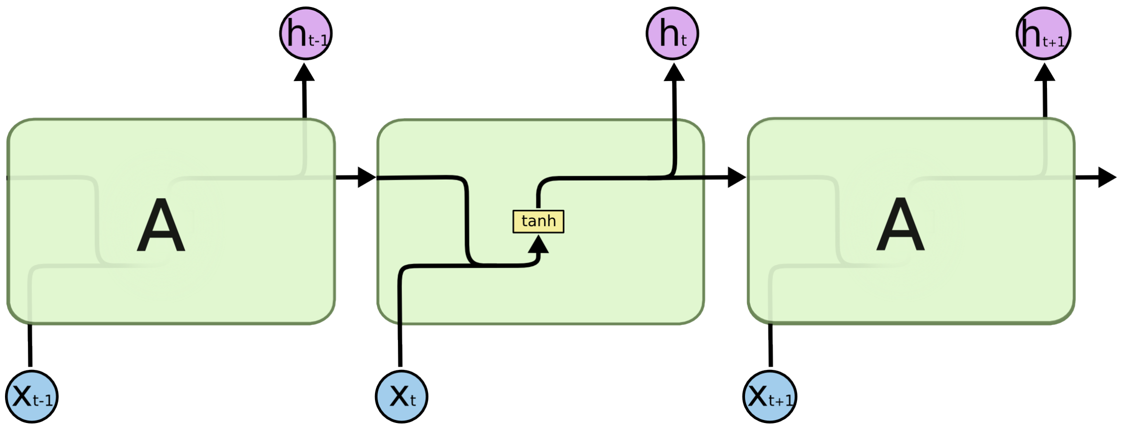


Figure 2.6: RNN inputs and outputs over multiple timesteps. Illustration from Olah (2015), used with permission.

2.2.3 Long Short-Term Memory

Long short-term memory (LSTM) enhances the standard RNN architecture with “gates” that regulate how data flows through the network. Specifically, an LSTM unit keeps track of both a cell state and a hidden state. The cell state keeps the relevant information based on the current input, previous cell state, and previous hidden state, whereas the hidden state keeps information about previous inputs. Whether something is relevant or not is determined by three gates: *forget*, *input*, and *output*. As the name suggests, the *forget* gate determines what should be forgotten and what to keep from the previous cell state. The *input* gate regulates what information to use from the new input and previous hidden state. Finally, the *output* gate creates a new hidden state based on the new cell state.

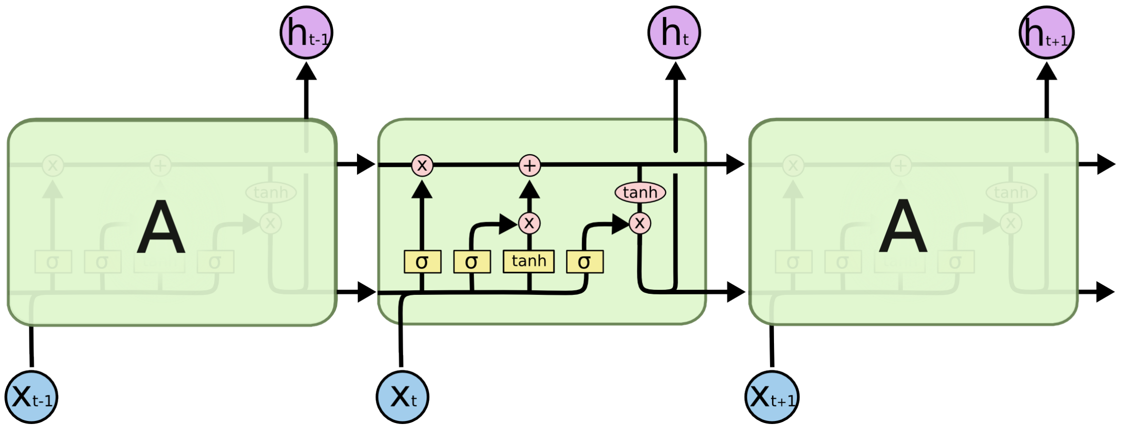


Figure 2.7: LSTM diagram. Illustration from Olah (2015), used with permission.

Figure 2.7 illustrates an LSTM similarly to how figure 2.6 depicted the standard RNN. The general way in which data is processed is the same, but the LSTM includes additional operations to control the network’s memory. In the figure, each gate is instantiated as a sigmoid function which is the most commonly used activation for LSTM gates. This means that when the previous hidden state h_{t-1} and input x_t is sent through the forget gate, it results in a vector with values between 0 and 1. This vector is then multiplied with the previous cell state c_{t-1}, forgetting (or giving less weight to) the values in the cell state corresponding to lower values in the forget vector. In other words, the gate decides which information from the previous cell state to keep. A similar computation is performed by the input gate, except that the gate output is used to decide which information from the input and previous hidden state to use. The new cell state c_t can then be computed by adding together the outputs of the forget and input gates. Finally, the output gate creates a new hidden state h_t from c_t using h_{t-1} and x_t to determine what information is relevant for future inputs. The “tanh” nodes in the figure are there to make sure that the processed values do not become dominantly large or small.

To summarize, LSTMs can be used to avoid the problem of short-term memory that standard RNNs have. The key difference is that LSTMs use “gates” to control what to keep in memory and what to forget.

2.2.4 Gated Recurrent Unit

A *gated recurrent unit* (GRU) can be thought of as an LSTM without an output gate and only a single state, instead of a cell and a hidden state. Illustrated in figure 2.8, a GRU has a *reset* gate and an *update* gate. The update gate controls what to discard and what new information to incorporate, similar to the combined efforts of the forget and input gates of an LSTM. The reset gate also regulates what previous knowledge to discard.

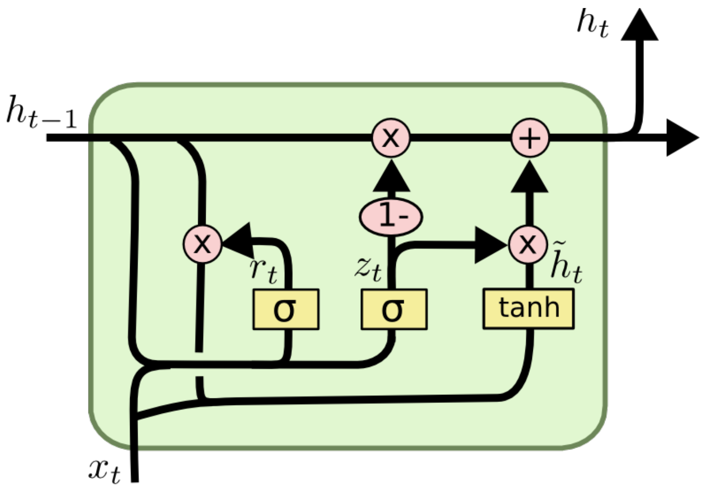


Figure 2.8: GRU diagram. Illustration from Olah (2015), used with permission.

Having one less gate than the LSTM, the GRU has fewer operations to perform and parameters to tune, making it a bit faster to train. When it comes to deciding between implementing an LSTM or a GRU there is generally no clear answer to which is better. The best performer of the two may vary between use cases and any performance differences need to be found through experiments. Independent of which architecture is chosen, the main point is that both LSTMs and GRUs can be used in place of RNNs to handle data where the RNNs short-term memory is problematic.

2.2.5 Principal Component Analysis

Principal component analysis (PCA) is a tool for reducing the dimensionality of a dataset. That is, PCA is used to reduce the number of features. At the same time, PCA attempts to retain as much of the information in the data as possible. In general, it is not possible to keep all of the information, but the loss is compensated for by (hopefully) easier analysis.

PCA is based on finding the *principal components* (PCs) of a feature space, where a PC is a linear combination of all the original features. The components are sorted such that the first PC can be used to explain most of the variance in the data, the second PC the second most variance, etc. Principal components are found as linear combinations of the original features where the combination is weighted using an eigenvector of the covariance matrix. The number of a principal component is determined by the eigenvalue corresponding to the eigenvector used for the projection. Specifically, the first principal component is found by using the eigenvector with the highest eigenvalue as a weight vector. The second principal component of a feature vector is subsequently found by using the eigenvector with the second-highest eigenvalue. That is, the eigenvalues directly correspond to the amount of information in the principal component found by using its corresponding eigenvector. In general, the eigenvectors and eigenvalues of the covariance matrix are calculated and create a projection matrix whose columns are the calculated eigenvectors. The columns are sorted such that the first column corresponds to the highest eigenvalue, the second column to the second-highest eigenvalue, etc. The matrix can then be used to transform a feature vector to its principal components. The dimension of the principal component vector can be reduced as desired by removing columns from the projection matrix, starting with the columns corresponding to the lowest eigenvalues.

In summary, PCA is a technique for dimensionality reduction that discards the features with the least information first. It should be noted that while PCA reduces the number of features, the transformed feature values generally become harder to interpret. That is, the resulting principal component variables no longer correspond to any measurable feature because they are linear combinations of all original features. This generally makes the features harder to interpret, but easier to analyse. PCA is therefore a tool that can reduce the dimensionality of data while keeping as much information as possible, but at the cost of interpretability.

2.3 Speech Representations

For a long time, cepstral features such as *mel-frequency cepstral coefficients* (MFCCs) were the primary signal representation used in speaker diarization systems. These coefficients try to capture the shape of a signal's power spectrum, which for speech is connected to the shape of the speaker's vocal tract. This means that if the MFCCs do their job properly, they should represent characteristics unique to a given speaker. The process of extracting MFCCs requires some basic understanding of the Fourier transform, spectrograms, and the mel scale which is covered in sections 2.3.1, 2.3.2 and 2.3.3 respectively. Section 2.3.4 presents the log-mel spectrogram, combining spectrograms and the mel scale before MFCCs are explained in section 2.3.5. The section is finished with an introduction to bottleneck features (BNFs) in section 2.3.6. While MFCCs represent physical aspects of speech signals directly, BNFs are more abstract representations. Often taking MFCCs or similar features as inputs, BNFs utilise neural networks to extract embeddings. The hope is that the neural networks transform the input MFCCs in a way that accentuates speaker characteristics and removes features that are less useful for distinguishing speakers.

2.3.1 Fourier Transform

The Fourier transform is a method that can be used to break a signal down to its constituent frequencies. For speech processing applications this information can be used to differentiate between speakers. A basic assumption is that, even if two people record the same utterance with the same intonation, the frequencies that make up the resulting signal will vary due to differences in their physical vocal tracts. The Fourier transform allows us to analyse signals in terms of what frequencies are present and the magnitudes of these frequencies.

The standard Fourier transform is defined for continuous signals, but in speaker diarization discrete, finite signals are used. The *discrete Fourier transform* (DFT) is a version of the Fourier transform that handles exactly such signals.

2.3.2 Spectrogram

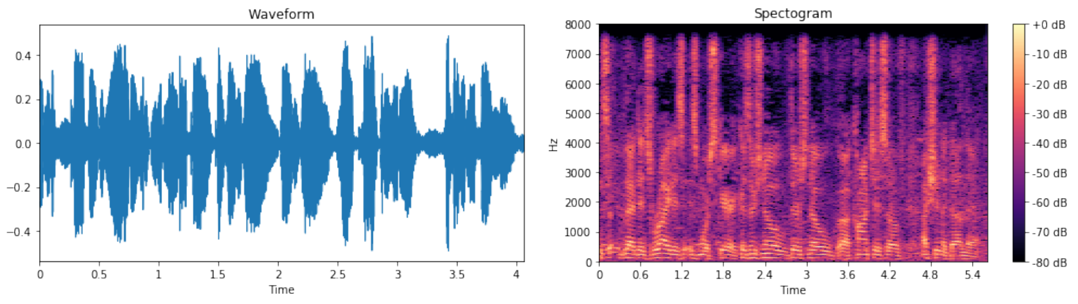
A spectrogram is a visual representation of a signal that conveys information about its frequencies over time. This differs from the standard waveform which displays the signal's amplitude over time. Figure 2.9 shows the waveform and spectrogram for the same short speech signal.[1] The waveform, the signal's amplitude over time, shows when in the audio segment there is activity and when there is silence. However, for speech recognition tasks such as speaker diarization, the amplitude alone is not always enough to be able to distinguish between different speakers. The spectrogram visualizes information about the frequencies in the signal, usually by taking the *short-time Fourier transform* (STFT). That is, divide the signal into smaller windows and take the discrete Fourier transform (DFT) of each window. This results in a series of frequency spectra that, when plotted in temporal order, describes how the frequency content of the signal changes over time. The resulting plot is the spectrogram of the signal.

An inconvenience with the DFT is that it outputs complex values and the resulting spectrum is therefore not easily visualized. A common solution is to transform the initial DFT spectra X_k to *log-spectra* using the formula $20 \log_{10} |X_k|$. If this is done for each window in an STFT the result is a log-spectrogram. The spectrogram in figure 2.9b is, in fact, the log-spectrogram of the speech signal.

2.3.3 Mel Scale

When presented with two sounds at different frequencies, humans are better at identifying small changes at lower frequencies. That is, the perceived difference between the two frequencies is greater if the frequencies are lower. A toy example is that of comparing a sound at 300Hz to one at 400Hz, and then comparing two other sounds at 900Hz and 1000Hz respectively. Even though the difference within each pair is the same (100Hz), a human would perceive a greater difference between the 300-400Hz pair. This is what the *mel scale* is for. It is a non-linear transformation that makes sounds that are perceived as

[1]The waveform and spectrograms in this chapter were extracted from the "id10001/1zcIwhmdeo4/00003.wav" file in the VoxCeleb1 dataset.



(a) Waveform of an audio signal.

(b) Spectrogram of an audio signal.

Figure 2.9: Waveform and Spectrogram.

equal in distance, also measure as such. Using the mel scale, the difference between the example pairs is 108 and 68 respectively. In conclusion, the mel scale puts more emphasis on the lower frequency sounds and less weight on higher frequency sounds. In practice, this puts less weight on the frequencies that are less noticeable by people.

2.3.4 Log-Mel Spectrogram

Scaling the frequency component of a spectrogram to the mel scale results in a *mel spectrogram*. If the scaled spectrogram was a log-spectrogram it becomes a *log-mel spectrogram*. Figure 2.10 shows the log-mel spectrogram of the same audio signal used for the illustrations in figure 2.9. Similar to the perception of frequencies, humans are also better at identifying small changes at lower magnitudes. The logarithmic scaling done in section 2.3.2 not only allows easier visualisation of the spectrogram. It also scales the signal amplitudes similar to what the mel scale does for the frequencies. That is, more weight is put on low magnitude audio.

In summary, a log-mel spectrogram contains information about a signal's frequencies over time (like any other spectrogram), but the magnitude and frequency axes are transformed to account for human sensitivity.

2.3.5 Mel-frequency Cepstral Coefficients

Mel-frequency cepstral coefficients (MFCCs) are a signal representation commonly used in speaker diarization based on the signal's log-mel spectra. For each spectrum, a set of coefficients is obtained that represents its overall structure. The process consists of taking either the Discrete Fourier transform (DFT) or the Discrete Cosine transform (DCT) of each spectrum producing their corresponding *cepstrum*. The cepstrum contains information about the periodic structure of the corresponding spectrum.

Mel-frequency cepstral coefficients are the coefficients obtained when applying either the DFT or DCT to a log-mel spectrum. Most speaker diarization implementations using MFCCs use the DFT both to get the spectrum and cepstrum when extracting the coefficients.

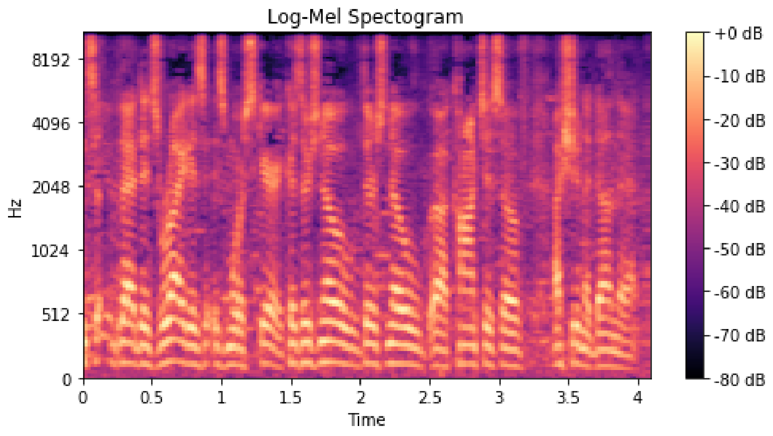


Figure 2.10: Log-mel spectrogram of an audio signal.

2.3.6 Bottleneck Features

Bottleneck features (BNFs) are activations from a hidden layer in a neural network. Figure 2.11 illustrates the concept of BNFs being extracted from the last hidden layer of a neural network. In practice, the last hidden layer is a fully connected projection layer with a lower dimensionality than the previous layers. Thus the name "bottleneck" features. For generating speaker embeddings, a standard technique is to train the network for speaker verification[2] and extract BNFs from the last hidden layer during inference. The hope is that the network, having learnt to verify speaker identities, has developed an internal, compact representation that can be utilized for other speaker recognition tasks.

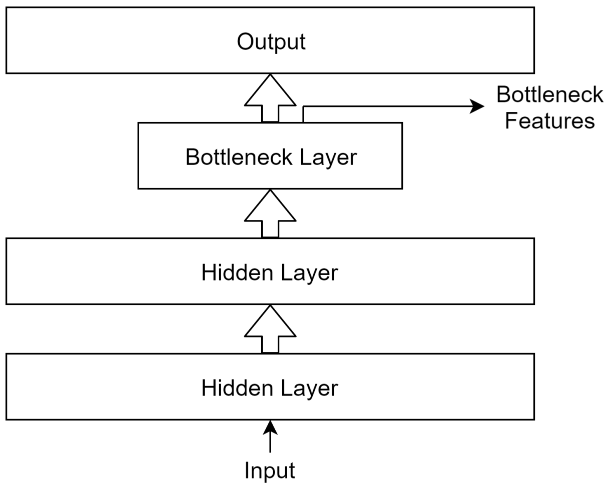


Figure 2.11: A simple neural network with a bottleneck layer.

[2]Given two audio segments, determine whether they originate from the same speaker.

At the time of writing, most state-of-the-art speaker diarization systems implement some type of BNFs in the form of **d-vectors**. The term d-vector was coined by Variani et al. (2014) and defined as the aggregation of frame-level bottleneck features for a given segment. The process is illustrated in figure 2.12. First, the segment is considered as, usually overlapping, frames that are fed to a neural network encoder. BNFs are then extracted for each frame. The d-vector representing the segment is defined as the aggregation of the frame-level BNFs. Most d-vector implementations in the literature take the average of the frame-level BNFs as proposed in the original d-vector paper. Although the original d-vector definition by Variani et al. (2014) used a Deep Neural Network (DNN) and took the average of frame-level BNFs, the term is used for other similar representations as well. In general, d-vector is often used for any speaker embedding that is based on bottleneck features from a neural network. The exact specifications such as if the network is a DNN, a recurrent neural network (RNN) or long short-term memory (LSTM) network does not matter. In most papers, both the frame-level embeddings in figure 2.12 and the aggregated embedding would be referred to as d-vectors. In such cases, the two can be separated by specifying whether they are frame-level or segment d-vectors.

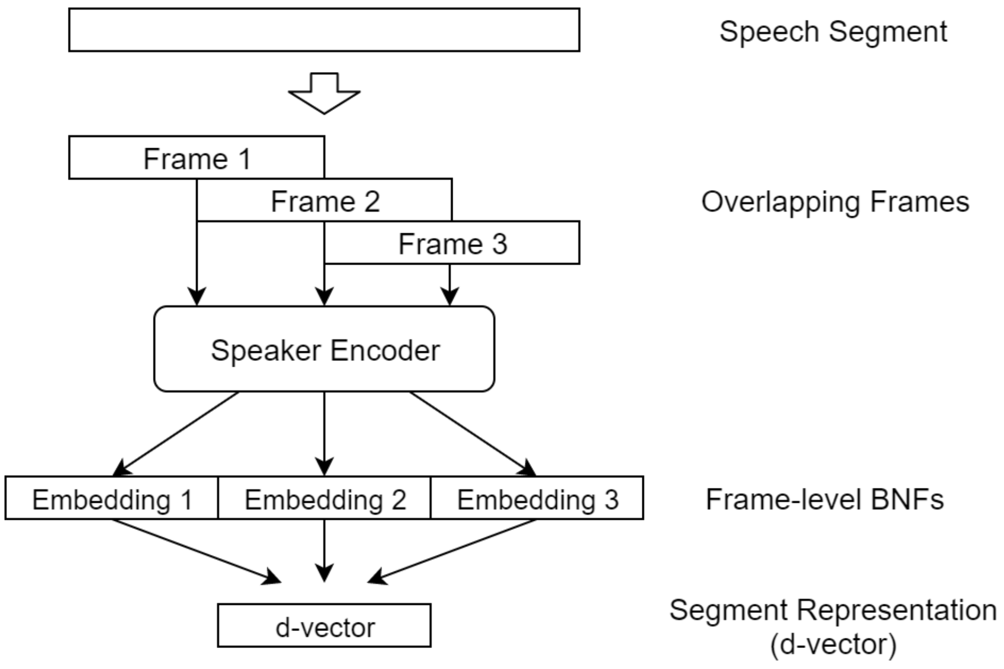


Figure 2.12: d-vector pipeline.

3 Related Work

This chapter presents works on speaker diarization to give the reader a basic introduction to the field. The chapter is divided into two sections based on the most common focal points in the speaker diarization literature, feature extraction (or speaker embeddings) and clustering. The first section covers three of the most important representations that are used, *mel frequency cepstral coefficients*, *i-vectors* and *d-vectors*. Two subsections, 3.1.1 and 3.1.2, are included where the theory directly related to the work is explained in-depth. The second section of this chapter covers clustering methods. *Agglomerative hierarchical clustering* and *spectral clustering*, two of the most used clustering algorithms in the field, and some later approaches based on machine learning. Finally, a specific machine learning model, the *unbounded interleaved-state recurrent neural network*, is described in-depth.

3.1 Speaker Embeddings For Speaker Diarization

A fundamental representation used in speaker diarization is *mel frequency cepstral coefficients* (MFCCs) described in section 2.3.5. Traditionally, audio segments were represented by their MFCCs and the coefficients were used directly by the clustering algorithm. With the introduction of abstract features, such as i-vectors (introduced below) and d-vectors (section 2.3.6), it became more common to use cepstral representations as input to feature extractors, rather than as features themselves. Acquiring the MFCCs of a signal is straightforward and leaves little room for implementation differences between systems. That is, the MFCCs of a signal is mostly independent of implementation details. This means that any difference in performance between two systems using MFCCs as features most likely is caused by their choice of segmentation and clustering algorithms. In practice, this is only partly true, because systems do not necessarily use the same amount of coefficients. For example, Tritschler and Gopinath (1999) used 24 MFCCs as their features while Ning et al. (2006) only used 20. The number of MFCCs used can also vary between systems using more abstract representations. An example is Yella et al. (2014) and Wan et al. (2018). Both systems extracted abstract d-vector representations based on MFCCs, but the former used 19 coefficients while the latter used 40. The reasoning behind using only some of the cepstral coefficients is the fact that the first coefficients contain most of the important information (Zhen et al., 2000). Generally, sufficient representations of the signals' spectra can be acquired without using all possible MFCCs and at some point, adding additional coefficients contribute little to the performance of the system. Traditionally, speaker diarization systems used around 20 coefficients, but later works trend towards using around 40. By using the mel scale, MFCCs emphasize on the parts of speech that humans use to differentiate between speakers, but there is no explicit effort to filter out information not related to the speaker. That is, MFCCs can contain some variability related to the actual words that are spoken, but an ideal representation would only contain speaker specific characteristics. Abstract representations such as i-vectors and d-vectors were developed as a way to extract and condense only the information relevant for distinguishing between speakers.

Identity-vectors usually referred to as *i-vectors*, were the result of research towards more speaker discriminative features using factor analysis. The term was coined by Dehak et al. (2010), but the concept had been presented earlier in Dehak et al. (2009). The method followed current trends in using factor analysis to define a new speaker space from which features were extracted by projecting speech utterances onto the new space. At the time, speaker factors were acquired from a super vector from two separate subspaces, a speaker and a channel space. Dehak et al. (2009) proposed to define a single space, encompassing both the speaker and channel variability called the *total variability space*. The representation acquired by projecting a speech utterance onto the total variability space is what they defined as an i-vector. After their introduction, i-vectors quickly became a standard within the field and was not challenged until the introduction of neural network embeddings.

Inspired by the success of neural networks in speaker recognition and verification, Yella et al. (2014) proposed a speaker diarization system using a neural network as a feature extractor (or speaker encoder). In particular, Yella et al. (2014) extracted bottleneck features as explained in section 2.3.6. To summarize, a set of bottleneck features (BNFs) is the activation of a hidden layer in a neural network. The system is often implemented as a neural network trained to classify whether two given utterances originate from the same speaker or not. The hope is that the network's internal representation of speech utterances contain information about speaker specific characteristics. Around the same time that BNFs started to see use in diarization systems, Variani et al. (2014) proposed a new speaker embedding scheme building on BNFs for a speaker verification system. Their scheme was practically equivalent to that of Yella et al. (2014), with an additional step. For each utterance, instead of extracting BNFs directly, they split it into smaller frames. Each frame was then encoded using the neural network and the final embedding was the average of these frame-level representations. The final embeddings were called d-vectors and similar schemes are still used in state-of-the-art systems at the time of writing. Similar to the development of d-vectors was the work by Snyder et al. (2016) and Garcia-Romero et al. (2017) which culminated in an embedding scheme they called x-vectors (Snyder et al., 2018). While the original implementation details of d-vectors and x-vectors had some differences, the two were practically the same. In the literature, the two terms are rarely distinguished and both are used to describe fixed length speaker embeddings based on the activations of a hidden neural network layer. Today, most state-of-the-art speaker diarization systems employ some type of d-vectors as speaker embeddings.

Fujita et al. (2019a) and Fujita et al. (2019b) stepped away from the d-vector standard with their *end-to-end neural diarization* (EEND) method. In fact, they removed the need for a separate speaker encoder entirely and used only a single neural network for the entire speaker diarization process.

3.1.1 Generalized End-To-End (GE2E) Loss

The speaker encoder, to be described in section 5.2, was based on the model used by Wang et al. (2018). Their paper described a diarization system where speaker embeddings were bottleneck features (BNFs) extracted from a neural network trained for speaker verification. They used the speaker verification model by Heigold et al. (2016) with improvements presented in Wan et al. (2018).

Speaker verification is the task of verifying whether a given utterance originates from a known speaker based on previous utterances from that speaker. The standard training and enrollment procedure consists of creating speaker models from so-called “enrollment” utterances. During inference, the system is given a new “evaluation” utterance that is compared to each speaker model. If the similarity is above a certain threshold the evaluation utterance is predicted to originate from the model’s speaker. Heigold et al. (2016) proposed a method where speaker models were estimated using only a small subset of a speaker’s enrollment utterances called the *tuple-based end-to-end* (TE2E) model. Unlike the standard speaker verification procedure, they did not estimate speaker models during enrollment. Instead, they calculated a new speaker model for each evaluation. In practice, this meant that most of the work done in the enrollment phase was moved to the evaluation phase. For evaluation, the system was fed the normal evaluation utterance together with a small subset of the enrollment utterances (around 5-6 in their experiments). A d-vector was then extracted for each of the utterances and a speaker model defined as the average of the enrollment d-vectors. This scheme was shown to outperform a baseline scheme where a static speaker model was defined as the average d-vector of all enrollment utterances.

Wan et al. (2018) continued the work from Heigold et al. (2016) and developed an improved training method of the verification model. For both systems, training consisted of adjusting the model such that each utterance embedding was as close to the belonging speaker’s centroid[1] as possible. For the model in Heigold et al. (2016), this meant that for each training utterance U, the distance from U to all speaker centroids was calculated one by one and the model adjusted accordingly. The optimization presented in Wan et al. (2018) was to process a larger number of utterances at once with a new loss function called *generalized end-to-end* (GE2E) loss. More specifically, instead of handling one utterance at a time, they took M utterances from N speakers simultaneously. The distance from each of the input utterances to all speaker centroids was then calculated and put into a single similarity matrix. The following update consisted of moving all of the $M \times N$ inputs towards their desired centroids. The optimization meant that the original TE2E model required at least $2 \times (N - 1)$ steps to perform the same amount of updates as the GE2E loss allowed for in a single step.

In Wang et al. (2018) the authors implemented a speaker diarization system that

[1]A speaker centroid was defined as the average of all utterance embeddings belonging to that speaker.

used a speaker encoder based on the verification system in Wan et al. (2018) using the GE2E loss. The encoder was trained on fixed-length speech segments as if it was for speaker verification but was used to extract bottleneck features (BNFs) for diarization. The process they used for embedding the audio was defined in two parts. First, sliding windows with 50% overlap were fed into the encoder, resulting in a d-vector (a set of BNFs) embedding for each window. Second, the audio was split into non-overlapping segments to be used for clustering. Segment embeddings were defined as the average of the L2 normalized d-vectors of all windows corresponding to the segment. The same embedding technique was subsequently employed by Zhang et al. (2019).

3.1.2 Improved Speaker Embeddings

Most speaker diarization systems in the literature take the average of smaller frame embeddings to represent larger segments that are then used for clustering. Dimitriadis (2019) proposed a new aggregation method that they argued was more robust in regards to noise and outliers. The proposed method consisted of two steps, a low-pass filtering and taking a median. The low-pass filter was a moving average defined as $F_j = F_{j-1} \times F_0$ for $j = 1, ..., N$ and $F_0[n] = \frac{1}{2}(\delta[n] + \delta[n-1])$ where $\delta[\cdot]$ was the Dirac function. This filter was applied to each individual d-vector before combining them to segment representations by taking their element-wise median. Before clustering, a final dimensionality reduction using *principal component analysis* (PCA, see section 2.2.5) was applied to the embeddings. However, Dimitriadis (2019) experimented with multiple clustering methods and two of these, spectral clustering and *improved deep embedded clustering* (IDEC) (Guo et al., 2017), have built-in dimensionality reduction before clustering. For these two methods, in particular, the PCA was therefore not applied before clustering. Using their proposed method, Dimitriadis (2019) reported significant improvements with a 22% relative improvement in diarization error rate (DER) for their best system.

3.2 Clustering Approaches for Speaker Diarization

In addition to highly discriminative features, speaker diarization systems are dependent on having a proficient clustering algorithm. This section first introduces the two most common unsupervised clustering methods used in the speaker diarization literature, *agglomerative hierarchical clustering* and *spectral clustering*. The latter part of this section presents two speaker diarization systems that implemented a supervised clustering module, the *unbounded interleaved-state recurrent neural network* and the *end-to-end neural diarization* method.

Historically, the most used approach in the literature has been to use agglomerative hierarchical clustering (AHC), also referred to as “bottom-up” clustering. The typical AHC scheme models each cluster as a Gaussian or a Gaussian mixture model (GMM) and has a stopping criterion based on the Bayesian information criterion (BIC). As proposed by Scott Shaobing Chen and Gopalakrishnan (1998), the BIC is used to evaluate whether two models are best modelled as a two separate or a single Gaussian, merging the two if the latter is the case. The scheme is still prominent in the field and was used as part of newer diarization systems Yella et al. (2014) Garcia-Romero et al. (2017) and as a baseline for others Zhang et al. (2019) Lin et al. (2019) Yoshioka et al. (2019).

Ning et al. (2006) proposed a spectral clustering scheme for speaker diarization which performed similarly to the AHC systems at the time. Following efforts had varying results with Luque and Hernando (2012) and Shum et al. (2012) not achieving any significant improvements compared to the standard AHC method. However, both groups did report better run times when using spectral clustering algorithms. Iso (2010) presented a spectral clustering scheme using a new speech segment encoding based on vector quantization. Unlike the other systems, this approach managed to outperform the AHC with BIC baseline but was not tested on the same datasets as other works. Whether the system in Iso (2010) was state-of-the-art at the time is therefore hard to say. Later, spectral clustering methods became more prominent in state-of-the-art publications. Wang et al. (2018) were one of the earliest to implement a diarization system using spectral clustering with d-vector embeddings.

At the time of writing, supervised machine learning methods have started to be used for clustering in speaker diarization systems. In the literature, most speaker diarization systems employ some type of d-vector embedding scheme in combination with either AHC or a spectral clustering algorithm. In other words, the majority of systems implement a supervised neural network as a speaker encoder and then cluster in an unsupervised fashion. An early work on supervised clustering was proposed by Zhang et al. (2019) with the *unbounded interleaved-state recurrent neural network* (UIS-RNN). Motivated by the fact that diarization systems were unable to take advantage of available labelled data, Zhang et al. (2019) proposed their "fully supervised" diarization system, the UIS-RNN. The system was fully supervised in the sense that both core components, the speaker encoder and clustering module, were trainable. While trainable speaker encoders were the norm, Zhang et al. (2019) made their system fully supervised by modelling speakers as instances of a recurrent neural network (RNN, section 2.2.2). The UIS-RNN however, had some limitations. First, the clustering module assumed that there would be no overlap in the speech data. This effectively made the system unable to handle any sort of overlap and also limited the type of training data that could be used. That is, the system required the training data to be split into segments with only a single speaker. Second, the model kept to the standard diarization framework of separate modules for the different diarization tasks. In particular, the speaker encoder was separate (and independent) from the clustering module. Further, the encoder was trained using data samples with only a single speaker, while the UIS-RNN required data of actual conversations. The UIS-RNN was therefore dependent on more training data than traditional systems, needing both single-speaker and multi-speaker data.

The end-to-end neural network-based speaker diarization model (EEND) by Fujita et al. (2019a) and Fujita et al. (2019b), tackled many of the limitations that the UIS-RNN ran into. The authors named their system end-to-end because the whole system consisted of a single neural network that learnt the entire diarization process. In short, the system was trained using conversational data and directly learnt to output diarization results.

That is, the separate speaker encoding process was removed. Additionally, the EEND did not assume single-speaker segments and was able to both be trained on overlapped speech and handle it during inference.

Here, a UIS-RNN was implemented as a supervised speaker diarization system. The model was chosen over the EEND because the focus was to experiment with different speaker embedding techniques (see research question 3 in section 1.2) and the EEND does not use a traditional speaker encoder. The following subsection goes into more detail about the UIS-RNN architecture and also presents a modified training scheme developed by Fini and Brutti (2020).

Unbounded Interleaved-State Recurrent Neural Network (UIS-RNN)

The core concept of the *unbounded interleaved-state recurrent neural network* (UIS-RNN) was to model speakers as instances of a recurrent neural network (RNN). That is, a single RNN was trained and an instance of the network created for each unique speaker found during inference.

As input, the system was given utterances as sequences of embeddings $X = (x_1, x_2, ...x_T)$, where each embedding represented a particular segment of the utterance. In particular, each embedding x_t was a d-vector representation (described in section 2.3.6) of segment t of the given utterance. For training, a sequence of truth labels $Y = (y_1, y_2, ..., y_T)$ was provided in addition to the embeddings, indicating the actual label of each segment. During inference, the UIS-RNN searched for the sequence of labels Ŷ such that the joint probability $P(X, Y)$ with $Y = \hat{Y}$ was maximized. Zhang et al. (2019) explicitly modelled speaker changes using a binary sequence $Z = (z_2, z_3, ...z_T)$ where $z_t = 1$ when $y_t \neq y_{t-1}$. That is, z_t was 1 when the speaker at time t was different from the speaker at time $t - 1$ and 0 when it was the same speaker.

Given these three sequence variables the authors defined the problem using the joint probability at time t, defined as[2]:

$$p(x_t, y_t, z_t \mid x_{[t-1]}, y_{[t-1]}, z_{[t-1]}) = \underbrace{p(x_t \mid x_{[t-1]}, y_t)}_{\text{sequence generation}} \times \underbrace{p(y_t \mid y_{[t-1]}, z_t)}_{\text{speaker assignment}} \times \underbrace{p(z_t \mid z_{[t-1]})}_{\text{speaker change}} \tag{3.1}$$

where $[t]$ was the ordered sequence $(1, 2, ..., t)$. The sequence generation modelled the authors' assumption that the sequence X of speaker embeddings was generated by a Gaussian distribution parameterized by the output of an RNN, specifically a gated recurrent unit (GRU, section 2.2.4). Speaker embeddings were modelled by $x_t \mid x_{[t-1]}, y_{[t]} \sim \mathcal{N}(\mu_t, \sigma^2 I)$ where μ_t was the averaged output of the RNN for speaker y_t.

The binary speaker changes were considered independent and modelled by a single parameter p_0, the probability of switching to another speaker. Zhang et al. (2019) set the probability of no speaker change, $p(z_t = 0)$, equal to the constant p_0. In practice, this meant that speaker changes were implemented as a weighted coin flip.

[2]Equation 3 in the original paper by Zhang et al. (2019).

The speaker assignment process was modelled as a distance dependent Chinese restaurant process (ddCRP, see section 2.1.5) where each speaker embedding x_t represented a new customer. Given that there was a speaker change, i.e. when $z = 1$, the process decided which speaker to assign the current frame to. A crucial point during this process was to decide whether the current speaker had been seen previously or if it was an entirely new speaker. This was modelled by a parameter α, a constant proportional to the probability of assigning the current frame to a new speaker.

The joint probability described by equation 3.1 only considered previous observations at each time step. This allowed the diarization system to perform predictions in an online fashion, that is, it could perform inference without having access to all observations from the start. This property is analogous to the ddCRP where each new customer only requires knowledge about the previously seated customers. The UIS-RNN model was, therefore, able to produce results online, as each assignment step was independent of future observations and the total number of observations.

Following the introduction of the UIS-RNN, Fini and Brutti (2020) proposed a new loss function for training the UIS-RNN, resulting in the modified UIS-RNN-SML. They also proposed an analytical method for estimating the α parameter of the ddCRP for the speaker assignment process. The loss function, called *sample mean loss* (SML), was calculated based on the mean of a given speaker's embeddings while the original loss function used a general mean, not speaker specific. The original loss used by Zhang et al. (2019) was formulated as:

$$\mathcal{L}_{MSE} = \sum_{i=1}^{|\mathcal{D}_A|} \sum_{j=1}^{|A_i|} ||a_{i,j} - \mu(GRU_\theta(a_{i,[j-1]}))||^2 \tag{3.2}$$

Where $\mathcal{D}_\mathcal{A}$ was a set of speaker sequences $A_i = (a_{i,1}, ...a_{i,L_i})$ with $a_{i,j}$ being the j-th embedding of A_i. The SML replaced the single embedding target with a mean of randomly sampled embeddings resulting in:

$$\mathcal{L}_{SML} = \sum_{i=1}^{|\mathcal{D}_A|} \sum_{j=1}^{|A_i|} ||\hat{\mu}_N(a_{i,[j,L_i]}) - \mu(GRU_\theta(a_{i,[j-1]}))||^2 \tag{3.3}$$

$\hat{\mu}_N(A_i)$ was defined as the average of N randomly sampled embeddings. In other words, the UIS-RNN-SML compared the mean of the network outputs for sequence i to the mean of a subset of unseen embeddings in the same sequence. Ideally, Fini and Brutti (2020) would have used the mean of the actual probability distribution, but that was not possible in practice. $\hat{\mu}_N$ was therefore used as an estimate of the distribution.

The second improvement proposed in Fini and Brutti (2020) was a method to estimate the α parameter of the ddCRP, the probability of switching to a new speaker. The parameter was crucial to the performance of the UIS-RNN (and UIS-RNN-SML) as it affected the total number of speakers it predicted. Specifically, a too large α would make the system overestimate the number of speakers and a smaller value would predict too few speakers. For speaker diarization in general, predicting the correct amount of speakers is an important (often implicit) subtask. For the UIS-RNN-SML, this meant that an imprecise value for α would negatively impact system performance and a proper estimate was therefore crucial. The solution proposed by Fini and Brutti (2020) was to estimate α directly from the training dataset using the following formula:

$$\alpha = \frac{\sum_{m=1}^{|\mathcal{D}|}(max(Y_m) - 1)}{\sum_{m=1}^{|\mathcal{D}|}\sum_{t=1}^{|Y_m|} 1(y_{m,t} \neq y_{m,t+1})} \tag{3.4}$$

Training with the modified loss function and using their estimated α, Fini and Brutti (2020) reported a significantly lower diarization error rate (DER) compared to the original UIS-RNN by Zhang et al. (2019). It should be noted, that Fini and Brutti (2020) tested their system on the DIHARD-II (Ryant et al., 2019) challenge dataset, while Zhang et al. (2019) tested their UIS-RNN on the 2000 NIST SRE CALLHOME dataset (described in section 4.2). The two groups also used different speaker encoders and their results are therefore not directly comparable. That being said, the relative improvement achieved by the UIS-RNN-SML compared to the benchmark UIS-RNN in Fini and Brutti (2020), suggested that the modifications were worthwhile.

4 Data

This chapter describes the datasets used for the experiments (see chapter 6). The datasets were divided into two classes based on whether a single or multiple speakers were present in the audio files. The single-speaker datasets were used for developing the speaker encoder and the multi-speaker datasets were used for diarization experiments[1].

4.1 Single Speaker Datasets

The purpose of a speaker encoder is to extract representations from speech segments that contain speaker specific information. Ideally, these speaker embeddings should also make it easy to distinguish between speakers. That is, two segments from a single speaker should be represented as similar as possible and segments originating from different speakers should be as different as possible. This section describes the three datasets used to train the speaker encoder described in section 5.2. Table 4.1 lists the number of speakers, the total number of utterances and a reference to each of the datasets. Table 4.2 lists the same information for the corresponding test partitions.

Set	**#Speakers**	**#Utterances**	**Reference**
LibriSpeech (train-other-500)	1166	148,043	Panayotov et al. (2015)
VoxCeleb1	1,211	148,642	Nagrani et al. (2017)
VoxCeleb2	5,994	1,092,009	Chung et al. (2018)

Table 4.1: Training data for the speaker encoder.

Set	**#Speakers**	**#Utterances**	**Reference**
LibriSpeech (test-other)	33	2,864	Panayotov et al. (2015)
VoxCeleb1 test	40	4,874	Nagrani et al. (2017)
VoxCeleb2 test	118	36,237	Chung et al. (2018)

Table 4.2: Test data for the speaker encoder.

[1]Speaker diarization on single speaker data is trivial.

4.1.1 LibriSpeech

The LibriVox project is a collection of free audiobooks that are read by volunteers in multiple languages under varying recording conditions[2]. Based on audiobooks found in the LibriVox project's library, Panayotov et al. (2015) prepared the LibriSpeech corpus consisting of read English speech.

The creators of LibriSpeech segmented the corpus into three non-overlapping subsets to make distribution more convenient. The subsets contain approximately 100, 360 and 500 hours of data respectively. The main criterion when the subsets were decided was a simple quality check. Quality was defined, for each file, by the word error rate (WER) of an automatic transcription process. The files were then sorted from lowest to highest WER before being divided into subsets. The two smaller sets of 100 and 360 hours were created by randomly selecting the files with the fewest transcription errors, leaving the largest subset with the lowest quality data. Because of this quality and size difference, the subsets are referred to as *train-clean-100*, *train-clean-360* and *train-other-500.* There are also corresponding development and test partitions for each of the subsets, such as *dev-other* and *test-other.*

For the experiments in chapter 6, LibriSpeech-other-500 and its corresponding test set were used. This decision was based on hard drive memory limitations (on the computing cluster) and time. If all three subsets were to be used, it would require significantly more time to download and process the data. The "other" partition was chosen because it was the largest single subset and also because it was more challenging due to the variations in quality. The corresponding test set, *test-other*, was also the most challenging of the subsets. For the rest of this Thesis, LibriSpeech will refer specifically to the *train-other-500* subset unless specified otherwise.

4.1.2 VoxCeleb

Nagrani et al. (2017) published the first VoxCeleb dataset *VoxCeleb1* in 2017. The speech was collected from YouTube videos and contains utterances from 1251 speakers[3]. A year later, the same authors published a new dataset *VoxCeleb2*, also extracted from YouTube videos but on a much larger scale (Chung et al., 2018). VoxCeleb2 contains speech from 6112 speakers with over 1 million utterances in total. Both of the VoxCeleb datasets are available for free online[4].

4.2 Multi Speaker Datasets

The goal of speaker diarization is to determine "who spoke when" and data with at least two speakers is therefore required for the task to make sense. This section presents a popular dataset in the speaker diarization field, the 2000 NIST Speaker Recognition Evaluation dataset. A specific partition of the dataset referred to as CALLHOME is

[2]`https://librivox.org/pages/about-librivox/`
[3]Total number of speakers between development and test partition.
[4]`http://www.robots.ox.ac.uk/~vgg/data/voxceleb/`

introduced. Table 4.3 lists the number of recordings, the number of speakers and a reference to the CALLHOME dataset.

This section was also meant to describe a set of simulated emergency calls that were to be provided by Laerdal for diarization experiments. However, the simulated emergency calls were not received in time because the consent forms required for handling the data were never received. The data is therefore not described in-depth, as it was never actually processed. Instead, a full explanation of the situation with the simulated emergency call data is given in section 4.2.2.

Dataset	**#Recordings**	**#Speakers**	**Reference**
CALLHOME	500	2-7	Przybocki and Martin (2001)

Table 4.3: Data for speaker diarization experiments.

4.2.1 2000 NIST Speaker Recognition Evaluation Disk-8: CALLHOME

2000 NIST Speaker Recognition Evaluation (SRE) Disk-8, typically referred to as CALLHOME, is a dataset by the National Institute of Standards and Technology (NIST) for the annual NIST Speaker Recognition Evaluation series started in 1996. The dataset from 2000 has been used extensively for the development and evaluation of speaker diarization systems since its release. It contains 500 samples of telephone conversations between two to seven unique speakers in six languages: English, German, Japanese, Spanish, Mandarin and Arabic.

4.2.2 Simulated Emergency Calls

Goal 2 and research question 3 in section 1.2 defined that the purpose of this Thesis was to investigate the performance of state-of-the-art diarization systems on a set of simulated emergency calls. The calls were to be provided by Laerdal Medical for the purposes mentioned above. The data was described (during project startup) as recordings of training sessions between an emergency call centre operator and a trainer. In each file, the trainer acted as if they needed emergency help and the operator was to get their information, send help and give them instructions on what to do while waiting for the ambulance to arrive. Because the data contained personal information, voiceprints of the people involved, consent had to be acquired from all people present in the data per the guidelines of the Norwegian Centre for Research Data (NSD). A notification form was created with support from NSD and the approved form was sent via Laerdal Medical to the relevant people. Unfortunately, signed consent forms were not received before the deadline (10. June 2020)[5] and the data was therefore not available for processing.

[5]This was the original deadline before it was extended to 19. June 2020.

5 Architecture

This chapter presents the architecture of the system implemented for the experiments in chapter 6. The implementation was based on the speaker diarization process used by Wang et al. (2018), illustrated in figure 5.1. The chapter is divided into three parts according to the three main steps of the process. Section 5.1 covers how the data was preprocessed before being fed to the speaker encoder. Section 5.2 describes the speaker encoder and different variations that were implemented, corresponding to the "Run LSTM" and "Aggregate" parts in the figure. Finally, section 5.3 presents the two clustering modules, UIS-RNN-SML and spectral clustering, that were implemented.

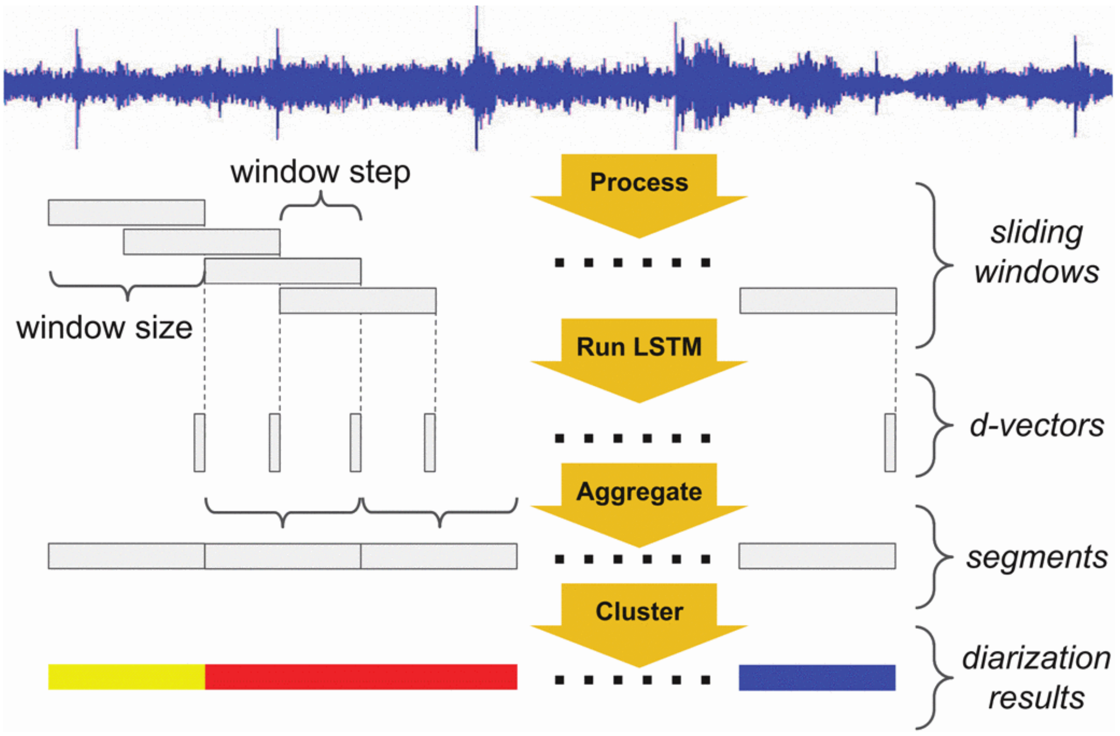


Figure 5.1: Flowchart of a d-vector based speaker diarization system by Wang et al. (2018). Used with permission.

5.1 Preprocessing

Before being fed to the encoder, each file was preprocessed as described in this section. This was done for both the single-speaker (section 4.1) and the multi-speaker files (section 4.2).

First, the sample rate of each file was verified to be 16kHz and resampled if necessary. This was done to ensure that all the data used the same sample rate. The specific value of 16kHz was selected because it was the native sample rate of the single-speaker data. After confirming the correct sample rate, the average volume of the files was normalized. During this step, all files with an average volume less than -30dB were normalized to -30dB. The purpose of this control was to remove variations in the data caused by differences in volume. The decision to normalize at -30dB was based on the fact that it was used successfully by Corentin (2019). The final preprocessing step was to remove silences from the files, but here different methods were used for the single-speaker and multi-speaker datasets. For removing silences in the single-speaker data, a voice activity detector (VAD) was used. In particular, the VAD developed for Google's WebRTC project[1]. The audio was split into 30ms frames and fed to the VAD to be classified as speech or non-speech. Segments consisting of six or more non-speech frames were considered as silences and removed. The VAD was not used to remove silences from the multi-speaker data. Instead, oracle timestamps were used to separate speech from non-speech. The oracle timestamps were based on the ground truth labels corresponding to the multi-speaker files, where each ground truth consists of a start time, end time and a speaker identifier. Based on this, silences were defined as the times where no speaker was present in the ground truth. Using ground truth labels to remove silences for diarization experiments is not unusual in the speaker diarization field. The main argument is that speaker diarization research is not focused on improving VAD performance, but errors by the VAD (false alarms or missed detections) negatively impact the diarization error rate (DER, section 2.1.7). Oracle timestamps are therefore used to ensure that the DER is not affected by a specific VAD's performance. This makes it easier to compare different systems because their error rates are only dependent on the diarization process itself. On the other hand, using oracle timestamps is not possible for most real-life applications. Thus, the performance of systems evaluated using oracle timestamps does not necessarily represent how they would perform outside of the research bubble. After standardising the audio files and removing non-speech segments, log-mel spectrograms were extracted using *librosa*, a Python package for audio analysis. The spectrograms were created using 25ms sliding windows with a 10ms step size and 40 log-mel channels.

5.2 Speaker Encoder

The speaker encoder was similar to the one used by Zhang et al. (2019) for the original UIS-RNN. The implementation was based on the open-source implementation by Corentin

[1] `https://webrtc.org/`

(2019)[2]. Figure 5.2 shows the encoder architecture. The model was a 3-layer LSTM with 768 hidden nodes, a fully connected projection layer and a final *rectified linear unit* (ReLU) layer. The ReLU, coloured red in the figure, was not present in the architecture used by Zhang et al. (2019) but was introduced in the implementation of Corentin (2019). Based on the results of experiment 1, reported in section 6.3.1, the extra ReLU was used for the final diarization system. In short, the results of the experiment indicated that including the ReLU produced better embeddings, as discussed in section 7.1.1.

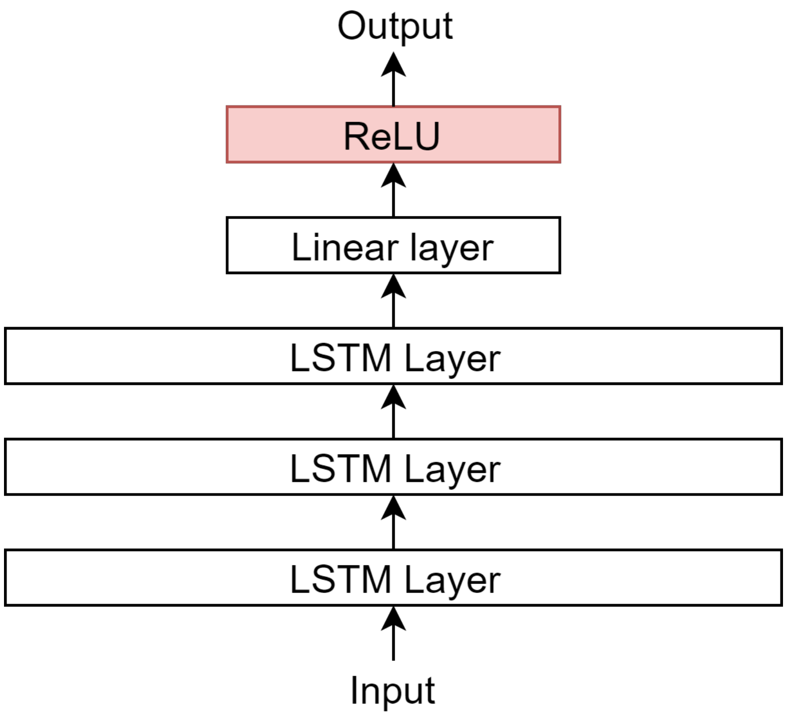


Figure 5.2: Speaker encoder architecture.

For training, the *generalized end-to-end* (GE2E) loss proposed by Wan et al. (2018) was used. The method was described in detail in section 3.1.1

Speaker embeddings were produced according to the d-vector definition described in section 2.3.6 and illustrated in figure 2.12. First, the audio was split into 240ms frames with 50% overlap. Embeddings were then extracted for each frame and aggregated into 480ms segment embeddings. Unless specified otherwise, the frame-level embeddings were aggregated by taking the element-wise average and L2 normalising them. For experiment 3, described in section 6.1.3, other aggregation methods were implemented and tested. These alternative aggregation methods are described next.

[2]`https://github.com/CorentinJ/Real-Time-Voice-Cloning`

Aggregation Methods

The standard method for extracting speaker embeddings in the literature follows the d-vector scheme described in section 2.3.6 and illustrated in figure 2.12 on page 19. In this section, the focus is on the last part of the extraction process illustrated in figure 5.3, the aggregation step.

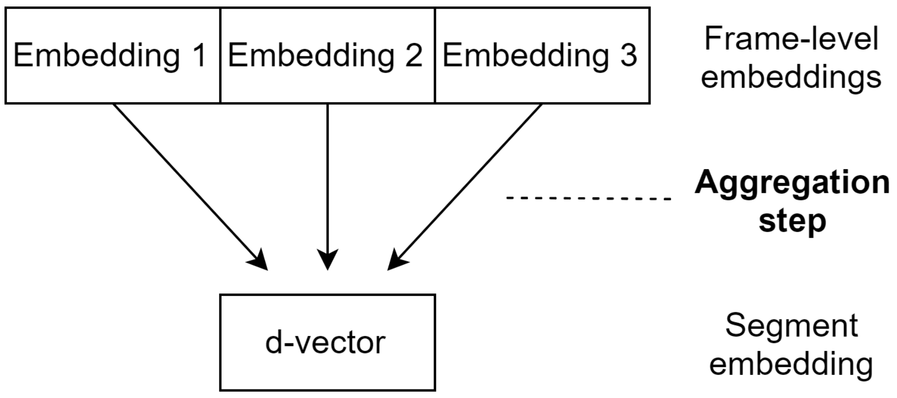


Figure 5.3: Aggregation of frame-level embeddings to a segment embedding.

Usually, embeddings from overlapping frames are combined to create segment embeddings by taking the element-wise average of the frame-level embeddings. A goal of this Thesis was to investigate other methods for combining the frame-level embeddings. Specifically, enhancements proposed by Dimitriadis (2019) were implemented and tested. The enhancements were described in detail in section 3.1.2. To recap, the proposed changes consisted of applying a moving average filter on the frame-level embeddings, before aggregating and taking the element-wise *median* instead of the average. Lastly, an optional dimensionality reduction, specifically Principal Component Analysis (PCA, section 2.2.5), was applied to the segment embeddings before clustering.

For the experiments, tests were also run using only parts of the modifications suggested by Dimitriadis (2019). By experimenting with only parts of the changes the impact of each change could be investigated. The following six schemes were implemented: Because the encoder produced sparse embeddings (see section 6.3.1), the *TruncatedSVD* package from scikit-learn (Pedregosa et al., 2011) was used in place of the basic PCA as it is better suited for handling sparse data.

5.3 Clustering

For speaker clustering, both a UIS-RNN-SML model and a spectral clustering module were implemented. The primary focus was on the UIS-RNN-SML, while the spectral method was included to have a separate system for comparison to accomplish Goal 2 (see section 1.2). A spectral method was specifically chosen as it was a popular state-of-the-art approach with a state-of-the-art implementation available from Wang et al. (2018).

5.3.1 Unbounded Interleaved-State Recurrent Neural Network (UIS-RNN)

The clustering module of the speaker diarization system was built using the *unbounded interleaved-state recurrent neural network* with *sample mean loss* (UIS-RNN-SML) model, introduced in section 3.2. The decision to use the UIS-RNN-SML over the original UIS-RNN was based on the results of experiment 2, described in section 6.1.2. The results showed that the UIS-RNN-SML performed better than the UIS-RNN, given the same parameters and amount of time. The implementation was based on the open-source code of Fini and Brutti (2020)[3] using PyTorch (Paszke et al., 2019) with layers as illustrated in figure 5.4. The components were an RNN layer, specifically a gated recurrent unit (GRU, section 2.2.4) with 512 nodes, followed by two fully connected layers with a ReLU operation in between.

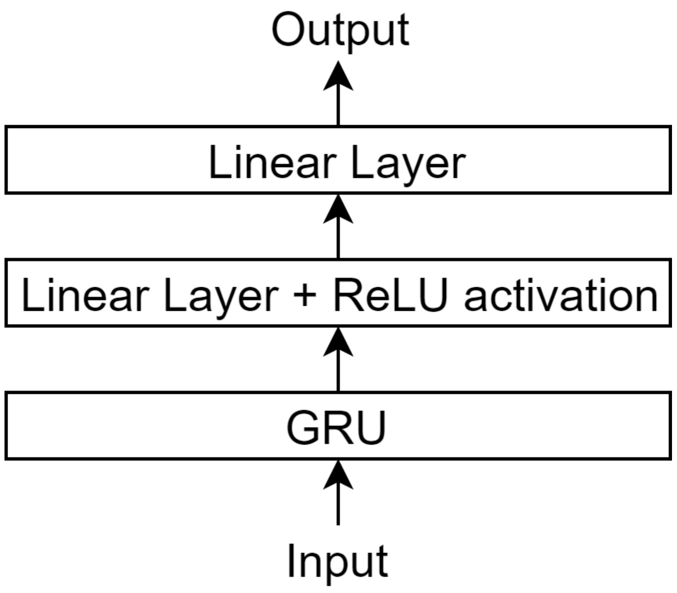


Figure 5.4: Architecture of the RNN used for the UIS-RNN-SML.

5.3.2 Spectral Clustering

A spectral clustering system using the implementation published by Wang et al. (2018)[4] was prepared as an unsupervised clustering system. The purpose was to compare the performance of the UIS-RNN-SML and the spectral clustering system on the simulated emergency calls. However, because permission to handle the data was not received (see section 4.2.2) the spectral clustering system was not used in any of the experiments.

[3]`https://github.com/DonkeyShot21/uis-rnn-sml/`
[4]`https://github.com/wq2012/SpectralCluster`

6 Experiments and Results

This chapter presents the experimental results. The goal of the experiments was to investigate the performance of the system described in chapter 5 using different speaker embedding methods. The chapter is structured as follows: section 6.1 introduces the experiments and their purposes, section 6.2 covers the experimental setups to make future reproduction easier and section 6.3 presents the experimental results.

Four experiments were initially planned, but only three were carried out. The fourth, and final experiment was planned to answer research question 3, comparing the performance of a supervised diarization system and an unsupervised system on simulated emergency calls. The supervised system was to use the architecture and techniques that gave the best results in experiments 1 through 3. Because the amount of simulated emergency call data was expected to be insufficient for properly training the supervised system, the 2000 NIST SRE CALLHOME (section 4.2) was to be used for off-domain training. A spectral clustering system was to be used as the unsupervised method. In the end, the experiment was not possible to carry out because the consent forms required to handle the simulated emergency calls were not received in time. The situation was described in detail in section 4.2.2.

6.1 Experimental Plan

This section introduces the three experiments that were carried out. Leading up to the tests of different speaker embedding aggregation methods, two experiments were planned to investigate implementation details of the speaker encoder and clustering module.

6.1.1 Experiment 1: Encoder Architecture

This experiment was planned as part of the development of the speaker diarization system used for testing speaker embedding aggregation methods. Its purpose was to determine the details of the speaker encoder architecture. In particular, the addition of a *rectified linear unit* (ReLU) layer as done by Corentin (2019) was investigated. Corentin (2019) did not document the effects of the ReLU and this experiment was planned to determine if it should be included in the architecture.

6.1.2 Experiment 2: UIS-RNN Loss Function

The speaker diarization system used the *unbounded interleaved-state recurrent neural network* (UIS-RNN) developed by Zhang et al. (2019). As presented at the end of section

3.2, Fini and Brutti (2020) proposed a set of modifications for the UIS-RNN resulting in the UIS-RNN-SML. Their modified version reported better results on the DIHARD-II dataset and this experiment was planned to investigate if similar results could be achieved on the 2000 NIST SRE CALLHOME dataset (see section 4.2). The UIS-RNN-SML was to be used for the final experiment if it performed better on the CALLHOME dataset compared to the UIS-RNN.

6.1.3 Experiment 3: Aggregation Methods

This experiment was planned to answer research question 2 directly, determining how different embedding aggregation methods impact speaker diarization performance. Alternative methods for the speaker embeddings were implemented and tested, inspired by the enhancements proposed in Dimitriadis (2019) (see section 3.1.2). The specific methods were described in section 5.2 on page 36.

6.2 Experimental Setup

This section covers the experimental setups. Before each experiment, the data was preprocessed following the procedure described in section 5.1. Each file was normalized, silences removed and spectrograms extracted using 25ms sliding windows with a 10ms step size.

6.2.1 Experiment 1: Encoder Architecture

Two encoders were trained and tested using the same data and parameters. Both consisted of a 3-layer LSTM with 768 hidden nodes and a linear projection layer with 256 nodes as illustrated in figure 5.2. However, only one of the encoders used the ReLU coloured in red. From here on, the encoder with a ReLU is referred to as *encoder R* and the other as *encoder N* (no ReLU). The training data consisted of LibriSpeech (train-other-500) and the training partitions of VoxCeleb1 and VoxCeleb2. Each training batch had 10 utterances from 64 speakers, 640 utterances in total. The training was run for 7 days on the NTNU IDUN/EPIC computing cluster (Själander et al., 2019) with a learning rate of 0.0001. For evaluation, the test partitions corresponding to the training datasets were used to find the equal error rate (EER, see section 2.1.8). The EER was calculated over 10 epochs using the same batch configurations as for training, 10 utterances from 64 speakers.

6.2.2 Experiment 2: UIS-RNN Loss Function

The UIS-RNN and UIS-RNN SML were tested using 5-fold cross-validation on the 2000 NIST SRE CALLHOME dataset. Embeddings were extracted using the best performing encoder from experiment 1 (encoder R) with oracle timestamps. A relatively short training time of 50 epochs with 2000 iterations each was used to make the experiments proceed smoothly. More epochs or iterations were deemed unnecessary because the purpose of the experiment was to uncover differences between the UIS-RNN and the UIS-RNN-SML, not maximizing their performance. The number of loss samples for the sample mean loss (SML) was set to 2, the best performing value in Fini and Brutti (2020). Finally, the α parameter for the distance dependent Chinese restaurant process (ddCRP, see section 2.1.5) was estimated using formula 3.4 on page 28 and set to 0.06.

6.2.3 Experiment 3: Aggregation Methods

For this experiment, based on the results of experiment 1 and 2, the UIS-RNN-SML was used together with encoder R. The encoder had been allowed to train for an additional 14 days, completing approximately 437,000 steps and achieving an EER of 10.58%. Embeddings were extracted with the different aggregation methods and used to perform 5-fold cross-validation. Each fold was run for 1000 epochs with 1000 iterations each. The learning rate was set to 0.001 and halved at the start of every epoch. Prediction results were refined three times before calculating the diarization error rate (DER) using the *pyannote.metrics* toolkit by Bredin (2017). The evaluation followed the conventions for the DER described in section 2.1.7, allowing errors less than 250ms between segments and excluding overlapped speech.

6.3 Experimental Results

This section presents the results of the experiments described previously in this chapter. Obvious implications are noted, but detailed discussions are deferred to chapter 7.

6.3.1 Experiment 1: Encoder Architecture

The results of experiment 1 are shown in table 6.1, where *encoder R* was the encoder with an extra *rectified linear unit* (ReLU) operation and *encoder N* the one without. In addition to the EER, the table lists the average number of zeros in the embeddings and the total number of training steps. Encoder R had the lowest EER and was therefore used to create embeddings for experiments 2 and 3. The other results indicate that encoder R produced sparser embeddings than encoder N, but required more time to complete each training step.

	EER	Avg. number of zeros	# Training steps
Encoder R	**11.2%**	173	$\sim 200k$
Encoder N	12.8%	127	$\sim 250k$

Table 6.1: EER of the speaker encoders.

6.3.2 Experiment 2: UIS-RNN Loss Function

Table 6.2 presents the average DER of the UIS-RNN and UIS-RNN-SML using 5-fold cross-validation on the CALLHOME dataset. Both error rates were far from state-of-the-art, but this was to be expected with the minimal amount of tuning and training. The purpose of this experiment was to compare the relative performance of the two models, and the results showed that the UIS-RNN-SML significantly outperformed the UIS-RNN. Experiment 3 was, therefore, carried out using the UIS-RNN-SML.

	DER
UIS-RNN	38%
UIS-RNN-SML	**29%**

Table 6.2: Average DER from cross-validation with 5-folds of UIS-RNN and UIS-RNN-SML.

6.3.3 Experiment 3: Aggregation Methods

The diarization error rates using the different aggregation methods are presented in table 6.3. Each DER was the average error rate achieved when using the different aggregation techniques and 5-fold cross-validation on the 2000 NIST SRE CALLHOME dataset. The best result was achieved when applying the filter and combining frame-level embeddings by taking the median, giving a DER of 13.72%. Comparable error rates were achieved when skipping the filter and only doing dimensionality reduction before clustering. On the other hand, no improvements were apparent when only applying the filter.

Aggregation method	DER
Average	18.79%
Median	18.96%
Average + PCA	14.17%
Median + PCA	14.50%
Filter + Median	18.97%
Filter + Median + PCA	**13.72%**

Table 6.3: Average DER from cross-validation with 5-folds using different embedding aggregation methods.

7 Evaluation and Discussion

In this chapter, the results of the experiments in the previous chapter are evaluated and discussed. The chapter also addresses new questions that may arise throughout the discussions. Section 7.1 evaluates the results in regards to the goals set in section 1.2. In section 7.2, the performance of the system is discussed and compared to other work in the speaker diarization field.

7.1 Evaluation

This section evaluates the results of the experiments in chapter 6 and their contributions to the research questions.

7.1.1 Experiment 1: Encoder Architecture

Experiment 1 was planned to decide on whether a rectified linear unit (ReLU) should be part of the speaker encoder architecture in the following experiments. The results were reported in table 6.1 on page 41 and showed that the encoder with a ReLU layer (encoder R) had the lowest equal error rate (EER). This suggests that encoder R produced more discriminative embeddings than encoder N. It also produced significantly sparser embeddings indicating that it was able to represent the data more efficiently. The increased sparsity was a natural consequence of the ReLU, forcing the encoder to use non-negative values. However, sparser embeddings would not necessarily imply a lower EER. An argument could be made that denser embeddings contain more information and should allow for better separability. The fact that the sparse embeddings outperformed their denser counterparts suggested otherwise. A possible explanation would be that the ReLU acted as a sort of filter to remove unnecessary and redundant information, making the embeddings easier to analyse. Determining the exact difference between the embeddings would require further investigation, but interpreting such abstract representations is difficult and often in vain.

The cost of acquiring the more powerful embeddings was an increase in complexity, evident by the number of training steps completed. Given the same amount of time (7 days), encoder R performed 50,000 training steps less than encoder N. In other words, including the ReLU increased the amount of time required for each training step. This attests to the ability of encoder R, performing better with less refinement.

Based on the experimental results, the evidence pointed towards using encoder R for the other experiments. This did not directly contribute towards answering the research questions, but it was crucial for justifying the encoder architecture. Experiment 1 therefore served its purpose. Further, although not a specific goal, the results contribute to the research on feature extraction for speaker diarization.

7.1.2 Experiment 2: UIS-RNN Loss Function

Experiment 2 was meant to aid in the decision on whether the *sample mean loss* (SML) by Fini and Brutti (2020) should be used to train the unbounded interleaved-state recurrent neural network (UIS-RNN). In particular, which of the UIS-RNN and the UIS-RNN-SML would perform best on the 2000 NIST SRE CALLHOME dataset (see section 4.2). The experiment was carried out using 5-fold cross-validation and equivalent hyperparameters for the models. Table 6.2 on page 42 presented the diarization error rates (DERs) and showed that the UIS-RNN-SML significantly outperformed the UIS-RNN. With both models having equivalent configurations, a reasonable conclusion is that the UIS-RNN-SML is more potent than the UIS-RNN. Thus, the results of experiment 2 were aligned with the conclusions of Fini and Brutti (2020). Further, the difference in performance gave clear evidence for which model to use for the rest of the work.

Experiment 2 answered its intended question and contributed by warranting the use of the UIS-RNN-SML over the UIS-RNN. The results also contribute to research on the UIS-RNN model and its variations by providing evidence for the effectiveness of the SML.

7.1.3 Experiment 3: Aggregation Methods

For experiment 3, multiple aggregation methods were implemented and tested to answer research question 2. Table 6.3 on page 42 presented the results and showed varying degrees of improvement. The standard *average* set the baseline DER for the other methods at 18.79%. No improvements were achieved by the *median* aggregation, even performing slightly worse. However, the minimal difference did not come as a surprise. The embeddings were extracted from short 240ms frames and the values of adjacent embeddings should therefore be fairly similar. Taking the median should, therefore, not result in significantly different values and the system could be expected to perform comparably to the baseline. A similar argument can be made for the *filter+median*. By definition, the moving average filter practically performed the same operation as the *average*. Thus, *filter+median* is approximately the same as combining the basic *average* with the *median*. A proportional error rate is therefore not surprising. Significant improvements were obtained in all cases where the principal component analysis (PCA) was applied. This suggests that the dimensionality reduction successfully produced embeddings that were easier to cluster. The PCA also seemed to have more impact when used in combination with the moving average filter, giving the lowest diarization error rate (DER) of 13.72%. However, the gap between the *average+PCA* and *filter+median+PCA* was relatively small and the significance of the filter requires further research. In the end, the impact of the aggregation methods was documented through experiment 3 providing an answer to research question 2 as planned.

7.2 Discussion

In the previous section, the experiments were evaluated in light of their intended goals and contributions. This section discusses the performance of the implemented speaker diarization system in regards to other works in the literature. In particular, the speaker encoder from experiment 1 is compared to the original implementation by Corentin (2019). Finally, the full diarization system from experiment 3 is compared to the performance of the original UIS-RNN by Zhang et al. (2019).

7.2.1 Evaluation of the Speaker Encoder

The encoder implementation was based on the work of Corentin (2019) who reported an equal error rate (EER) of 4.5%. In comparison, the encoder from experiment 1 fell significantly short with 11.2% EER. A possible explanation for the discrepancy is the difference in training. In Corentin (2019) the encoder was allowed to complete 1 million training steps, whereas experiment 1 was stopped at 200,000 training steps. To make the comparison fairer, the encoder used for experiment 3 could be used instead. As mentioned previously in section 6.2.3, the encoder from experiment 1 was trained further while preparing experiment 3. The extended training reduced the EER to 10.58%, completing a total of 437,000 steps. Any further training was limited due to time constraints. Judging by the encoder's improvement from experiment 1 to experiment 3, it is reasonable to ascribe some of the difference in performance to lack of training. Still, the relative rate of improvement does not indicate that the encoder would match an EER of 4.5% even with 1 million steps.

Another difference between the speaker encoder and the encoder in Corentin (2019) was the architecture. As described in section 5.2, the implementation had 768 nodes in each hidden layer. This was adopted from the encoder used for the original UIS-RNN in Zhang et al. (2019). On the other hand, Corentin (2019) used a model with only 256 nodes in each hidden layer. The difference in performance could then suggest that the smaller model was more suited for the single speaker datasets (see section 4.1). More nodes do not necessarily imply a more powerful system and network architectures should be tailored to their specific contexts. With limited time, the larger network might have been especially detrimental to the performance of the encoder. As found when comparing encoder R to encoder N in experiment 1, larger models require more time to complete each training step. This explains why the amount of training for the encoder was unsatisfactory. In hindsight given the time limitation, a smaller encoder network would have been a better choice.

7.2.2 Evaluation of the Fully Supervised Speaker Diarization System

In this section, the performance of the best speaker diarization system from experiment 3 is compared to the original *unbounded interleaved-state recurrent neural network* (UIS-RNN) by Zhang et al. (2019). It was not compared to the UIS-RNN-SML (UIS-RNN with sample mean loss) by Fini and Brutti (2020) because they used a different dataset. In their paper, Zhang et al. (2019) reported the results of multiple diarization experiments using encoders trained on varying amounts of single speaker data. The most appropriate baseline for this Thesis was the results using *V1*, their simplest encoder. V1 was trained using 36 million single speaker utterances, approximately 36 times more data than available for the experiments. Using encoder V1 and 5-fold cross-validation, Zhang et al. (2019) achieved a diarization error rate (DER) of 11.7%. In comparison, the lowest DER obtained in experiment 3 was 13.72% and fell 2.02% short of the baseline. The difference in training data could explain why the system failed to match the baseline. Experiment 3 indicated that the embeddings had a significant impact on diarization performance and the quality of the speaker encoder is therefore pivotal. Further, the speaker encoder quality is highly dependent on the available training data. Therefore, with 36 times more sample utterances, Zhang et al. (2019) had a superior basis for their system.

In view of the tremendous disparity in training data, the difference in performance was astonishingly small. Looking at the results of experiment 3, the standard *average* aggregation method gave an error rate of 18.79%. This suggests that the alternative aggregation method was able to mostly make up for the lack of data. The question then arises as to whether the *filter+median+PCA* would have the same impact in combination with a better trained encoder. In either case, the reported results provided evidence that the enhancements by Dimitriadis (2019) were able to increase the performance of the speaker diarization.

The tuning, or lack thereof, could also explain the experimental results. Machine learning algorithms are dependent on their hyperparameter values and proper tuning can improve their performance significantly. For this work, no particular tuning was performed. Given that each run of experiment 3 took around 20 hours, exploring hyperparameter configurations would require a considerable amount of time not available for this work. Besides, the goal was to investigate the (relative) impact of different speaker embedding methods and fine tuning was therefore not essential. Overall, considering the data discrepancy and the minimal tuning, a difference of 2.02 percentage points was reasonably close to the baseline.

8 Conclusion and Future Work

This chapter concludes the Thesis. First, it revisits the goals and research questions defined in the introduction, and the answers to the questions are presented. Then, the contributions to the speaker diarization field are described. Finally, the last section presents some thoughts about what and how new questions could be addressed in future work.

8.1 Goals and Research Questions

In section 1.2, the goals and research questions were formulated. Here, they are reiterated and evaluated.

Goal 1 *Investigate the performance of state-of-the-art speaker diarization systems using different speaker embedding methods.*

To evaluate this goal, research questions 1a and 1b were defined.

Research question 1a *How are state-of-the-art Speaker Diarization systems built?*

Most speaker diarization systems contain at least four components, see section 2.1: feature extraction, voice activity detection, segmentation and clustering. At the time of writing, state-of-the-art systems use neural networks as feature extractors and either a spectral clustering algorithm (unsupervised) or a supervised clustering algorithm (see chapter 3).

Research question 1b *What features do state-of-the-art Speaker Diarization systems use?*

Mel frequency cepstral coefficients (MFCCs) were traditionally the most used features in speaker diarization but were later replaced by bottleneck features (BNFs) extracted using neural networks. In most literature, these are referred to as speaker embeddings or d-vectors.

Goal 2 *Determine whether an unsupervised or a fully supervised speaker diarization system performs best on simulated emergency calls.*

The second goal was to find an architecture for a speaker diarization system that could be applied successfully to a dataset of simulated emergency calls. To evaluate this goal, research questions 2 and 3 were defined. The purpose of question 2 was to determine how to extract speaker embeddings for the diarization systems to be used on the simulated emergency calls. Question 3 was formulated to determine which of the two methods, unsupervised and supervised, would perform best on the simulated emergency calls.

Research question 2 *How do different embedding aggregation methods impact system performance?*

The results of experiment 3 showed that applying dimensionality reduction on the speaker embeddings significantly improved the performance of the supervised diarization system. The moving average filter proposed by Dimitriadis (2019) did not have as big of an impact but did result in slightly better performance. Using the standard average scheme achieved a diarization error rate of 18.79%. Adding the dimensionality reduction reduced the error to 14.17% and the final combination of the moving average filter. The best performance with an error rate of 13.72% was achieved by combining frame-level embeddings by taking the median in addition to using the filter. Thus, different aggregation methods, in particular the dimensionality reduction, had a positive impact on overall system performance.

Research question 3 *How does a supervised diarization system trained on off-domain data perform compared to an unsupervised method on simulated emergency calls?*

The consent forms required to process the simulated emergency calls were not received and this research question could, therefore, not be answered.

Research question 2 was answered successfully, but goal 2 was not reached because research question 3 could no be answered.

8.2 Contributions

A contribution is made to the speaker diarization field by investigating the inclusion of a *rectified linear unit* (ReLU) to an encoder using *bottleneck features* (BNFs) in experiment 1. Including the ReLU was found to produce sparser embeddings that in turn produced better speaker verification results. This contributes to the effort in the speaker diarization field focused on extracting discriminative embeddings by using encoders trained as verification systems.

The work also contributes to the research on speaker embeddings for speaker diarization by investigating the effects of different embedding aggregation methods. That is, most speaker diarization systems create segment-level embeddings by taking the average of smaller frame-level embeddings. Dimitriadis (2019) proposed an alternative method where a moving average filter is applied to the frame-level embeddings before combining them by taking the median instead of the average. For experiment 3, the alternative method was implemented and used with an *unbounded interleaved-state recurrent neural network* (UIS-RNN) trained using *sample mean loss* (SML) for speaker diarization. This contributes to the little amount of research done on aggregation methods by applying the variation in Dimitriadis (2019) to a different system architecture. Varying effects of the different enhancement elements were also found. Hopefully, this could provide insight for future experiments on speaker embeddings.

An additional contribution was supposed to be made by answering research question 3 and recommending whether a supervised diarization system should be used over an unsupervised system for use on simulated emergency calls. As described in section 4.2.2, the consent forms required to process the simulated emergency calls were not received in time and research question 3 could not be answered (see the introduction of chapter 6).

8.3 Future Work

During the work, new questions and directions for improvements became apparent. This section presents some ideas for future work.

Speaker Diarization of Simulated Emergency Calls

The motivation when defining the Thesis work was to investigate the performance of different speaker diarization systems on simulated emergency calls. This could not be performed due to the missing consent forms. A future effort would be to prepare and execute speaker diarization experiments on the simulated emergency calls. Specifically, investigating whether the supervised diarization system using the embedding enhancements would perform better on the simulated emergency calls compared to an unsupervised spectral clustering algorithm.

Effect of ReLU on Speaker Encoder Architectures

The results of experiment 1 showed that including the ReLU increased the speaker verification performance of the encoder and it also resulted in sparser embeddings. Future work could investigate the effect of adding a rectified linear unit (ReLU) as a final layer to other encoder architectures. This could be performed by adding a ReLU to different encoder architectures and examining its impact on performance. If the modification successfully improves the performance of other architectures it could improve the general encoder template.

Effect of Moving Average Filter

By itself, the moving average filter did not have any significant effect on the diarization error rate. A future effort could aim to find out whether this was due to the specific encoder architecture. In particular, a long short-term memory (LSTM) was implemented, while Dimitriadis (2019) used a *time delay neural network* (TDNN). Investigating whether the individual impact of the moving average filter is greater for a TDNN encoder would provide more knowledge about the effects of the proposed filter. A particularly interesting case is if the filter (without a following dimensionality reduction) never has a big impact. This would suggest that the filter, to some extent, depends on a dimensionality reduction to provide any noticeable improvement to the diarization. Future work could investigate whether the filter has any effect at all, or if only the dimensionality reduction is necessary. This could be carried out by comparing the performance of multiple clustering methods using the moving average filter, with and without the dimensionality reduction.

Training and Tuning

The encoder used for experiment 3 was trained for a relatively short amount of time and had a very limited amount of training data available. In theory, this could have handicapped the speaker diarization system by producing speaker embeddings that were unnecessarily hard to separate. Assuming that, with more resources during training, the encoder would have produced more discriminative embeddings. A question to answer in future work would be: Do embeddings from a more trained encoder have greater effect from the enhancements? If the answer is yes, it could suggest that the enhancements by Dimitriadis (2019) can accentuate already existing differences between embeddings, making speakers easier to distinguish. Further, it would highlight the importance of high-quality embeddings.

Finally, it would be interesting to explore the potential of the final diarization system by putting effort into fine tuning parameters and also increasing the amount of training. In combination, these measures would most likely increase the overall performance of the system. However, this can be said for practically all machine learning systems and the amount of tuning and training has to be limited at some point. Still, investigating whether the current encoder and clustering architecture would be able to match the baseline diarization error rate of 11.7% would be interesting.

Bibliography

Joyanta Basu, Soma Khan, Rajib Roy, Madhab Pal, Tulika Basu, Milton Samirakshma Bepari, and Tapan Kumar Basu. An overview of speaker diarization: Approaches, resources and challenges. In *2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA)*, pages 166–171, Bali, Indonesia, 2016. IEEE.

Hervé Bredin. pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems. In *INTERSPEECH 2017, 18th Annual Conference of the International Speech Communication Association*, pages 3587–3591, Stockholm, Sweden, 2017. ISCA. URL `http://pyannote.github.io/pyannote-metrics`.

Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, pages 1086–1090, Stockholm, Sweden, 2018. ISCA.

Jemine Corentin. Automatic multispeaker voice cloning. Master's thesis, Université de Liège, Liège, Belgia, 2019.

Najim Dehak, R. Dehak, Patrick Kenny, Niko Brummer, Pierre Ouellet, and Pierre Dumouchel. Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification. In *INTERSPEECH*, pages 1559–1562, Brighton, United Kingdom, 2009. ISCA.

Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2010.

Dimitrios Dimitriadis. Enhancements for audio-only diarization systems. *Computing Research Repository (CoRR)*, abs/1909.00082, 2019. URL `https://arxiv.org/abs/1909.00082`.

Enrico Fini and Alessio Brutti. Supervised online diarization with sample mean loss for multi-domain data. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7134–7138. IEEE, 2020.

Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Kenji Nagamatsu, and Shinji Watanabe. End-to-end neural speaker diarization with permutation-free objectives. In *INTERSPEECH*, pages 4300–4304, Graz, Austria, 2019a. ISCA.

Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe. End-to-end neural speaker diarization with self-attention. In *IEEE*

Automatic Speech Recognition and Understanding Workshop (ASRU), pages 296–303, Sentosa, Singapore, 2019b. IEEE.

Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree. Speaker diarization using deep neural network embeddings. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4930–4934, New Orleans, LA, USA, 2017. IEEE.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1753–1759, Melbourne, Australia, 2017. IJCAI.

Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119, Shanghai, China, 2016. IEEE.

Ken-Ichi Iso. Speaker clustering using vector quantization and spectral clustering. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4986–4989, Dallas, TX, USA, 2010. IEEE.

Runar Ask Johannessen. Virtual standardized patients: State-of-the-art. Project report, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2019. Part of the course TDT4501. Unpublished.

Qingjian Lin, Ruiqing Yin, Ming Li, Hervé Bredin, and Claude Barras. LSTM based similarity measurement with spectral clustering for speaker diarization. In *INTERSPEECH*, pages 366–370, Graz, Austria, 2019. ISCA.

Jordi Luque and Javier Hernando. On the use of agglomerative and spectral clustering in speaker diarization of meetings. In *Odyssey - The Speaker and Language Recognition Workshop*, pages 130–137, Singapore, 2012. ISCA.

Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. VoxCeleb: a large-scale speaker identification dataset. In *INTERSPEECH*, pages 2616–2620, Stockholm, Sweden, 2017. ISCA.

Huazhong Ning, Ming Liu, Hao Tang, and Thomas Huang. A spectral clustering approach to speaker diarization. In *INTERSPEECH*, pages 2178–2181, Pittsburgh, PA, USA, 2006. ISCA.

Christopher Olah. Understanding LSTM networks, August 2015. URL `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed: 01.June 2020.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, Brisbane, QLD, Australia, 2015. IEEE.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Mark Przybocki and Alvin Martin. *2000 NIST Speaker Recognition Evaluation LDC2001S97*. Linguistic Data Consortium, Philadelphia, 2001. URL `https://catalog.ldc.upenn.edu/LDC2001S97`. Web Download.

Neville Ryant, Kenneth Church, Christopher Cieri, Alejandrina Cristia, Jun Du, Sriram Ganapathy, and Mark Liberman. The second DIHARD diarization challenge: Dataset, task and baselines. In *INTERSPEECH 2019*, pages 978–982, Graz, Austria, 2019. ISCA.

Scott Shaobing Chen and P. S. Gopalakrishnan. Clustering via the Bayesian information criterion with applications in speech recognition. In *1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 645–648, Seattle, WA, USA„ 1998. IEEE.

Stephen Shum, Najim Dehak, and Jim Glass. On the use of spectral and iterative methods for speaker diarization. In *INTERSPEECH*, pages 482–485, Portland, OR, USA, 2012. ISCA.

Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure. *Computing Research Repository (CoRR)*, abs/1912.05848, 2019. URL `https://arxiv.org/abs/1912.05848`.

David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur. Deep neural network-based speaker embeddings for end-to-end speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 165–170, Florence, Italy, 2016. IEEE.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust DNN embeddings for speaker recognition. In *2018 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5329–5333, Calgary, AB, Canada, 2018. IEEE.

Alain Tritschler and Ramesh A. Gopinath. Improved speaker segmentation and segments clustering using the Bayesian information criterion. In *Sixth European Conference on Speech Communication and Technology (EUROSPEECH'99)*, pages 679–682, Budapest, Hungary, 1999. ISCA.

Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4052–4056, Florence, Italy, 2014. IEEE.

Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4879–4883, Calgary, Alberta, Canada, 2018. IEEE.

Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopez Moreno. Speaker diarization with LSTM. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5239–5243, Calgary, Alberta, Canada, 2018. IEEE.

Sree Harsha Yella, Andreas Stolcke, and Malcolm Slaney. Artificial neural network features for speaker diarization. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 402–406, South Lake Tahoe, NV, USA, 2014. IEEE.

Takuya Yoshioka, Zhuo Chen, Dimitrios Dimitriadis, William Hinthorn, Xuedong Huang, Andreas Stolcke, and Michael Zeng. Meeting transcription using virtual microphone arrays. *Computing Research Repository (CoRR)*, abs/1905.02545, 2019. URL `https://arxiv.org/abs/1905.02545`.

Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang. Fully supervised speaker diarization. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6301–6305, Brighton, United Kingdom, 2019. IEEE.

Bin Zhen, Xihong Wu, Zhimin Liu, and Huisheng Chi. On the importance of components of the MFCC in speech and speaker recognition. In *6th International Conference on Spoken Language Processing (ICSLP)*, volume 2, pages 487–490, Beijing, China, 2000. ISCA.

NTNU
Norwegian University of
Science and Technology