BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

UNSUPERVISED EVALUATION OF SPEAKER RECOGNITION SYSTEM

EVALUACE SYSTÉMU NA ROZPOZNÁVÁNÍ MLUVČÍHO NA NEZNÁMÝCH DATECH

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ONDŘEJ ODEHNAL**
AUTOR PRÁCE

SUPERVISOR **Ing. PAVEL MATĚJKA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Department of Computer Graphics and Multimedia (DCGM) Academic year 2021/2022

Bachelor's Thesis Specification

24991

Student:	**Odehnal Ondřej**
Programme:	Information Technology
Title:	**Unsupervised Evaluation of Speaker Recognition System**
Category:	Speech and Natural Language Processing

Assignment:

1. Explore current state of the art systems for speaker verification based on DNN and different clustering techniques.
2. Propose a methods for automatic control of audio quality and recording quality in the evaluation set (length of recording, Signal to Noise Ratio, recording similarity, ...).
3. Propose a method for unsupervised evaluation of the speaker verification system on data without annotation.
4. Evaluate your method and compare with the results on the set where the annotation is known.

Recommended literature:

- D. Snyder, D. Garcia-Romero, G. Sell, D. Povey and S. Khudanpur, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329-5333, doi: 10.1109/ICASSP.2018.8461375.
- Josef Slavíček and Albert Swart and Michal Klčo and Niko Brümmer, The Phonexia VoxCeleb Speaker Recognition Challenge 2021 System Description, https://arxiv.org/abs/2109.02052, 2021.

Requirements for the first semester:

- Body 1 až 2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Matějka Pavel, Ing., Ph.D.**
Consultant:	Klčo Michal, Ing., Phonexia
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2021
Submission deadline:	May 11, 2022
Approval date:	November 1, 2021

Abstract

The context of this thesis is the state-of-the-art system for speaker identification (SID) based on the DNN with x-vector embeddings. This thesis aims to propose and experimentally assess several techniques for evaluating the SID system using unlabelled datasets. For this purpose, discriminative embedding is created for every recording in the dataset. These embeddings are used to cluster the recordings and thus create pseudo-labels corresponding to different clusters. The SID system evaluation is based on equal error rate (EER), which uses these pseudo-labels. We proposed several unsupervised learning algorithms to achieve this; K-means, Gaussian mixture models (GMM), and agglomerative hierarchical clustering (AHC). After thorough testing, the K-means model with the Silhouette value showed the best results. This method achieved an estimate of 5.72 % EER with the reference EER equal to 5.15 % on SITW dev-core-core. Similar results were observed on the SITW eval-core-core, where the estimated EER is equal to 5.86 % and the reference 5.08 %. The difference between estimated and reference EER is 0.57 % for the dev-core-core and 0.78 % for the eval-core-core. Another series of experiments were conducted on NIST SRE16 and VoxCeleb1 to verify robustness of the proposed method. Generally, the developed testing process had an estimated error of around ± 1 % in all test databases, an excellent result for an unsupervised learning technique.

Abstrakt

Tato práce je vystavěna nad moderním systémem pro rozpoznávání mluvčího (SID) založeného na x-vektorech. Cílem bakalářské práce je navrhnout a experimentálně vyhodnotit techniky pro evaluaci SID systému za použití audio nahrávek bez anotace tj. bez znalosti mluvčího. Pro tento účel je z každé nahrávky bez anotace vytvořen embedding. Ty se poté používají pro shlukování nahrávek a následné vytvoření pseudo-anotací. Na těchto anotacích se SID systém evaluuje pomocí equal error rate (EER) metriky. Za účelem vytvoření pseudo-anotací byly navrženy tyto shlukovací algoritmy učení bez učitele: K-means, Gaussian mixture models (GMM) a aglomerativní shlukování. Po testování vyšel jakožto nejlepší experimentální postup K-means se Silhouette metrikou, která používá kosinovou podobnost jako míru vzdálenosti. Nejlepší metoda dosáhla $5,72$ % EER s referenčním EER $= 5,15$ % , které bylo spočítané se znalostí anotace na části datasetu SITW dev-core-core. Podobné výsledky byly získány na části datasetu SITW eval-core-core s odhadnutým EER $= 5,86$ % a referenčním $5,08$ %. Rozdíl mezi hodnotami tvoří $0,57$ % pro eval-core-core a $0,78$% pro dev-core-core. Další testy na NIST SRE16 a VoxCeleb1 datasetech byly provedeny za účelem ověření správnosti navrženého postupu. Obecně se dá říct, že navržený testovací postup měl chybu přibližně ± 1 %, což je poměrně dobrý výsledek pro algoritmus učení bez učitele.

Keywords

speaker recognition, speech verification, unsupervised learning, clustering, evaluation, GMM, AHC, EER, elbow method, K-means

Klíčová slova

rozpoznávání mluvčího, verifikace mluvčího, učení bez učitele, shlukování, evaluace, GMM, AHC, EER, K-means

Reference

ODEHNAL, Ondřej. *Unsupervised Evaluation of Speaker Recognition System.* Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Pavel Matějka, Ph.D.

Rozšířený abstrakt

Tato práce se zabývá vývojem, implementací a testováním techniky pro evaluaci systému na rozpoznávání mluvčího (SRE) na neznámých datech. Evaluace takového systému standardně probíhá na anotovaných datech, kde je předem jasné, kdo ve které nahrávce mluví. Avšak v praxi tohoto není vždycky možné dosáhnout, neboť tvorba takového datasetu je časově i finančně náročná. Ovšem velká část zákazníků, kteří chtějí SRE využívat, mají k dispozici velké množství audio dat, které nejsou anotované. Kdyby existoval způsob, jak otestovat SRE systém na těchto neznámých datech, tak by se otevřel přístup k velkému množství testovacích dat a bylo by tak možné je využít pro evaluaci ještě před tím, než bude SRE systém nainstalován v produkčním prostředí.

Jakožto SRE systém je v této práci použit Phonexia SID [1]. Ten získává z řeči tzv. embedding, který je diskriminační pro daného mluvčího a používá se pro jejich rozpoznávání či verifikaci. Embedding extraktor, který je použit v této práci, je vystavěn na x-vector architektuře [2]. Avšak metodika navrhnutá v této práci se dá využít i v systémech založených na jiné architektuře.

Pro evaluaci systému na neznámých datech je potřeba prvně vytvořit pro každý embedding, který reprezentuje jednu audio nahrávku, nějakého pseudo-mluvčího. Pro tento proces jsme využili shlukovací algoritmy, které uskupí data do shluků. Každý shluk reprezentuje jiného pseudo-mluvčího pro každý embedding, který je součástí tohoto shluku. Takto vytvořená anotace se pak použijí pro výpočet metriky EER, která je standardní mírou používanou v biometrických systémech. Ale předtím než se vypočítá EER, tak je potřeba odhadnout, kolik shluků v datech skutečně je a ověřit, jestli dané shlukovací algoritmy uspořádali data adekvátně, tak aby reprezentovali skutečné uspořádání dat. Pro to využíváme sérii čtyř různých metrik a jejich závislosti na počtu shluků. V grafu vykresleném pomocí těchto metrik se objevuje tzv. loket podle kterého se dá odhadnout optimální počet shluků. Hledání tohoto bodu může být poměrně složité a nerozhodné, proto jsme využili algoritmus Kneedle [3], který tuto činnost automatizuje. S takto odhadnutými shluky je poté možné vypočítat metriku EER a porovnat ji s referenční, kde dopředu víme, od koho který embedding pochází.

Pro shlukovací analýzu jsme využili tyto algoritmy: K-means, Gaussian mixture models (GMM) a také aglomerativní shlukování (AHC) s úplnou a průměrovací spojovací funkcí. Bylo zjištěno, že se nejlépe chová K-means, protože shluky vytvořené tímto algoritmem nejvíce odpovídají realitě. AHC s úplnou spojovací funkcí trpí fenoménem nazvaným *řetězení*, kde se objeví jeden velký shluk, který začne pohlcovat další. Naopak AHC s průměrovací spojovací funkcí tvořil velké množství shluků, které měli jen jednoho člena, a to u dat, která podle reference žádný takový shluk neobsahovaly.

Jakožto evaluační metriky shlukovací analýzi byla zvolena čtveřice; EER, Silhouette hodnota, Calinski-Harabasz (CH), a Davies-Bouldin (DB) index. Nejlepší podle experimentů vychází Silhouette hodnota, kde bylo vidět zřetelné maximum, které bylo zvoleno jako loket. Podle EER se pokaždé určil vyšší počet shluků, tato metrika by potřebovala optimalizovat pro daný problém. CH a DB index u některých datasetů ukázali zřetelné lokální minimum nebo maximum v místě reálného počtu mluvčích, ale u jiných ne, proto je dobré je využívat jen v kombinaci s předešlými metrikami.

Vývoj evaluační metody nejprve extenzivně probíhal na datasetu SITW, kde jakožto nejlepší metrika byl algoritmus K-means se Silhouette hodnotou, ta se pak použila i pro testování na dalších datastech. Takto odhadnuté EER bylo $5,72$ % s dev-core-core testovací podmnožinou, což je poměrně dobrý výsledek vzhledem k tomu, že EER vypočítané se

znalostí mluvčích v nahrávkách bylo $5,16$ %. Na testovací podmnožině SITW eval-core-core vyšlo EER = $5,86$ % a referenční EER = $5,08$ %.

Další použité datasety byly NIST SRE16 a VoxCeleb1, na kterých bylo experimentálně odhadnuto EER s hodnotami $2,40$ % a $3,28$ %, respektive. Referenční hodnoty EER vyšli $2,81$ % a $3,86$%. Vzhledem k tomu, že se jedná o odhad metodou strojového učení bez učitele, tak jsou to velmi dobré výsledky. Obecně se dá říct, že se odhad EER od reálné hodnoty průměrně lišil o ± 1 %.

Pro další zlepšení výsledků je možné použít jiné metriky, nebo shlukovací algoritmy jako je X-means [4] a G-means [5], které sami odhadují počet shluků. Další experimenty by měli také prozkoumat, jaký vliv hraje na evaluaci systému na neznámých datech poměr šumu k řeči (SNR), nebo celková délka řeči v audio nahrávce.

Unsupervised Evaluation of Speaker Recognition System

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Pavel Matějka, Ph.D. The SID system and other supplementary information were provided by Ing. Michal Klčo from Phonexia. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

....
Ondřej Odehnal
May 11, 2022

Acknowledgements

I would like to sincerely thank my supervisor Ing. Pavel Matějka, Ph.D. for his support, advice, and helpful remarks while working on this thesis. In addition, I would like to thank Phonexia and Ing. Michal Klčo who provided me with valuable advices, audio datasets and the Phonexia SID system which was used in this thesis.

Contents

Nomenclature

Linear Algebra

$\mathbf{A}$	Matrix is written in bold capital letters
A_{ij}	j-th element of the i-th matrix
x	Vectors or observations are written in small letters
$\|\|x\|\|$	Vector length
$\hat{x}$	ℓ^2-Normalized vector $\hat{x} = \frac{x}{\|\|x\|\|}$
x_{ij}	j-th component or feature of the i-th vector
X	Vector features are denoted with capital letters

Abbreviations

AI	Artificial Inteligence
ASR	Automatic Speech Recognition
DNN	Deep Neural Network
EER	Equal Error Rate
GMM	Gaussian Mixture Model
LDA	Linear Discriminant Analysis
MFCC	Mel Frequency Cepstral Coefficient
ML	Machine Learning
PCA	Principal Component Analysis
PLDA	Probabilistic Linear Discriminant Analysis
SID	Speaker Identification
SITW	Speakers in the Wild
SRE	Speaker Recognition
VP	Voice Print

Chapter 1

Introduction

With the arrival of modern communication technologies, there has been a shift in the way humans interact. In some areas, in-person communication has been replaced by calls or other voice technologies. This trend has developed because this type of interaction is fast, inexpensive, and convenient. As such, banks, insurance companies, and retail finance quickly adopted them to communicate with their broad clientele. However, these technologies have weaknesses that criminals have exploited. Several layers of security have been added, such as various identification details or passwords, to prevent fraud and scams. The heightened security results in additional operational costs and time to authenticate clients, yet there is a unique medium suitable for authentication which we can easily share and that is difficult to fabricate or mimic – our own speech. In recent years, the so-called *voice biometry* or *speaker recognition* (SRE) systems have been successfully deployed to help companies verify their clients over the phone. The state-of-the-art speaker recognition technology can recognize speakers based on their voice in just three seconds. The speaker verification process can take place without clients even noticing, which leads to a better customer experience and increases account security.

In addition to the seamless voice authentication, the solution helps companies comply with various security regulations and shorten the authentication phase of an average call by 30+ seconds, leading to a significant cost reductions for the contact center [6].

So far, only the use of speaker verification for security purposes has been described. However, this technology can be used for different purposes, e.g. identifying criminals over a wide range of recordings, conversational AI platforms that can easily identify the person who is speaking, or other interactive voice response systems which can provide a more personalized experience.

Modern speaker recognition systems are language independent. The uniqueness of human voice characteristics makes it possible that even though someone tries to speak in a different language or accent, they are recognized anyway. This also means that the speaker does not need to say a specific sentence or word to be recognized successfully. Moreover, this technology is channel independent and works with various sources, such as phone calls, YouTube videos, eavesdropping recordings, and other types of audio media [7].

The performance of the speaker recognition system is evaluated on labeled data, i.e. we know who is speaking in which recording. Although, when the system is used in the production environment, it is not always possible to obtain a labeled dataset from the customer. This makes it impossible to evaluate it under the specific conditions of this work environment. Creating such a testing dataset is *time-consuming* and *financially costly.* Nevertheless, it is often the case that the customer has a lot of unlabeled data collected throughout their operations. It would be beneficial to analyze the collected data to evaluate the system on a *customer-specific dataset*.

Therefore, the aim of this thesis is to carry out experiments and discover possible ways to evaluate SRE systems on unlabeled, or, in other words, *unsepervised data*. For that, the speaker recognition system is introduced in Sections 2.1–2.6 followed by Section 2.8 dealing with clustering which is a machine learning technique, used to create pseudo-labels for unsupervised system evaluation. The evaluation of clustering algorithms and these pseudo-labels is explained in the next section 2.9, most notably *equal error rate* (EER), which is a standard measure used in biometric systems and also a main measure used in this thesis. Chapter 3 briefly introduces the datasets used and their testing conditions. Chapter 4 describes the experiments. It contains Section 4.1 on experimental setup, unsupervised technique development on a part of SITW dataset in Section 4.2 and evaluation on other dataset in Section 4.3. The results are summarized and discussed in the last Chapter 5 with reference to future work. This approach was developed and tested on Phonexia SID system [1], nonetheless, it is meant to be system-invariant.

Chapter 2

Theoretical Introduction

2.1 Speaker Recognition

Speaker recognition (SRE) is a process of recognizing a person based on their voice recording. Similarly to fingerprints, it is nearly impossible to have two people with the same voice. This uniqueness arises from both the physical and the learned parts. Every individual has a different physical structure of their speech organs, which functions as voice modulators – larynx – or signal filters – pharynx, tongue, nasal cavity, and other parts of voice production. In addition, every speaker has a distinct speaking habit, such as accent, pronunciation, intonation, and speech rhythm [8]. SRE applications can be divided into four main categories [1]:

- **Speaker Verification** – which can be used in banks to verify that the customer who calls the bank is the person he claims to be. This verification approach is also used in Voice-as-a-Password systems, adding additional layer of security to multifactor authentication over the telephone.
- **Speaker Identification** – where we want to know whose voice is this from a pool of people. A great example is an entrance access to the building.
- **Speaker Search** – a case where we are looking for the occurrence of a specific speaker in a large number of audio recordings.
- **Diarization** – technique for partitioning audio or video recordings to classes corresponding with speaker identity or, in short, a task of deciding who is speaking when.

The application of this system can be categorized by knowledge of the spoken text into; text-dependent or text-independent. This thesis considers only *text-independent recognition.*

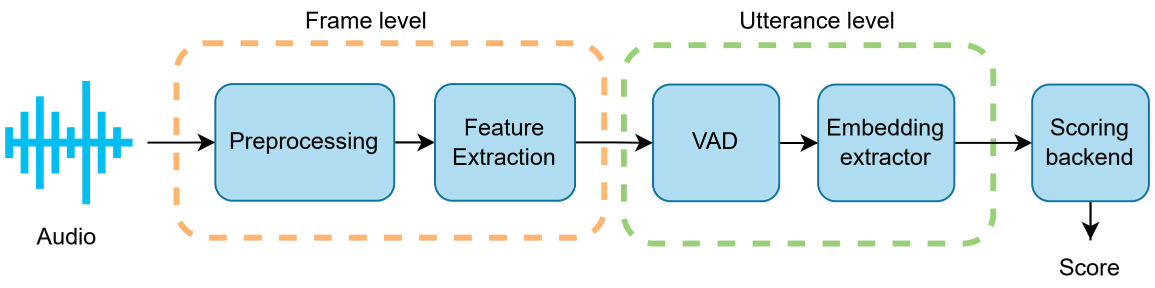


Figure 2.1: *Overview of the processing pipeline of the SRE system.*

The whole process of SRE starts with sampling the raw audio. This audio is used as input to the SRE processing pipeline shown in Figure 2.1, which works on two different levels. The frame level produces features (FBANK, MFCC) processed and aggregated on an utterance level. After all the audio is processed, the final product of this stage is single embedding. It is a vector used to represent the specific recording and speaker. These embeddings are used to compare two recordings, yielding a score that is a single positive or negative number that represents the likelihood that the same speaker will be present in these two recordings. The higher the score, the higher the likelihood.

The processing pipeline in Figure 2.1 can be divided into two parts. The first part, commonly referred to as the *frontend*, starts by processing the audio and ends with embedding from the embedding extractor. The second part, the scoring *backend*, consists of processing the embedding and obtaining a score.

2.2 Speech Feature Extraction

The extracted speech features are ideally discriminative for each speaker and robust to noise in the environment and other distortions, and are relatively simple to measure from an acoustic signal. The acoustic signal of a speech is information-rich. However, it contains various information that has little to no relevance to speaker recognition. Furthermore, we have a different sense of distance between frequencies. Physiological and psychological studies of human speech and auditory systems were used to develop the extraction of speech characteristics [9]. Feature extraction typically starts with applying a *pre-emphasis filter*. This filter amplifies higher frequencies. The signal is then cut into small segments (frames) by windowing. The bell-shaped *Hamming-window* function is commonly used for this process, usually with a window length of 20–25 ms and a time change of 10 ms. This process is illustrated in Figure 2.2. Such a window length is enough to consider the segment spectrum relatively stationary, and the time shift enables us to capture the dynamics among frames.

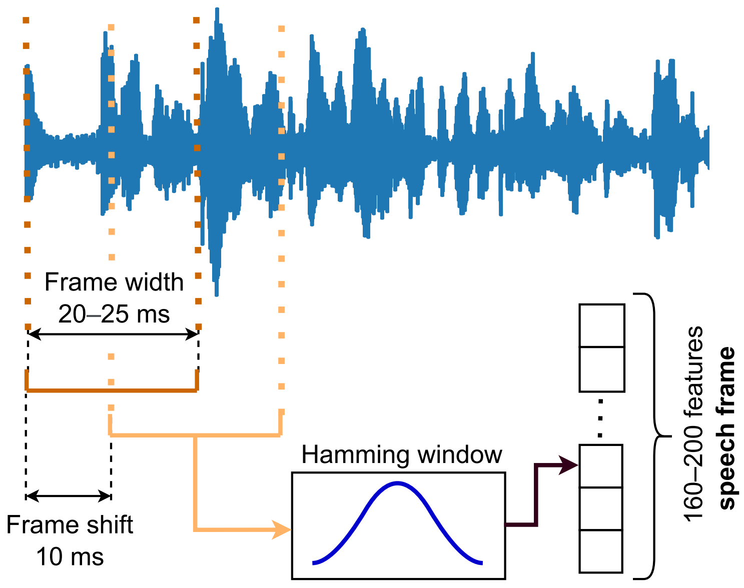


Figure 2.2: *Windowing of an audio with 20–25 ms frame width and 10 ms time shift. Each frame is processed by the Hamming window obtaining speech frames with 160–200 features depending on the frame width.*

The final features should be independent of each other, making it easier to develop machine learning (ML) models. Furthermore, the number of features should be relatively low due to the curse of dimensionality phenomena (see Section 2.8.1), as the number of training samples required increases exponentially with the number of features. Such features are explained in the following subsections. Another distinct approach explained in [10] is to use the raw waveform to give more freedom to neural networks, potentially allowing better capture of important narrow-band speaker characteristics such as pitch and formants.

2.2.1 MFCC – Mel Frequency Cepstral Coefficient

Mel Frequency Cepstral Coefficients (MFCCs) [11] are commonly used as features in automatic speaker recognition and speech recognition tasks. The extraction pipeline for a single MFCC vector is shown in Figure 2.3. It starts with applying a short-term Discrete Fourier Transform (DFT) to obtain the magnitude spectrum. This is followed by the use of the Mel filterbank, which is a set of triangular functions based on the Mel scale [12]. This is a perceptual scale of pitches, judged by listeners to be equal in distance from each other, and is related to the Mel frequency which has a relation to the original frequency f as $f_{Mel}(f) = 2595 \log\left(1 + \frac{f}{700}\right)$. The logarithm of the Mel spectrum is then reduced as well as decorrelated using the Discrete Cosine Transform (DCT) to an optimal number of coefficients called MFCC. The zeroth coefficient, which represents the average log-energy of the input signal, is often excluded because it only carries little speaker-discriminative information. Traditional systems for speech recognition use 8–13 coefficients, and for speaker recognition 12–20.

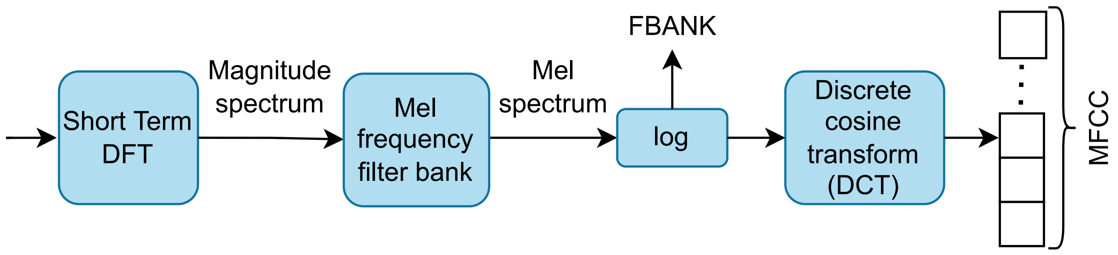


Figure 2.3: *Extraction of MFCC and FBANK from a single speech frame*

To further enhance the information collected from this process, *derivatives* are added to the static MFCC single speech frame to provide *dynamic information* on the power spectrum. The first-order derivative cepstrum can be expressed as

$$\Delta_i = \frac{\sum_{\tau=1}^{T} \tau(c_{i+\tau} - c_{i-\tau})}{2\sum_{\tau=1}^{T} \tau^2} \tag{2.1}$$

where T is the size of the width of the context, τ is the delay in the frame, and c_i is the ith frame of the vector of the basic cepstrum coefficient. This derivative is called the delta cepstrum Δ_i. The second-order derivative $\Delta\Delta_i$ (acceleration) is obtained similarly to (2.1), with the difference of using Δ_i instead of c_i. Both derivatives are used in modern SRE systems, increasing the number of single-frame MFCCs threefold. These coefficients can then be used as input in ML algorithms, such as deep neural networks (DNNs). Nevertheless, deltas are not used in some DNN architectures, notably time-delay neural networks (TDNNs), which use time delays to capture temporal patterns in speech signals.

2.2.2 FBANK Features

Another approach is to use the FBANK features, which are calculated similarly to the MFCC coefficients. They are extracted as logarithmic Mel-filter bank channel output, as shown in Figure 2.3. Typically 20-80 of these coefficients are used in modern SRE systems. Since FBANK features do not use deltas, they are generally used in DNNs that are designed to capture dynamic patterns, such as TDNNs.

2.3 VAD – Voice Activity Detection

Voice Activity Detection (VAD) is an important preprocessing stage of many systems that work with speech. It detects the presence or absence of human speech in frames from the analyzed utterance, and thus filters unwanted sections which provide little to no information about speaker or speech. This is beneficial for two reasons. First, a computational time and resources saving – the frames without speech are not further porcessed. Second, it helps to start and stop voice-triggered services. In SRE there are several approaches to VAD; they can be summarized as follows:

1. Extract features from noisy signal. (there might be a noise reduction stage)
2. Apply *classification rule* to determine if the section contains *speech* or *non-speech*.

The classification rules for VAD are various ranging from hidden Markow model (HMM), Gaussian mixture model (GMM) to neural networks (NN). The modern approach is to use the NN that was trained for this task from selected features on a large dataset. Previous methods typically used signal energy thresholding. VAD is aimed at being resource-leightweight, since its mostly used as a fast preprocessing step.

2.4 DNN – Deep Neural Network

The deep neural network (DNN) is an artificial neural network composed of several layers, sometimes considered a stacked neural network, and is used to obtain the x-vectors explained in more detail in the following Section 2.5. In recent years, it has been progressively gaining popularity as an alternative to i-vectors[1] [13] for speaker recognition, and x-vectors have de facto become a new standard in SRE. As mentioned above, the neural network consists of several layers and each layer consists of the same components: neurons, synapses, weights, biases, and activation functions. These components can be viewed as simple functions that together in DNN can model *complex non-linear relationships*. Extra layers of such components lead to the composition of lower-layer features, potentially modeling complex data. There is a wide range of DNN architectures; each of them is successful in specific domains. Some of the notable ones are introduced below:

- **Feed-Forward DNN** – the flow of information is *unidirectional* without looping back [14].
- **Recurrent Neural Networks (RNN)** – the data can *flow in any direction*, introduce loops in networks. They are quite effective in language modeling [15].
- **Convolutional Neural Network (CNN)** – *convolves* the input of previous layers with convolution kernels or filters that slide along the input features. They are used in automatic speech recognition (ASR) [16], SRE systems, acoustic modeling, or computer vision.
- **Residual Neural Network (ResNet)** – introduces *residual block* that utilizes a technique called *skip connections* that skips several layers of connections (typically up to three layers) and connects directly to the residual block output. ResNet is made up of several of these blocks stacked together. This type of DNN is popular in ASR, SRE systems [17], and computer vision.
- **Time-Delay Neural Network (TDNN)** – in TDDN the output of nodes from a *different time step* can be used as input for forward nodes. This type of network found a number of successful applications in speech, because in its nature it can capture dynamic information. TDDNs are nicely explained in this blog post [18].

These architectures can be combined to provide better efficiency, such as the residual convolutional neural networks used in SRE [19].

The topic of DNNs is a wide research area and contains much more than what is described in this section, which provided a brief overview of this topic, a detailed explanation is out of the scope of this thesis.

[1]In comparison to x-vectors, i-vectors are based on GMM, training is unsupervised and are more versetile. An i-vector extractor system can be used in a broader variety of applications. An x-vector extractor is trained supervised toward a specific application.

2.5 x-vector Extraction

One approach to extract discriminatory speech embeddings from speech features using DNN are the x-vectors introduced in [2] based on previous work [20]. As shown in Figure 2.4, the DNN is divided into a *frame level* and an *utterence level.* The network typically consists of several hidden layers followed by a *statistical pooling* layer. This layer is used to aggregate the output of previous layers over time, compute their average and standard deviation, and forward this pooled information to the next layer. The statistics aggregated at the frame

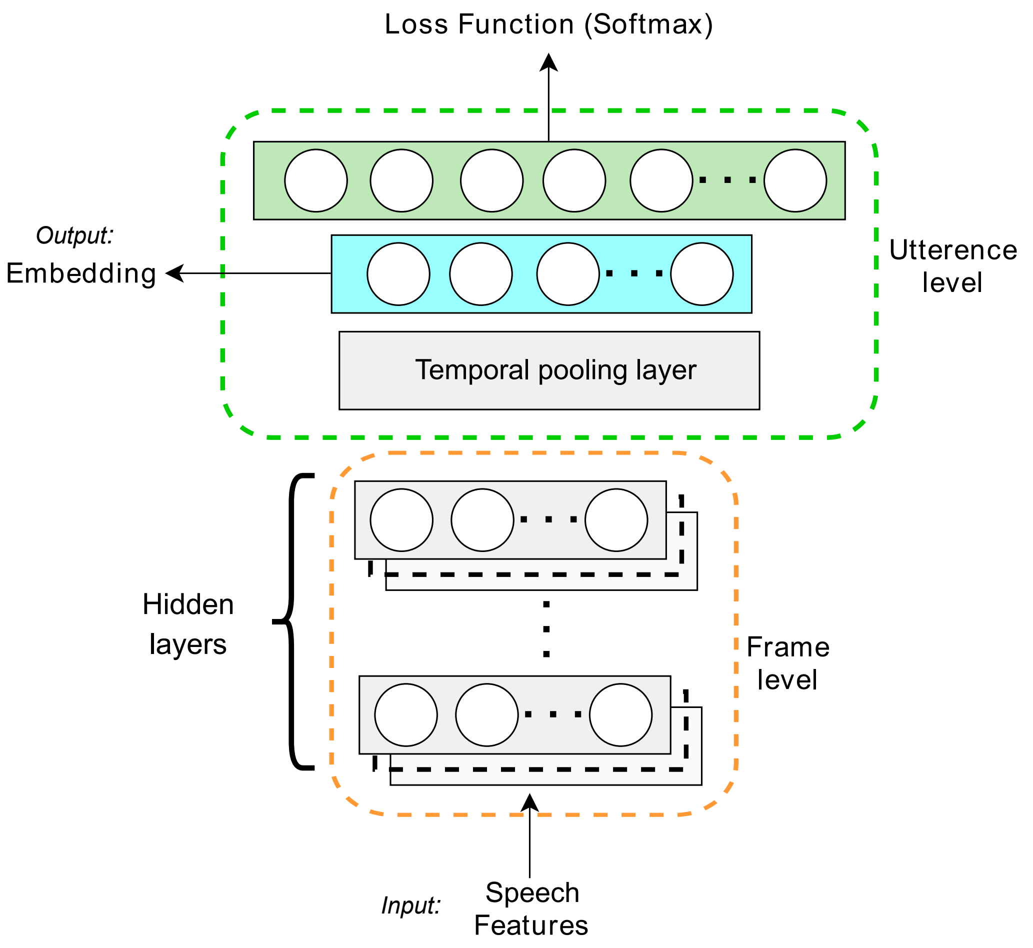


Figure 2.4: *DNN architecture with several hidden layers followed by temporal pooling layer which aggregates their output and passes it to final hidden layer. Linear output layer produces the speaker embedding. Last layer is used only during training.*

level are then propagated at an utterance level to a final hidden layer, succeeded by a linear layer that produces final speaker embedding – *x-vector.* The last layer is used for training purposes only and is discarded afterward.

The network is commonly trained with the softmax function, such as the ensemble additive margin softmax proposed in [21]. The objective function works to *maximizes the between-speaker* and *minimizes the within-speaker variances*, so the final embeddings are in the best case close together if they are extracted from the same speaker and, as far as possible, if from a different speaker. Backpropagation is performed with a parallelized stochastic gradient descent and angular margin loss. See [20] for a more detailed explanation. Modern x-vector extractors use the ResNet-based deep neural network architecture [22].

Robust x-vector extractors are trained using a wide range of datasets (private or public) consisting of different speaker groups such as age, sex, language, etc. To achieve better performance, artificial noise, music, or revarberation are added to the training data.

2.6 Scoring Backend

Scoring backend models provide a way to compare embeddings and get results. It is usually based either on cosine similarity explained in Section 2.8.3 or on a modification of PLDA described in Section 2.6.3. Before applying these algorithms, the embeddings are typically pre-processed with mean normalization, ℓ^2-normalization, and feature dimensionality reduction. Generally, for a smaller number of features (less than or equal to 256), LDA is used to reduce their dimension. However, it does not work well with too many features; this is the result of *curse of dimensionality phenomena* introduced in Section 2.8.1. To avoid this, PCA explained in Section 2.8.2 usually precedes LDA and reduces dimension to a suitable number for LDA, for example, from 1028 to 256. Algorithm 1 summarizes the version of the scoring backend used in the experiments.

Algorithm 1 Scoring backend algorithm – vpcompare SID model XL4

```
1: Mean normalization – subtract mean vector, calculated on a training dataset, from
   embeddings.
2: Apply $\ell^2$ normalization on 256 feature embeddings.
3: LDA dimensionality reduction from 256 features to 128.
4: Mean Normalization of 128 feature embeddings. Mean vector was calculated on a train-
   ing dataset.
5: Repeat step 2 on 128 feature embedding.
6: Calculate the cosine similarity. (or GPLDA)
7: Shift and scale to obtain the score.
```

The result of the scoring backend is a single number – *score*. It tells us how likely two embeddings are coming from the same speaker. The decision threshold is adjusted to meet the requirements for the use of SID. For example, it is lower for criminalists searching through numerous voice recordings to find a criminal, and it is higher for banks and other risk-sensitive applications. Score calibration is performed to obtain the shift and scale in step 7 of Algorithm 1, typically with logistic regression.

This section describes the score normalization in Section 2.6.1 the theoretical background of a dimensionality reduction technique LDA in Section 2.6.2 and a generative probabilistic model PLDA in Section 2.6.3. Cosine similarity is introduced in Section 2.8.3 in the context of K-means in the section on clustering and unsupervised learning, where it is more fitting.

2.6.1 Score Normalization

Score normalization can further *increase the accuracy* of SRE systems. This section mentions it just briefly, since it is not used in the version of Phonexia SID which was used in this thesis for the development and testing of the unsupervised speaker evaluation technique. The normalization process generally aims to reduce the variability of the scores, making it impossible to achieve a speaker-independent threshold during the decision phase. These variabilities can be caused by different recording conditions, such as channel, language, or acoustic conditions. This is reduced by normalization, which generally consists of *shifting* and *scaling* score distributions s for individual models and conditions as follows:

$$s_{norm}(s) = \frac{s - \mu}{\sigma} \tag{2.2}$$

Here, the mean μ and standard deviation σ are estimated using a *normalization cohort* – set of utterences of non-matching speakers. There are several types of normalization using different cohorts, such as *Z-norm, T-norm, S-norm* and others.

2.6.2 LDA – Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) [23] is a supervised technique whose objective is to identify a linear combination of features that:

- *maximize* the between-class separation of data (between-class variance)
- while *minimizing* the within-class scatter (within-class variance)

The resulting combination can be used for *classification* or, more commonly, for *dimensionality reduction* before a subsequent classification. LDA is closely related to the PCA described in Section 2.8.2. Both of them search for linear combinations of variables that best describe the data. However, the PCA algorithm identifies the subspace that concentrates most of the data's energy – variance.

Two distinct approaches can derive LDA. The first is generally referred to as *Fisher's linear discriminant* and is suitable for dimensionality reduction, although this term and LDA are often used interchangeably. The second approach is probabilistic based on *Bayes' theorem* and is better suited to explain LDA in terms of classification. They both make the same assumption about the data and their scatter/covariances are shown in Figure 2.5. In order to describe them, let us introduce some common ground terminology. Let $\{x_1, x_2, \ldots x_n\}$ denote a training dataset of N examples, where each is a column vector of length d. Each example belongs to exactly one of the K classes defined as $C_1, \ldots C_K$. The number of examples in class $k = 1 \ldots K$ is then $n_k = |C_k|$.

Multiclass Fisher's Linear Discriminant

The main idea is to find a linear projection that maximizes the ratio of the between-class scatter and within-class scatter; this ratio is called Fisher's criterion [24]:

$$J(\mathbf{W}) = \frac{\mathbf{W}^T \mathbf{S}_b \mathbf{W}}{\mathbf{W}^T \mathbf{S}_w \mathbf{W}} \tag{2.3}$$

$\mathbf{W}$ is a $d \times d'$ transformation matrix where d' is the desired number of dimensions. The between-class scatter $\mathbf{S}_b$ is defined as the sample covariance of the class means and within-class scatter $\mathbf{S}_w$ as covariance of samples accross all clusters. The matrices $\mathbf{S}_b$ and $\mathbf{S}_w$ are computed as follows:

$$\mathbf{S}_w = \frac{\sum_k \sum_{i \in C_k (x_i - m_k)(x_i - m_k)^T}}{N}, \ \mathbf{S}_b = \frac{\sum_k n_k (m_k - m)(m_k - m)^T}{N} \tag{2.4}$$

where $m_k = \frac{1}{n_k}\sum_{i\in C_k} x_i$ is the mean of the kth class and $m = \frac{1}{N}\sum_i x_i$ is the mean of the entire dataset. Our aim is to find the linear transformation $x \to \mathbf{W}^T x$ that satisfies the LDA objective – maximize $\mathbf{S}_b$ relative to $\mathbf{S}_w$, in other words, maximize Fisher's criterion (2.3). It can be shown that the columns w of the optimal $\mathbf{W}$ are generalized eigenvectors such that $\mathbf{S}_b w = \lambda \mathbf{S}_w w$ corresponds to the d' highest eigenvalues. For a more detailed description of this approach, see [25].

The LDA projection can be derived by fitting GMM to the training data. The results of the mixture model can be used to classify examples of the classes represented in the training data, but not the *novel classes*. A different probability model is required for this purpose and is provided by Probabilistic LDA described in Section 2.6.3.

LDA with Bayes' Theorem

What we want for classification is the probability that x comes from the cluster k or simply $P(Y=k|X=x)$. To obtain this, let us define $f_k(X) \equiv P(X=x|Y=k)$ as *density function* of X and *prior* $\pi_k \equiv P(Y=k)$ where π_k represents the probability that a randomly chosen observation comes from the kth class. Using the Bayes' rule, we get the following:

$$P(Y=k|X=x) = \frac{P(X=x|Y=k)}{\sum_i P(X=x|Y=i)P(Y=i)} = \frac{f_k(x)\pi_k}{\sum_i f_i(x)\pi_i} \tag{2.5}$$

The objective is to classify x into the cluster group k which has the highest probability. In (2.5) the denominator is constant for different clusters. This means that the probability is proportional to the enumerator:

$$P(Y=k|X=x) \sim f_k(x)\pi_k \tag{2.6}$$

We assume that the d-dimensional random variable X has a multivariate Gaussian distribution $X \sim \mathcal{N}(\mu, \mathbf{\Sigma}_w)$ where μ is the mean of X (vector with d components), $\mathbf{\Sigma}_w$ is the covariance matrix $Cov(X) = \mathbf{\Sigma}_w$ of X. The multivariate Gaussian density is defined as:

$$f(x) = \mathcal{N}(x|\mu, \mathbf{\Sigma_w}) = \frac{1}{\sqrt{(2\pi)^d|\mathbf{\Sigma_w}|}} \exp\left(-\frac{1}{2}(x-\mu)^T{\mathbf{\Sigma_w}}^{-1}(x-\mu)\right) \tag{2.7}$$

By comparing two classes k, l probabilities log-ratio

$$\log\frac{P(Y=k|X=x)}{P(Y=l|l|X=x)} \tag{2.8}$$

from the (2.8) we can derive to *linear discriminant function*:

$$\delta_k(x) = x^T\mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}^{-1}\mu_k + \log\pi_k \tag{2.9}$$

Where k is the specific class. Σ_k, μ_k, π_k are estimated from the training data. The linear discriminant function (2.9) is used to estimate the *LDA decision boundaries* of the data. Detailed explanation with derivations can be found in [23] and in [26].

2.6.3 PLDA – Probabilistic Linear Discriminant Analysis

Probabilistic LDA [27] is a generative probabilistic model that can be applied to a wide range of problems, such as classification, hypothesis testing, class inference, and clustering, all on classes not observed during training. On the contrary, LDA can only cluster data to known classes during model training. In the case of SRE systems, PLDA and its derivatives are frequently used [28][29][30] for *hypothesis testing* where they help to answer the question: *„Are those two examples from the same class?“* For that, it computes the likelihood ratio.

PLDA can be thought of as a model in which the vector x represents the observation of the speaker and y is a latent vector representing the class of speaker. The class-conditional distribution

$$P(x|y) = \mathcal{N}(x|y, \mathbf{\Sigma}_w) \tag{2.10}$$

have a common within-class covariance matrix $\mathbf{\Sigma}_w$, which is similar to LDA (2.7). However, in contrast to this, the prior $P(y)$ is made continuous by imposing a *Gaussian prior*

$$P(y) = \mathcal{N}(y|\mu, \mathbf{\Sigma}_b) \tag{2.11}$$

where μ is the mean of the class and $\mathbf{\Sigma}_b$ is a between-class covariance matrix. Note that the covariance matrices $\mathbf{\Sigma}_b$, $\mathbf{\Sigma}_w$ are same for all observations as illustrated in Figure 2.6.3.

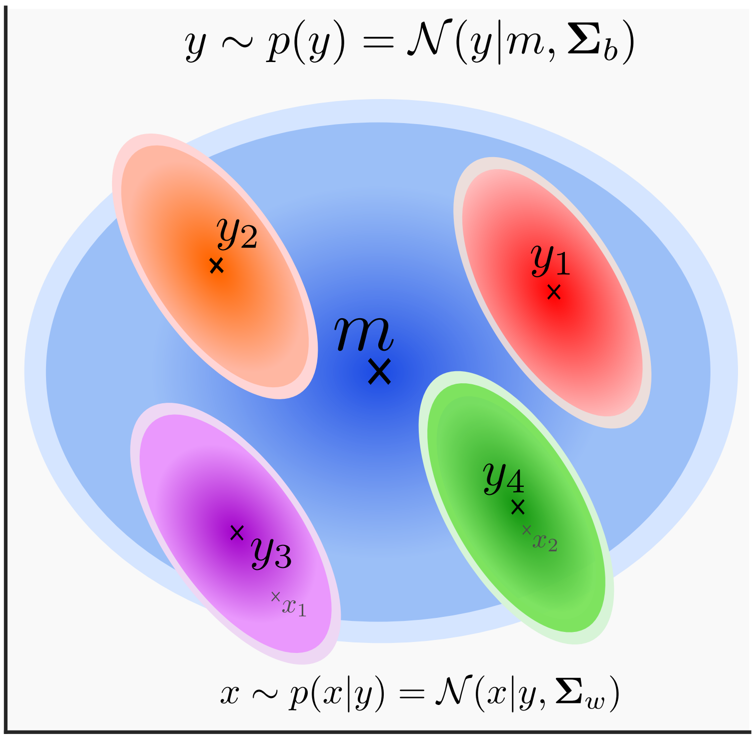

$y \sim p(y) = \mathcal{N}(y|m, \mathbf{\Sigma}_b)$
y_2
y_1
m
y_4
x_2
y_3
x_1
$x \sim p(x|y) = \mathcal{N}(x|y, \mathbf{\Sigma}_w)$

Figure 2.5: *PLDA (and also LDA) assumption about data. Class distribution* $y \sim \mathcal{N}(y|m, \mathbf{\Sigma}_b)$ *with center* m *representing the distribution of classes of speakers with class-conditional distributions* $x \sim \mathcal{N}(x|y, \mathbf{\Sigma}_w)$ *representing the distribution of y-th speaker recordings. Hue of the distributions illustrates the spread of the data.*

Variations of PLDA model use distinct priors or different numbers of latent variables. The two most popular variants are heavy-tailed PLDA (HTPLDA), which uses Student's t-distribution for priors, and Gaussian PLDA (GPLDA), which assumes Gaussian priors, as shown in Equation (2.11).

2.7 Filtering – Automatic Control of Audio Quality

Estimating the quality of speech in an audio recording is an important pre-processing step. It is used to improve the accuracy of SRE systems and increase the speed of data processing by discarding low-quality audio recordings or audio segments. In this section, we introduce several statistics that are used to estimate the quality of the audio. The most important are the length of the speech signal and the signal-to-noise ratio (SNR).

- **Speech signal length** – detected length of speech detected in audio [31]. This statistic is calculated with VAD explained in Section 2.3.
- **Signal-to-noise ratio (SNR)** – ratio of a speech signal to the power of background noise [32].
- **Perceptual Evaluation of Speech Quality (PESQ)** – subjective of an audio sample. Values in range $\langle -0.5, 4.5 \rangle$, the higher the rating, the better the quality of the recording [33].

2.7.1 Speech signal length

Generally, the more speech in the recording, the better decisions the SRE system makes. However, it is shown in [31] that on average as little as 2–10 seconds of active speech can produce results that are close to those of using an average of more than 100 seconds of speech. It is no surprise that the study also demonstrated a steep increase in EER if there was less than 2 seconds of speech in an audio. Production SRE systems typically use at least 3 seconds of speech [1].

2.7.2 SNR – Signal-to-noise ratio

SNR can be expressed as a measurement of decibels (dB), where a signal with more useful information, speech, typically has higher numbers than noise. If the background, for example during a phone call on a busy street, is noisy, then the SRE system will have difficulty recognizing the speaker.

Generally, we want to reduce background noise as much as possible. However, if we filter all noisy recordings during evaluation on unsupervised data, we will get an unrepresentative subset of the whole dataset and the estimated EER value will be too optimistic.

2.7.3 PESQ – Perceptual Evaluation of Speech Quality

PESQ is a family of standards which are used as a subjective quality test of audio recording. It is standardized as Recommendation ITU-T P.862 [34]. It analyzes specific parameters such as variable delays, noise, or time warping. PESQ is typically used in conjunction with other quality estimation techniques to test the quality of VoIP telephones or for codec evaluation. It requires setting a threshold for values between $\langle -0.5, 4.5 \rangle$, which can be dataset specific.

2.8 Clustering – Unsupervised Learning

Unsupervised learning, or „learning without a teacher,“ is a type of machine learning in which the algorithm learns or discovers patterns from data without labels. It is a set of statistical tools designed to provide thoughtful insight into measurements. The main difference between supervised learning and unsupervised learning is that the latter does not aim to predict the data [26].

In the case of this thesis, we can describe unsupervised learning as a setting in which there is a set of x-vector embeddings $x_1, x_2, ..., x_n$ extracted from n recordings. Each embedding has p features. Our aim is to discover the subgroups among the embeddings. These subgroups correspond to *pseudo-labels* or the so-called *pseudo-speakers*. Evaluation of the speaker recognition system is then done using these pseudo-labels.

This chapter describes two particular types of unsupervised learning techniques, their advantages and disadvantages:

- **Principal components analysis** – data pre-processing tool used before applying another supervised or unsupervised learning algorithm.
- **Clustering** – a class of algorithms for discovering subgroups in the data.

2.8.1 Curse of Dimensionality

Before explaining the unsupervised techniques, let us introduce the Curse of *Dimensionality phenomenon*. This phenomenon occurs naturally when analyzing high-dimensional data, and it might cause inaccuracies or overfitting when using machine learning techniques. The root of this problem is that when the number of dimensions increases, the space among observations expands exponentially as shown in, leading to a sparse dataset with huge distances among these observations as shown in Figure 2.6.

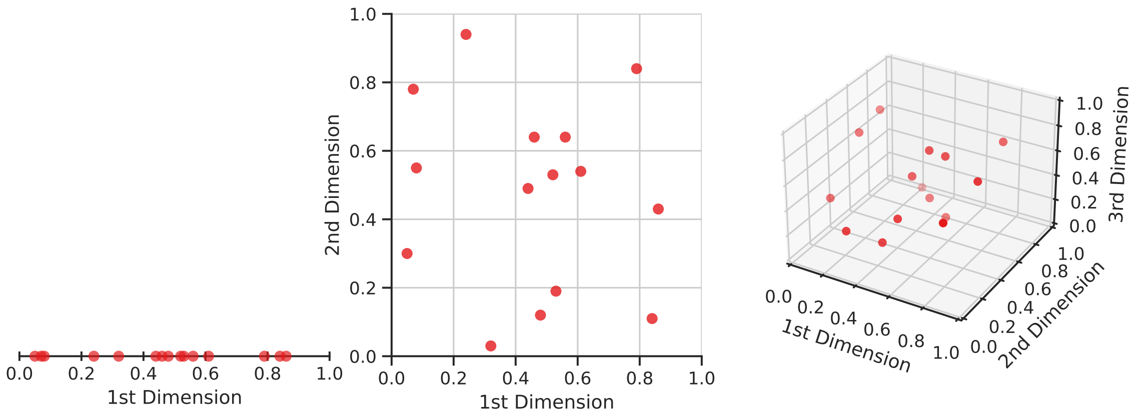


Figure 2.6: *The behaviour of the curse of dimensionality when projected in (1) one dimension, (2) two dimensions, and (3) three dimensions.*

Take an example of the Euclidean distance, in Equation (2.16), between two randomly chosen points. In a unit square, it is roughly 0.52 units. A unit 3D cube has the average distance increased to 0.66 units. However, in a unit 256-dimensional hypercube, the distance grows to 6.5 units.[2]

The curse of dimensionality is related to the Hughes phenomenon [35], which can be described as follows: As the number of dimensions used increases, the performance of a machine learning model increases for the first time. However, it starts to deteriorate with higher dimensions beyond a certain point, rather than gradually improving.

Therefore, with increasing number of dimensions, it becomes cumbersome to calculate the distance between observations, making difficulties for clustering algorithms that rely on a distance metric, such as K-means described in Section 2.8.3. One way to solve this problem is to select features and reduce dimensionality. In the context of processing the x-vectors, feature selection becomes difficult since the features do not coincide with anything tangible. For dimensionality reduction, there are several choices with different objectives, such as; unsupervised t-SNE, PCA explained in Section 2.8.2 or supervised LDA introduced in Section 2.6.2 and PLDA in Section 2.6.3.

The high dimensionality might not always be a curse, but sometimes a blessing, as outlined in a quote from *Blessing of dimensionality* [36]:

> For example, the typical property of a random finite set in a high-dimensional space is: the squared distance of these points to a selected point are, with high probability, close to the average (or median) squared distance. This property drastically simplifies the expected geometry of the data (blessing) but, at the same time, makes the similarity search in high dimensions difficult and even useless. (curse)

To summarize this subsection, the curse and blessing of dimensionality are two sides of the same coin, and there is always a trade-off between using many features or not. Some algorithms, such as clustering, are more sensitive to the curse, yet others are favorable to the blessing and can find patterns throughout many dimensions – notably the neural networks.

2.8.2 PCA – Principal Components Analysis

Principal component analysis (PCA) [23][37] is a multivariate statistical technique that computes the *principal components* (*PC*) and, using them, performs a change in basis on the dataset utilizing a portion of the *PCs*. To do that, PCA uses an unsupervised approach that involves only a set of features and no associated response or labels.

This algorithm is used mainly because it can significantly reduce the size of the whole dataset, remove noise features or those with low information value, and thus make other algorithms, in general, more effective on such a reduced dataset as was previously mentioned with LDA in Section 2.6. The objective of PCA can be summarized as follows [23]:

(a) Extract essential information from the data table in form of *principal components*.

(b) Compress the size of the dataset by keeping only this essential information by the *change of basis*.

(c) Simplify the description of the dataset.

(d) Analyze the structure of the observations.

[2] Average distance between two randomly selected points is $d = \sqrt{\frac{n}{6}}$, where n is the magnitude of a dimension.

Finding the Principal Components

Using the example of observation in a two-dimensional space as shown in Figure 2.7, the PCA begins with the extraction of essential information. It is done by finding the p *principal components* (PC) of the p-dimensional space.

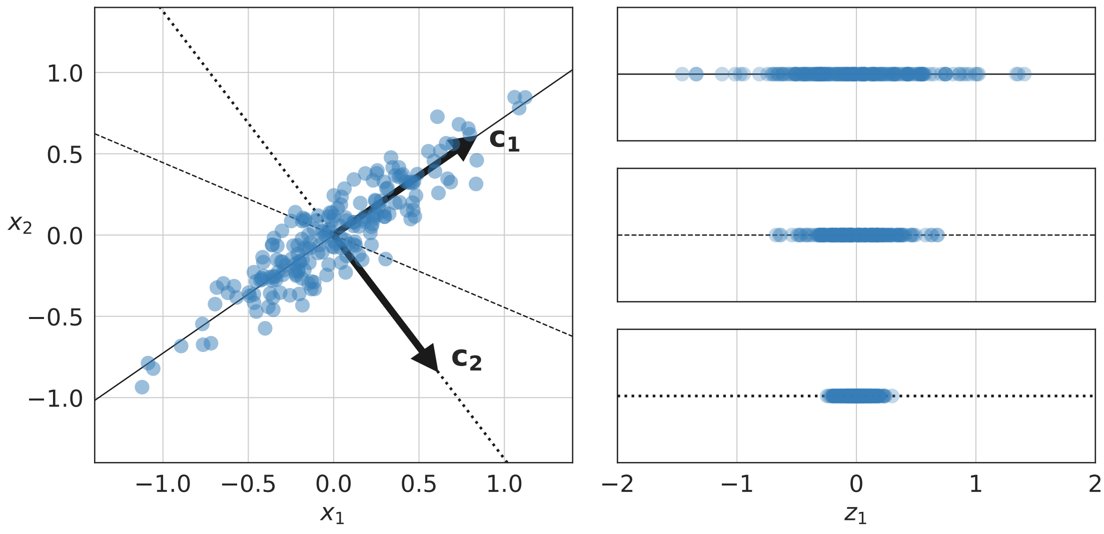


Figure 2.7: *PCA – selecting the subspace to project on the observations from two features space X_2, X_1, on the left, to the single feature space Z_1, on the right, where the solid, dashed, and dotted lines represent projection axes. The singular vectors c_1 and c_2 are the eigen-vectors for PC_1 and PC_2. Projection onto PC_1 preserve most variance of the data.*

The first component PC_1 must have the largest variance, as shown in the projection to one dimension in Figure 2.7 (upper left figure). The second component PC_2 is calculated as orthogonal to PC_1 and is also required to have the largest possible variance. This process would repeat for the p dimensional space until PC_p. Mathematically, *first principal component* PC_1 of the p-dimensional space with features $X_1, X_2, ..., X_p$ is a linear combination of the features.

$$Z_1 = c_{11}X_1 + c_{21}X_2 + ... + c_{p1}X_p \tag{2.12}$$

with the largest variance where the $c_{11}, c_{21}, ...c_{p1}$ are called *loading scores* of the PC_1 and are ℓ^2-normalized[3]. Together, these scores form the principal component *loading vector* $\hat{c}_1 = (c_{11}, c_{21}, ...c_{p1})^T$.

In other words, the first principal component PC_1 solves the optimization problem of finding the linear combination that has the largest variance across samples $x_1, x_2, ..., x_n$. The samples are centered to have mean zero since we are interested only in variance. Using (2.12), this can be written as

$$\underset{c_{11},...,c_{p1}}{maximize}\left\{\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}c_{j1}x_{ij}\right)\right\} \; subject\ to \; \sum_{j=1}^{p}c_{j1}^2 = 1. \tag{2.13}$$

[3] ℓ^2-normalized vector is $\hat{c}_i = \frac{c_i}{||c_i||}$

The optimization problem in Equation (2.13) is solved using *Singular Value Decomposition* or SVD [37] that can decompose the training set matrix into the matrix multiplication of three matrices $\mathbf{U\Sigma V}^T$, where all *loading vectors* $\hat{c}_p$ are contained in $\mathbf{V}$.

Dimensionality Reduction – Change of Basis

Now that we have defined how to obtain all principal components, the dimensionality of the dataset matrix $\mathbf{X}$ can finally be reduced. A matrix multiplication

$$\mathbf{X}_d = \mathbf{X}\mathbf{W}_d \tag{2.14}$$

projects the dataset down to dimensionality d. $\mathbf{W_d}$ is defined as the matrix that contains the first d columns of $\mathbf{V}$ from SVD.

Explained Variance Ratio

The *explained variance ratio* is useful information about *PCs*. Indicates the proportion of variance in the dataset along each *PC*. If a component has a relatively smaller variance than the others, it is reasonable to assume that it carries little information.

Choosing the Right Number of Dimensions

PCA is a trade-off between having all the information at hand or keeping just the most valuable part – with the most variance. Using the explained variance ratio, it is possible to choose how much variance to preserve. For example, 95%. Therefore, the cumulative sum of *PCs'* variance must be above this threshold, and the lower components will be reduced to the lowest dimension possible given the condition of total variance 95%.

PCA variations

With randomized PCA, the algorithm runs much faster, but finds only an approximation of *PCs*. If the data are larger than what can fit in main memory, then Incremental PCA [38] can be used. It splits data into mini-batches.

2.8.3 K-means

The K-means algorithm is one of the most popular iterative descent clustering methods and is comparatively fast [39]. This subsection covers a more detailed explanation of this algorithm, which is important to explain why and how *cosine similarity* can be used in K-means. To perform this algorithm, the number of clusters K must be specified; then the K-means algorithm assigns each observation x_i to exactly one of the K clusters. First, let us define some notation. Let $C_1, ..., C_K$ denote sets containing the indicies of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup ... \cup C_K = 1, ...n$. In other words, each observation belongs to at least one of the K clusters.

2. $C_k \cup C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster

For every cluster C_k there is a *centroid* c_k that is a mean of points in the C_k given as

$$c_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i. \tag{2.15}$$

The K-means algorithm is intended for a situation in which all variables are of quantitative type and squared *Euclidean distance*

$$d(x_i, x_i') = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2 \tag{2.16}$$

is chosen as *dissimilarity measure*. Where x_i is in our case the x-vector of p *features* and x_{ij} is a *single feature* of this vector.

The amount in which the observations within a cluster differ from each other can be defined as *cluster variance*:

$$W(C_k) = \sum_{i \in C_k} ||x_i - c_k||^2 \tag{2.17}$$

This metric is also called the model's *inertia*[4] and is the mean squared distance between each instance and its cluster's centroid. (also the closest centroid) The *objective function* of the K-means is to solve the optimization problem of

$$\underset{C_1,\ldots,C_K}{minimize} \left\{ \sum_{k=1}^{K} W(C_k) \right\}. \tag{2.18}$$

In other words, divide the observations into K clusters while *minimizing* the total cluster variance (2.17). The complexity of the problem is $\mathcal{O}(k^n)$ and it is difficult to find a precise solution. However, there is a fairly simple yet effective algorithm to find *local optimum* for the K-means optimization problem (2.18). This approach is laid out in Algorithm 2.

Algorithm 2 K-means Clustering [23][Algorithm 14.1]

```
1: Initialize the algorithm with the centroids $c_1, ..., c_K$ chosen from the observations
   $x_i, ..., x_n$ where $n \geq K$. The value $K$ corresponds to both the number of clusters,
   and centroids.
2: ASSIGNMENT STEP: Given a current set of centroids $c_1, ..., c_K$, the dissimilarity measure
   (2.16) is minimized by assigning each observation to the closest cluster centroid.
   That is,
       $C(i) = \underset{1 \leq k \leq K}{argmin} ||x_i - c_k||^2.$      (2.19)
3: UPDATE STEP: For every cluster $C_k$, where $1 \leq k \leq K$, recalculate the cluster centroid
   $c_k$ with (2.15) and thus minimize the total cluster variance (2.18).
4: Steps 2 and 3 are iterated until the assignments do not change.
```

At each step, the cluster variance (2.17) is guaranteed to decrease. With this approach, the algorithm eventually reaches *local optimum* when the assignments do not change anymore. An example of nicely clustered observations with cluster centroids is shown in Figure 2.8.

[4]Inertia used in Scikit-Learn https://scikit-learn.org/stable/modules/clustering.html#k-means

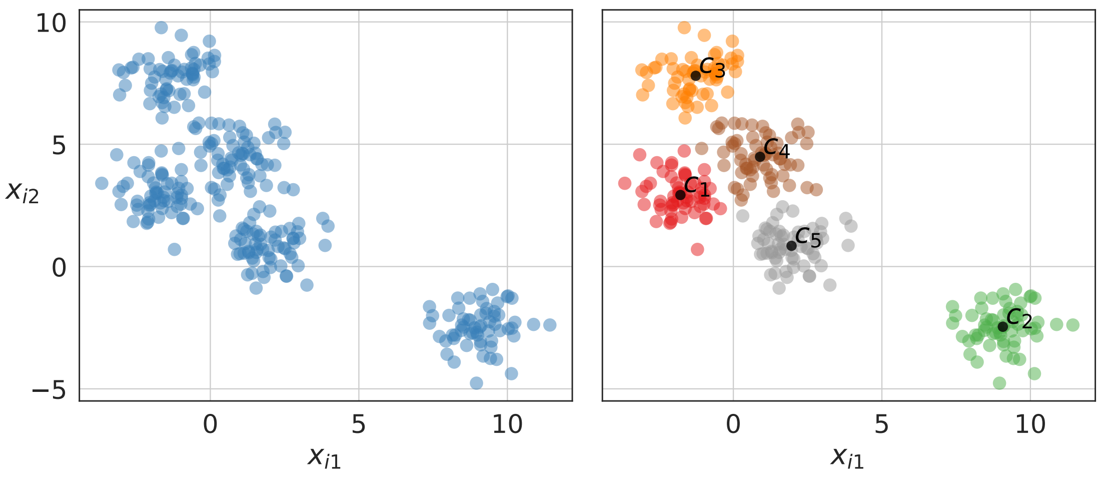


Figure 2.8: *Example of a clustered dataset with K-means where the dataset has 2 features X_1, X_2, 300 observations x_i and the number of clusters K set to 5.*

Optimization of K-means Algorithm

Although the K-means algorithm is guaranteed to converge, it usually converges to *local optimum* most of the time. The final solution depends mainly on *centroid initialization*, which is an essential part of the algorithm and influences the result. The different centroids used for initialization usually give different results. The traditional approach is to run the algorithm several times and select the best solution – the one with the lowest total cluster variance introduced in Equation (2.18). Another option to find a better solution is to use a smart *initialization technique* [40].

Initialization Techniques

There are many K-means initialization techniques. Traditional ones are the Forgy method, which chooses the initial centroids at random. However, the K-means++ [41] method is currently the most widely used. The Scikit-Learn library, which is used for experiments in Chapter 4, uses K-means++ as the default initialization method [42].

Mini Batch K-means

Another important variance to the K-means algorithm is the Mini Batch K-means [43]. The algorithm can use *mini-batches*, instead of the entire dataset in each iteration. Mini-batches are subsets of the input data, randomly sampled in each training iteration. The speed-up is typically by a factor of three or four, and also it makes it possible to cluster huge datasets that do not fit in memory. Although providing a significant acceleration, the total cluster variance explained in Equation (2.18) is generally slightly worse [42].

Cosine Similarity

As mentioned in this section, the K-means algorithm uses the Euclidean distance. This metric is not always helpful, mainly when the magnitude of the observations/vectors does not matter. A typical case is text data represented by word count in documents of uneven length. For this example, the cosine similarity is a more suitable metric and is also preferable in speaker recognition systems.

The cosine similarity is simply the cosine of the angle between two vectors u, v of non-zero length defined as:

$$cos(u, v) = \frac{u \cdot v}{||u|| \cdot ||v||} \tag{2.20}$$

A smaller angle between two vectors results in a higher cosine similarity. Our motive is to use this metric for K-means. Recall that the K-means algorithm is defined for the Euclidean distance, where the objective is to minimize the cluster variance (2.18). It can be proved that the cosine similarity for ℓ^2-normalized vectors[5] is proportional to the Euclidean distance as follows:

$$cos(\hat{u}, \hat{v}) = \frac{\hat{u} \cdot \hat{v}}{||\hat{u}|| \cdot ||\hat{v}||} = \hat{u} \cdot \hat{v} \tag{2.21}$$

From (2.16):

$$||\hat{u} - \hat{v}||^2 = \left(\sqrt{(\hat{u} - \hat{v})^2}\right)^2 = (\hat{u} - \hat{v})^2 = \hat{u}^2 - 2 \cdot \hat{v} \cdot \hat{u} + \hat{v}^2 = 2 - 2 \cdot \hat{u} \cdot \hat{v} \tag{2.22}$$

Substitute the $\hat{u} \cdot \hat{v}$ for ℓ^2-normalized cosine similarity (2.21):

$$||\hat{u} - \hat{v}||^2 = 2 - 2 \cdot cos(\hat{u}, \hat{v}) \tag{2.23}$$

From now on, it is clear that for ℓ^2-normalized vectors, solving the optimization problem of minimizing the sum of cluster variances in all clusters (2.18) also maximizes the sum of cosine similarity between cluster observations x_i and centroid c_k over all clusters. In other words,

$$\underset{C_1,\ldots,C_K}{minimize}\left\{\sum_{k=1}^{K}\sum_{i \in C_k} ||x_i - c_k||^2\right\} \sim \underset{C_1,\ldots,C_K}{maximize}\left\{\sum_{k=1}^{K}\sum_{i \in C_k} cos(x_i, c_k)\right\} \tag{2.24}$$

In summary, the use of the cosine similarity metric in the K-means algorithm is achieved using ℓ^2-normalized vectors / observations and running the algorithm with Euclidean distance. The resulting clusters have minimized the cluster variance and maximized cosine similarity.

2.8.4 GMM – Gaussian Mixture Model

Gaussian mixture model (GMM) is a probabilistic model that makes the assumption that data were generated from a mixture of Gaussian distributions. The goal of this model is to find the parameters of such distributions. The probability density function for multivariate Gaussian density is illustrated in Figure 2.9, and described in Equation (2.7), from which we need to find $\boldsymbol{\Sigma}_k$, μ_k of each class k. This is typically done using the *Expectation-Maximization* soft clustering algorithm, which is in some cases similar to the K-means algorithm described in Section 2.8.3; Clusters also have to be initialized, randomly or ideally with some better heuristics, and it reapeats *expactation* and *maximization* steps until convergence. For each observation, in the expactation step, the algorithm estimates

[5] The ℓ^2 norm is defined for a vector u as $||u| = \sqrt{\sum u_i^2}$. ℓ^2-normalized vector is $\hat{u} = \frac{u}{||u||}$ where $||\hat{u}|| = 1$

the probabilities of observations belonging to each cluster[6], and in the maximization step it updates the groups using all observations in the dataset.

The covariance of the clusters $\mathbf{\Sigma}_k$ of each mixture can be of any ellipsoidal shape, size, and orientation, i.e. has its own general covariance matrix. Such a matrix is usually refered to as *full*. However, this can additionally be adjusted to several variants:

- **Spherical** – all clusters are spherical, but with different diameters. (very similar to how K-means works)
- **Diagonal** – clusters are of ellipsoidal shape of varying sizes. The ellipsoids are parallel to the coordinate axes as in Figure 2.9. (The $\mathbf{\Sigma}_k$ is diagonal)
- **Tied** – clusters have the same shape and size. In other words, they come from the same distribution with different mean (mixture center); this is illustrated in Figure 2.5.

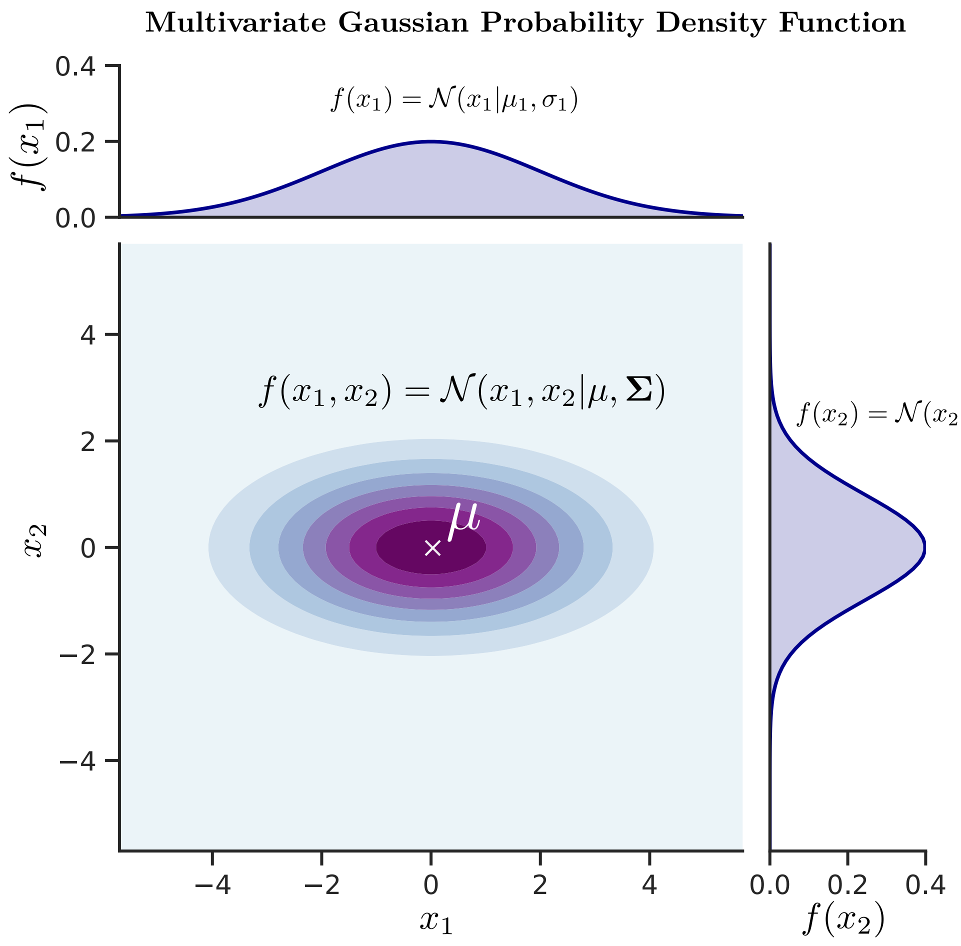


Figure 2.9: *Multivariate (bi-variate) gaussian probability density function with two compounding gaussians. Covariance matrix $\mathbf{\Sigma}$ is diagonal. Darker hue represents more density.*

The selection of the optimal number of clusters for GMM can also be achieved with *Bayesian information criterion* (BIC) or *Akaike information criterion* (AIC) using the minimum of these criteria or the elbow method which is introduced in Section 2.9.2.

[6]In the context of GMM, clusters are sometimes called mixtures

2.8.5 AHC – Agglomerative Hierarchical Clustering

Agglomerative clustering (AHC) is a type of hierarchical clustering that uses a bottom-up approach [23]. It begins with assigning all observations into a singleton cluster, and at each step two clusters are joined according to the *linkage function* producing one less cluster at each step. This process can be visualized in the form of a hierarchical tree called *dendogram*. Clustering can be set to a specific *number of clusters* or a *threshold value*. In connection with SRE, it is popular in speaker diarization and is featured in a tool called Phonexia Orbis Investigator[7].

Linkage Functions

The linkage function can be defined as a measure of dissimilarity $d(C_k, C_{k'})$ between two clusters or groups C_k and $C_{k'}$ with assigned observation x_i, $x_{i'}$, respectively. It is calculated from the set of pairwise dissimilarities $d_{x_i x_{i'}}$ using the Euclidean distance, cosine similarity, or even a precomputed dissimilarity matrix, such as the one created from scores from the scoring backend introduced in Section 2.6. Cluster pairs with the lowest dissimilarity are merged at each step of the AHC algorithm.

Single linkage (SL) considers the pair with a minimal dissimilarity measure

$$d_{SL}(C_k, C_{k'}) = \min_{\substack{x_i \in C_k \\ x_i' \in C_{k'}}} d_{x_i x_{i'}} \tag{2.25}$$

This single linkage criterion (2.25) combines two clusters based only on two observations, regardless of the others in those clusters. It tends to combine observations, at relatively low thresholds, through intermediate observations, leading to a phenomenon called *chaining*, the potential drawback of this method, which also causes not very compact clusters with relatively large diameters.

Complete linkage (CL) is the opposite of single linkage, it takes two clusters and their two *furthest neighbors* i.e. the most dissimilar pair. This measure is given as follows:

$$d_{CL}(C_k, C_{k'}) = \max_{\substack{x_i \in C_k \\ x_i' \in C_{k'}}} d_{x_i x_{i'}} \tag{2.26}$$

CL merges two clusters with minimal d_{CL} (2.26). This results in very *compact clusters*. The negative side of this linkage function is that, as it considers only two observations at a time, it might assign some observations in a way that makes them closer to members of a different cluster than of their own cluster members.

Average linkage (AL) is the average dissimilarity between the groups

$$d_{AL}(C_k, C_{k'}) = \frac{1}{|C_k||C_{k'}|} \sum_{x_i \in C_k} \sum_{x_{i'} \in C_{k'}} d_{x_i x_{i'}} \tag{2.27}$$

The effects of the average linkage function (2.27) are clusters *relatively far apart*, as well as *relatively compact*. However, it is influenced by *scale* of the observations and, as a consequence, scaling could change the result of clustering. The AHC with average linkage is also known as the „Unweighted Pair Group Method with Arithmetic Mean“ (UPGMA).

[7]Orbis, a tool which automatically identifies speakers and other key information in audio https://www.phonexia.com/product/orbis/

2.8.6 Comparison of Clustering Algorithms

This section shows a concise comparison of the clustering algorithms described above in Table 2.1. In addition to this information, it is also important to note that only AHC can use a *precomputed distance matrix*[8], which can significantly accelerate the algorithm; additionally, due to its nature, it is possible to run the algorithm just once with the stopping condition set to a single cluster $K = 1$ and use previous steps to calculate various metrics, since the algorithm starts with all features in a separate cluster and merges them one by one. In contrast, K-means and GMM have to be initiated every time for different numbers of clusters K.

Comparison of Clustering Algorithms

Algorithm	**K-means**	**GMM**	**AHC**
Data assumption	Spherical clusters of even size	Gaussian, ellipsoidal shapes	None
Scalability	Very large number of samples, medium number of clusters[9]	Not scalable	Large number of samples, large number of clusters
Noisy data resistance	Sensitive to outliers	Robust to outliers	Sensitive to outliers
Distance metric	Euclidean[10]	Mahalanobis	Any pairwise distance, very flexible
Deterministic	Depends on initialization	Depends on initialization	Deterministic
Required parameters	Number of clusters K	Number of clusters (mixtures) K	Number of clusters K, or threshold for merging

Table 2.1: *Comparison of K-means from Section 2.8.3, GMM from Section 2.8.4, and AHC from Section 2.8.5 algorithms.*

[8]There is a derivation of K-means called K-medoids which can also use precomputed distance matrix, but its runtime is very slow.

[9]Good scalability due to the MiniBatch K-means introduced in Section 2.8.3

[10]Distance metric is only Euclidean, if not convergance is not guaranteed. However, cosine similarity can be utilized as explained in Section 2.8.3

2.9 Evaluation Metrics and Criteria

The goal of this thesis is to find the most accurate way to evaluate SRE on unlabeled datasets. For that we need to create pseudo-labels which correspond to clusters obtained from clustering algorithms. This is influenced not only by the choice of the algorithm, but also by the number of created clusters. All the clustering algorithms mentioned in Section 2.8 have one thing in common; the number of clusters must be specified beforehand. The evaluation metrics and criteria described in this section can be divided into two groups.

- Biometric system evaluation metric
  - Equal Error Rate (EER) in Section 2.9.1.
- Clustering performance metric
  - Elbow Method in Section 2.9.2.
  - Silhouette Score in Section 2.9.3.
  - Calinski-Harabasz (CH) Score in Section 2.9.4.
  - Davies Bouldin (DB) Score in Section 2.9.5.

2.9.1 EER – Equal Error Rate

Equal Error Rate (EER) is a measure frequently used in biometric systems, typically when operating in the verification task [44]. It predetermines the threshold values for the system's *false alarm (FA)* rate and *miss* rate. For clarity, the miss and the false alarm are shown in Table 2.2. The EER is the location on the receiver operating characteristic (ROC) or detection error tradeoff (DET) curve where the common value of the false alarm and the miss rate are equal. The lower the EER, the higher the accuracy of the biometric system. It is also important to note that this metric is calibration-insensitive. In the context of this thesis, EER is used to obtain the baseline calculated in the labeled dataset, and is also used in the elbow method in Section 2.9.2.

Confusion Matrix

		Actual	
		Same Speaker	Imposter
Predicted	Same Speaker	Hit (True Positive)	**False Alarm** (False Positive)
	Imposter	**Miss** (False Negative)	Correct Rejection (True Negative)

Table 2.2: *Confusion matrix with actual and predicted result. The* same speaker *is the correct outcome and the* imposter *is the false outcome. The cells in red are used to obtain the EER.*

The scores gathered from SRE on testing data with several non-target and target embeddings typically create two Gaussian-like curves as shown in Figure 2.10. This shows us how scores are distributed and can be used for a simple visual evaluation of each algorithm against the baseline.

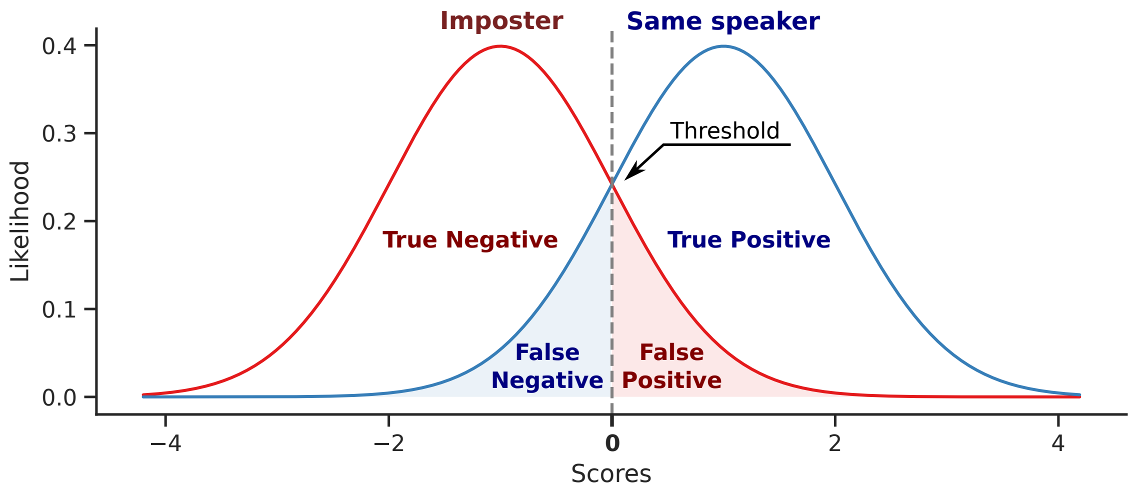


Figure 2.10: *Two probability distributions of an imposter and same speaker scores. The threshold corresponds to the score where the probability (area under curve) of* false negative *and* false positive *are the same values. This value is* EER*.*

2.9.2 Elbow Method

The elbow method is an approach used in the analysis of clustering algorithms to find the right number of clusters K, which in perfect case can be infered *visually* from the figure [45]. The idea behind this method is to choose a point where increasing the number of clusters K is no longer worth the additional cost and adding another cluster $K+1$ does not provide better data modeling and leads to *overfitting*.

In cluster analysis, the elbow curve is displayed as a dependency of the number of clusters K on an evaluation metric that forms a shape that resembles an „elbow“. The criterion can be inertia for K-means explained in Section 2.8.3, as is also illustrated in Figure 2.11, BIC and AIC for Gaussian Mixture Models in Section 2.8.4, or even EER introduced in Section 2.9.1.

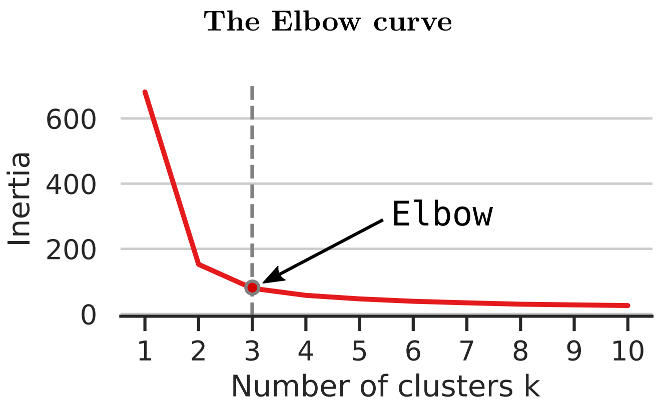


Figure 2.11: *Function in a shape of an elbow showing inertia per number of clusters*[11]*. Optimal number of clusters K was chosen by* Kneedle [3]*.*

The main drawback of this method, as we can see in Figure 2.11, is the „fuzzines“ of choosing a single value as the *elbow point*. If we ask several people to find the elbow, we might get different results. This problem becomes even worse when the spread between possible elbow points increases and the elbow point becomes less sharp. Using the visual elbow method becomes unreliable; luckily, there exists a way to automate it. Another issue arises when the curve does not resamble the elbow curve; however, this might be a problem of the chosen metric, the clustering algorithm, or the data itself might not be suitable for cluster analysis as a result of too much noise or the overall distribution of the data which does not form groups or clusters at all.

The Elbow Method and its Automation

The problem of finding an elbow (or a knee of curvature) in systems behavior seems straightforward. Nonetheless, there are not many general-purpose tools to automate this task, nor is there an accepted definition of what an elbow is. In many areas, researchers usually use system-specific approaches to detect elbows. However, this requires a *robust* and *fine-tuned* detection system. That might be demanding or close to impossible to develop if the input to such a system is difficult to predict and volatile. In the case of speech singals, there are way too many variables such as channel, language, background noise etc. These directly influence the output of the SRE systems and thus make the creation of a system-specific approach problematic.

Fortunetely, there is an algorithm and tool called *Kneedle* [3] that can be used for *general-purpose elbow detection*. It defines the elbow with a mathematical definition of *curvature* – the measure of how much the function differs from a straight line. Detailed explanations of the inner workings of the Kneedle algorithm are described in this paper [3]. The authors also maintain a Python package[12] with implementation of this tool, which is simple to integrate, and found its use cases in various areas.

2.9.3 The Silhouette Score

The Silhouette score is a metric that is used to calculate the goodness of a clustering technique [46][47]. Assume that the data are clustered into K groups C. The Silhouette score takes into account the mean within-cluster distance of an observation x_i assigned to C_k:

$$a(x_i) = \frac{1}{|C_k| - 1} \sum_{i' \in C_k} d(x_i, x_{i'}) \tag{2.28}$$

The lower the value of $a(x_i)$, the better the assignment. Next, we define the mean dissimilarity of x_i from a different cluster $C_{k'}$ as:

$$b(x_i) = \min \frac{1}{|C_{k'}|} \sum_{i' \in C_{k'}} d(x_i, x_{i'}) \tag{2.29}$$

$b(x_i)$ is the mean distance from x_i to all the points in $C_{k'}$. The distance $d(x_i, x_{i'})$ used in (2.28) and (2.29) can be defined as Euclidean, cosine similarity, or any other pairwise

[11] Iris dataset provided by Scikit-learn Python library with three different groups was used and clustered with K-means.

[12] Kneed – Knee-point detection in Python

distance. Even the scores generated from the scoring backend described in Section 2.6 can be provided as a pre-computed distance matrix.

The Silhouette score for the observation x_i is then defined as

$$s(x_i) = \frac{a(x_i) - b(x_i)}{\max\{a(x_i), b(x_i)\}} \tag{2.30}$$

It yields values between -1 and 1. The closer to -1, the more likely the observation is assigned to a wrong cluster, and the closer to 1 the more likely it is to be in the correct cluster.

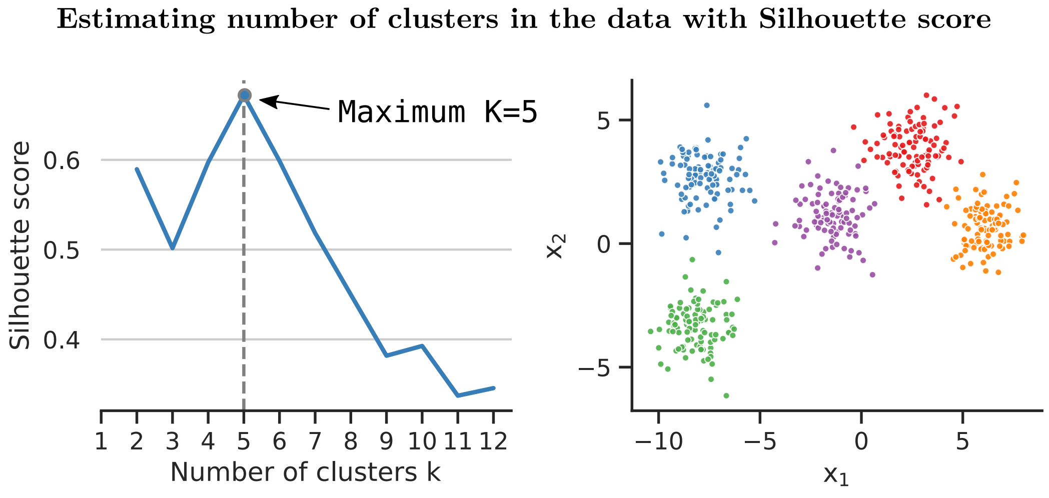


Figure 2.12: *Silhouette score for K-means with number of clusters K on the horizontal axis in (1). The maximum of the Silhouette curve $K = 5$ was used in clustering, result is shown in (2). Data were clustered with K-means.*

The average Silhouette score (2.30) across all observations x_i is then a measure of the quality of the clustering of the data. As illustrated in Figure 2.12, the maximum value of the Silhouette score computed for different numbers of clusters K can be a good indicator of how many clusters there are in the data.

2.9.4 Calinski-Harabasz (CH) score

The Calinski-Harabasz (CH) [48] score (or index) is an unsupervised evaluation method for clustering models. The CH score is also known as Variance Ratio Criterion because it is defined as

$$CH(C) = \frac{N-K}{K-1} \cdot \sum_{k=1}^{K} \frac{B(C_k)}{W(C_k)} \tag{2.31}$$

where N is the total number of all observations and K is the total number of all clusters, within cluster variance $W(C_k)$ defined in (2.17) and $B(C_k)$ is between cluster variance defined as

$$B(C_k) = \sum_{k=1}^{K} n_k ||c_k - c||^2 \tag{2.32}$$

where c is the centroid of the entire dataset and c_k is a class centroid, n_k is the number of points in the cluster C_k.

From (2.31) it is clear that models with clusters that are further apart with higher $B(C_k)$ and have low within-cluster variance $W(C_k)$ will have higher $CH(C)$ Which means that higher values of CH score means indicate better clustering. Additionally, we might also be interested in peak values with local maximum.

2.9.5 Davies-Bouldin (DB) score

The Davies-Bouldin (DB) score [49][47] is an unsupervised evaluation technique for clustering models. The DB score is obtained as

$$DB = \frac{1}{K}\sum_{k=1}^{K} \max_{k \neq l} R_{kj} \tag{2.33}$$

where K is the number of clusters and R_{kj} is the similarity defined as

$$R_{kl} = \frac{s_k + s_l}{d_{kl}} \tag{2.34}$$

where s_k is *cluster diameter*, the average distance between each point of a cluster k and the centroid c_k and d_{kj} is a distance between cluster centroids c_k and c_j. The similarity R_{kj} compares distance between clusters with their size. The farther apart and with lower variance the clusters are, the better score from (2.33) we get. Thus, *lower* values of DB are an indication of better clustering models.

2.9.6 Other Notable Methods

Several other methods were tested in regard to estimating the number of pseudo-labels from the embeddings. However, they did not show reliable results for this type of data and were not used later:

- Gap Statistics
- Jensen-Shannon metric (GMM only)
- Elbow strength[13]

[13]How to Automatically Determine the Number of Clusters in your Data: https://community.ibm.com/community/user/datascience/blogs/moloy-de1/2020/07/02/points-to-ponder

Chapter 3

Datasets

3.1 SITW – Speakers in the Wild

Speakers in the Wild (SITW) speaker recognition database contains open source media samples [50]. As the name suggests, the database includes speech data acquired under „wild“ conditions with natural speech-degrading artifacts, such as real noise, reverberation, and compression artifacts. The same factors are ubiquitous in the real world. Speech samples are hand-annotated and consist of 299 different speakers. All of the above makes the database well-suited for speaker recognition technology benchmarking.

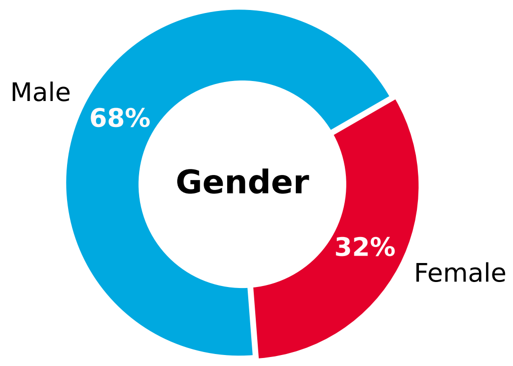


Figure 3.1: *SITW gender distribution.*

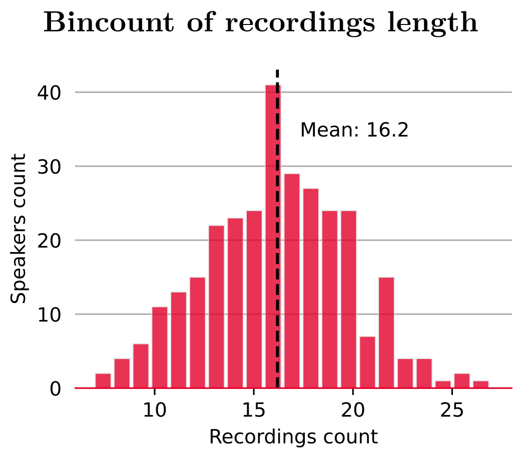


Figure 3.2: *Bincount of recordings count per speaker with binwidth set to 1.*

Most SITW speakers are male, as shown in Figure 3.1. The number of recordings per individual speaker in Figure 3.2 has a mean of 16 recordings. This might be different from some production environments, where the ratio of the number of recordings to a speaker can follow an exponential distribution. The average length is approximately 216 *s* with a median of 91 *s*. The lower part of the length distribution of the recording is shown in Figure 3.3. The longest recording has as much as 5715 *s*.

The sitw dataset features two *enrollment conditions* (core, assist) and two *test conditions* (core, multi). The final four *trial conditions* for development and evaluation purposes are formed as their combination. In this thesis, we use only the *dev-core-core* and *eval-core-core* conditions with the core enrollment and core test conditions.

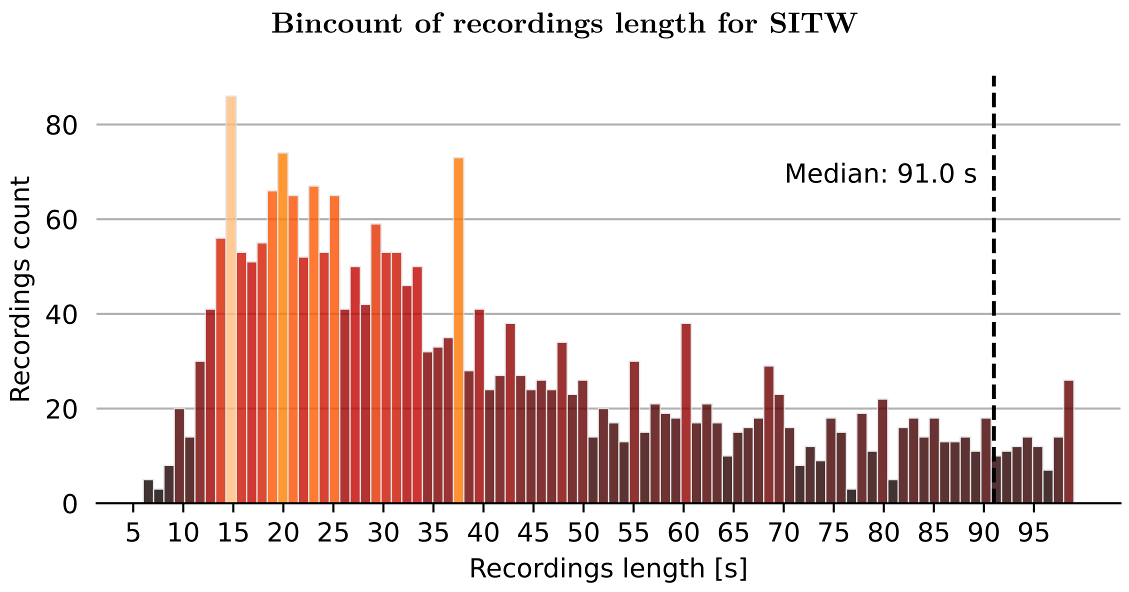


Figure 3.3: *SITW recordings bincount showing subset of recordings length distribution from* 0 *s to* 90 *s with median at* 91 *s.*

The core enrollment condition contains audio files with a contiguous speech segment of only *one speaker*. Similarly, the core test condition also contains speech only from a *single speaker* with an expected amount of speech between 6 and 180 seconds. Note that we use these trial conditions to select recordings for further processing with clustering algorithms and to evaluate the result on all selected recordings. A detailed explanation of the extraction of audio segments in SITW dataset is given in [50].

3.2 NIST SRE16

The 2016 speaker recognition evaluation (SRE16) dataset is part of the ongoing series of speaker recognition started in 1996 by NIST[1] [51]. The SRE16 is made of telephone speech recordings that were collected as part of the Call My Net Speech Collection to support research in speaker recognition. Participants were instructed to call people from their social networks. They were encouraged to use different types of phone devices, such as a cell phone or landline, and to make calls from various environments ranging from a noisy street to a quiet office.

The mean recording length is 88 *s* with a median of 80 *s* and the longest recording is 386 *s* long as illustrated in Figure 3.5. It has a similar proportion of male and female speakers as shown in Figure 3.4 and the distribution of the recordings approximately follows the normal distribution with a mean of 52.2 recordings. However, this data set features less speech-degrading artifacts than SITW introduced in Section 3.5 and VoxCeleb1 explained in following Section 3.3.

For this data set, we use the languages *Tagalog*[2] *(tgl)* and *Cantonese*[3] *(yue)* as test conditions in the experiments in Chapter 4.

[1] National Institue of Standards and Technology

[2] Tagalog language is spoken by native Tagalog people who make up a quarter of Philippines.

[3] Cantonese is a dialect of the Chinese language spoken by more than 60 million people in China.

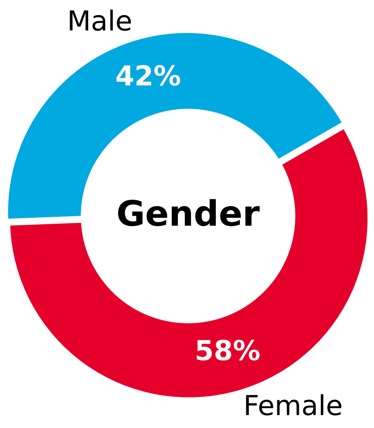


Figure 3.4: *SRE16 gender percentage*

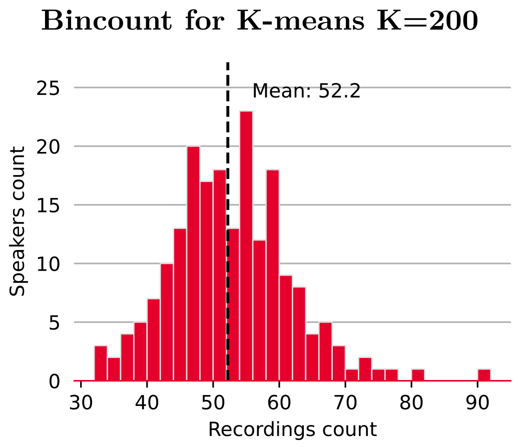


Figure 3.5: *Bincount of recordings count per speaker with binwidth set to 2.*

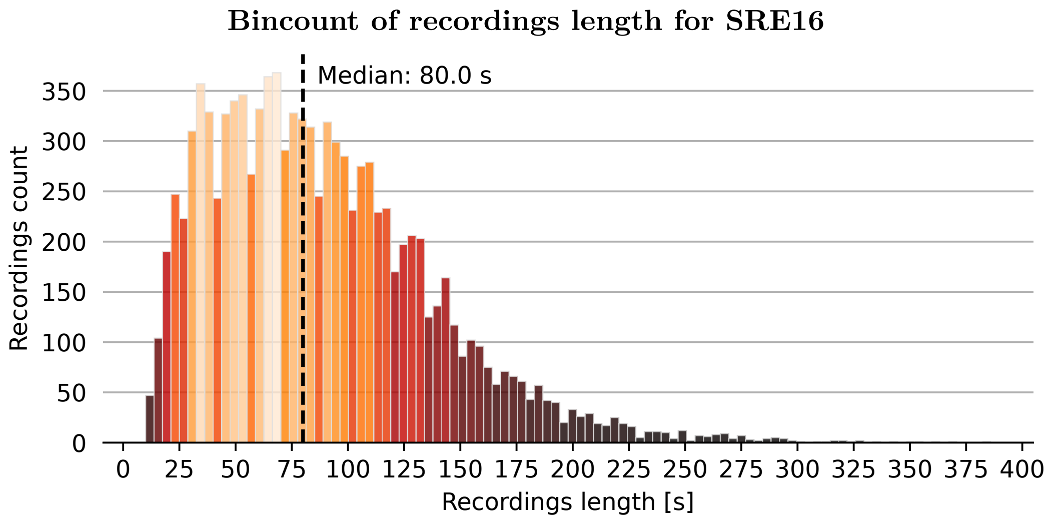


Figure 3.6: *SRE16 recordings bincount showing subset of recordings length distribution from* 0 *s to* 386 *s with the median at* 80 *s.*

3.3 VoxCeleb1

VoxCeleb1 dataset consists of audio recordings of celebrities which were obtained from YouTube videos [52]. The entire dataset contains over 100,000 utterances for 1,251 celebrities obtained with an automated extraction pipeline. This dataset was created for speaker recognition under noisy and unconstrained conditions. It contains extrinsic variations such as background music, chatter, reverberation, and channel or microphone effects. In addition, there are various intrinsic variants including different ages, accents, intonations, or emotions.

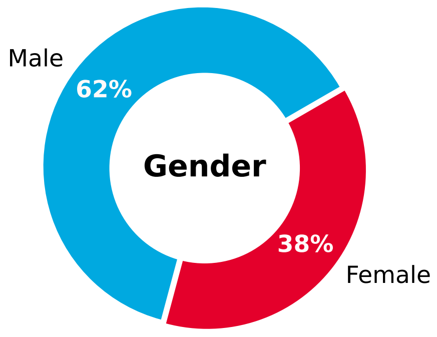


Figure 3.7: *VoxCeleb1 gender percentage.*

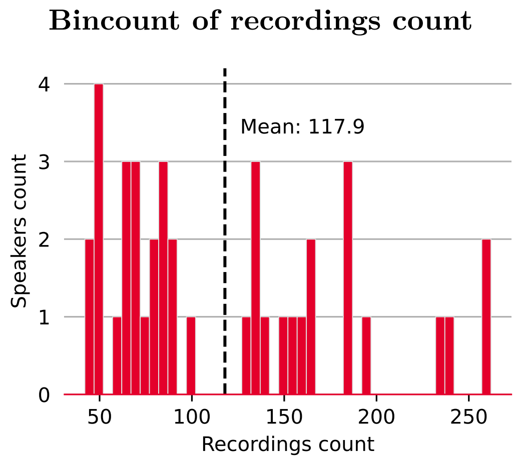


Figure 3.8: *Bincount of recordings count per speaker with binwidth set to 5.*

This dataset is divided into verification and identification split. We used only the verification split with 40 different speakers and 4715 recordings. This dataset has an average recording length 8 s with a median of 6 s as shown in Figure 3.8. The longest recording has length of 69 s. It also roughly follows an exponential distribution. In this work, the whole dataset is labeled as the *eval* condition. The distribution of recordings in Figure 3.8 has a large variance ranging from speakers with around 50 recordings to speakers with a recording count greater than 250 – a very disproportionate dataset. As mentioned in Section 2.8.6, various cluster sizes might be problematic for some clustering algorithms. In this verification split, there are slightly more male speakers than female, as shown in Figure 3.7.

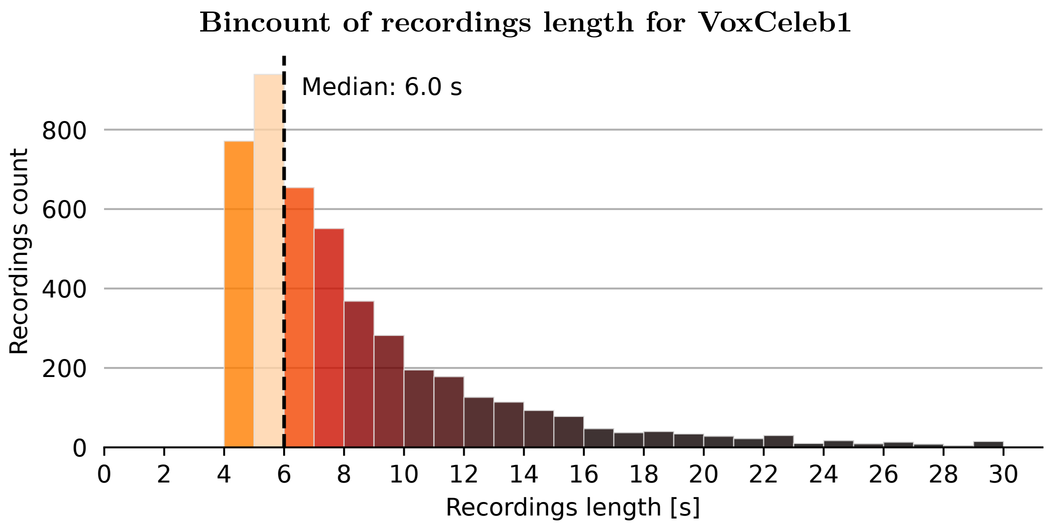


Figure 3.9: *VoxCeleb1 recordings bincount showing subset of recordings length distribution from* 0 *s to* 30 *s with median at* 6 *s.*

3.4 Datasets Summary

Datasets used in this thesis are summarized in Table 3.1. SITW is the smallest dataset, as such, it is fast to process and also has a large number of speakers. We used the dev-core-core condition for the development of the evaluation technique in Section 4.2. The other datasets and conditions were used during the testing phase in Section 4.3.

Dataset	**Condition**	**Recordings count**	**Speakers count**	**Mean** [s]	**Median** [s]
SITW	dev-core-core	823	119	37	29
	eval-core-core	1,202	180	38	30
SRE16	yue (Cantonese)	5,152	100	84	77
	tgl (Tagalog)	4,744	101	85	77
VoxCeleb1	eval	4,715	40	8	6

Table 3.1: *Comparison of datasets with specific conditions and their mean and average recording durations.*

Chapter 4

Experiments

4.1 Experimental Setup

The experimental setup is divided into two parts. Firstly, a brief description of the SID system used for extracting the embeddings from voice recordings, and the scoring backend which compares two embeddings and returns a single score. Secondly, the setup for clustering algorithms and their evaluation techniques. These algorithms are used to create pseudo-labels from the embeddings or by using a score matrix.

4.1.1 SID System

All experiments were carried out with the Phonexia SID system (SID4-XL4), which consists of these programs:

- **vpextract4** – SRE system *frontend* for feature extraction using the x-vector extraction technique described in Section 2.5. The output after processing one recording is a single voiceprint[1], an embedding that represents speech in this recording.
- **vpcompare4** – SRE *scoring backend* explained in Section 2.6 that compares two voiceprints and returns the likelihood score as a floating point number. It can also compare a list of files and in this case create a *score matrix*. This scoring backend uses the cosine metric (older versions use GPLDA).

For the evaluation of SID system we use EER explained in Section 2.9.1.

SID Frontend and x-vector Extractor Architecture

The x-vector extractor works on 8 kHz recordings and uses a deep neural network architecture based on ResNet34 [22]. The input to the neural network are 64 log FBANK features extracted with a 25 ms window with a 10 ms time shift. The frequency range spanned by the Mel filters is 20–3700 Hz. Stochastic gradient descent and angular margin loss are used to train the x-vector extractor. After the pooling layer, a linear transformation is used to reduce the dimensionality to obtain the (256-dimensional) x-vectors.

[1]Voiceprint is a catchy name for embeddings used in the context of Phonexia SID which is more suitable for marketing purposes. Although it is not frequently used in research.

The frontend was trained on various datasets, private and several public[2] such as NIST SRE19 CTC [53] and SWITCHBOARD [54] datasets. To further increase performance of this system, an artificial noise, reverberation, or music was added to the training data. Specific training details are Phonexia's trade secret and are not further discussed.

Scoring Backend Architecture

The scoring backend uses Algorithm 1 explained in Section 2.6. Both the precomputed mean vectors, the LDA projection matrix, and the PLDA were trained on a subset of the training set which was used to train the SID frontend.

4.1.2 Clustering Algorithms

We decided to use centroid-based *K-means* described in Section 2.8.3, distribution-based *GMM* introduced in Section 2.8.4 and an algorithm based on hierarchical clustering called *AHC* explained in Section 2.8.5 with a single and average linkage function. All of these algorithms are used for the development of a technique for unsupervised evaluation and are briefly compared in Table 2.1. We used these distance metrics and hyperparameters for each algorithm:

- **K-means** – Euclidean[3], initialization with K-means++, initialized 10 times.
- **GMM** – Mahalanobis, K-means initialization, full covariance matrix, initialized once.
- **AHC average linkage** – cosine similarity.
- **AHC single linkage** – cosine similarity.

The performance of these clustering algorithms was evaluated with the metrics described in Section 2.9. To find an optimal number of clusters, we fit a 7th-order polynomial curve to the data and used the elbow method and automated it with a Kneedle algorithm introduced in Section 2.9.2. The 7th-order polynomial is a default in Kneedle and works optimally. We tested Kneedle algorithm without a polynomial curve fitting, but it did not worked well. The clustering algorithms created pseudo-labels for the embeddings. The pre-processed embeddings and pseudo-labels were then used in following metrics and criteria:

- **EER** – equal error rate calculated on trials created from pseudo-labels and scores from scoring backend described in Section 2.9.1.
- **Silhouette value** – with the cosine metric explained in Section 2.9.3.
- **CH score** – Calinski-Harabasz score introduced in Section 2.9.4.
- **DH score** – Davies-Bouldin score explained in Section 2.9.5.

[2]Some public datasets have licensing issues and cannot be easily used in commercial products, for example VoxCeleb1 which is created from YouTube videos.
[3]Euclidean distance for K-means with ℓ^2-normalized embeddings also maximize cosine similarity.

4.2 Evaluation Method Development

SITW dataset was chosen for the development of the evaluation method; this dataset contains male and female speech recordings, the length is variable, and the speech conditions are with and without speech-degrading artifacts. Thus, it makes for a great development dataset that is close to real-world data. The development of the evaluation method consists of several parts. Preprocessing of the embeddings, grouping with clustering algorithms using different numbers of clusters at each iteration to obtain pseudo-labels, and finally evaluating the clustered data with several evaluation metrics.

4.2.1 Baseline

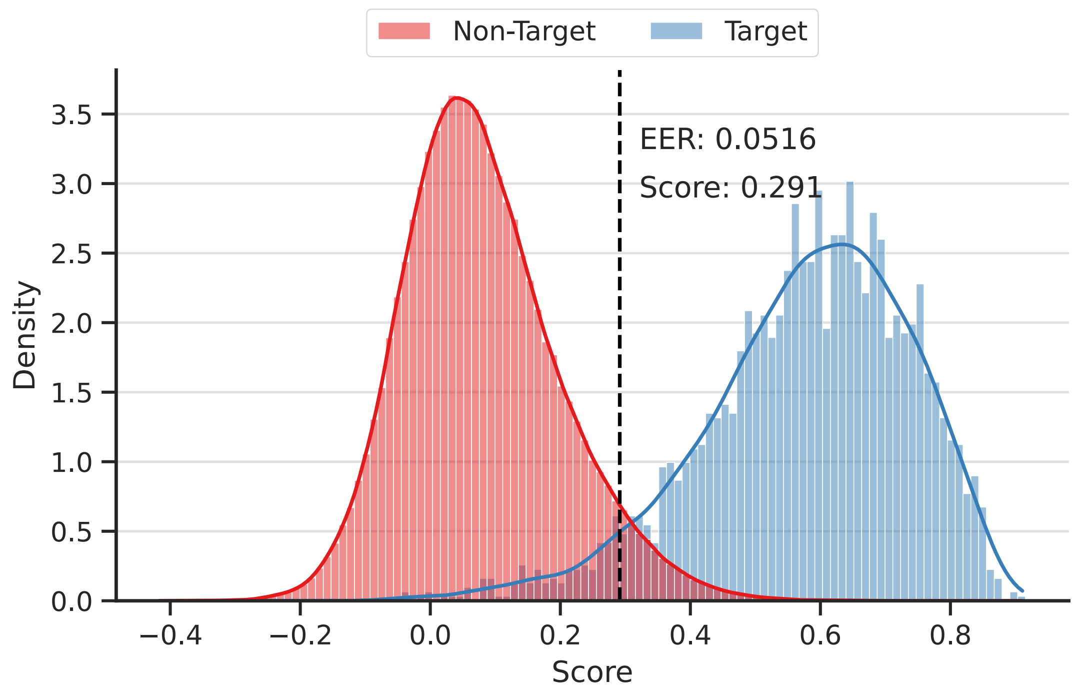
SITW baseline was evaluated for conditions *eval-core-core* and *dev-core-core*. Baseline values were obtained with reference labels by following these steps:

1. *Extract embeddings* of recordings which are part of the condition mask, i.e. remove those columns/rows from the score matrix that are not part of the condition mask.
2. Use the upper triangular part of the score matrix (excluding the diagonal) to *create trials* consisting of two embeddings, the target or non-target label, and the comparison score of these two embeddings from the score matrix.
3. Use these trials to create *baseline figures* and *calculate baseline EER*.

The results of these steps are the baseline histograms for the conditions dev-core-core Figure 4.1 showing two Gaussian-like curves of the score density at each subfigure. Densities of target and non-target trials are normalized separately. The area under each score density is equal to 1. Lines are estimated with KDE[4]. Note that the number of non-target scores is in an order of magnitude larger than the number of target scores; however, when they are normalized with respect to each score type, they are more or less of the same height and can be visually compared.

[4]Kernel density estimation (KDE) – non-parametric way to estimate the probability density function of a random variable.

(a) ***coisne scoring backend*** – *SITW baseline for dev-core-core condition*



(b) ***GPLDA scoring backend*** – *SITW baseline dev-core-core for condition*

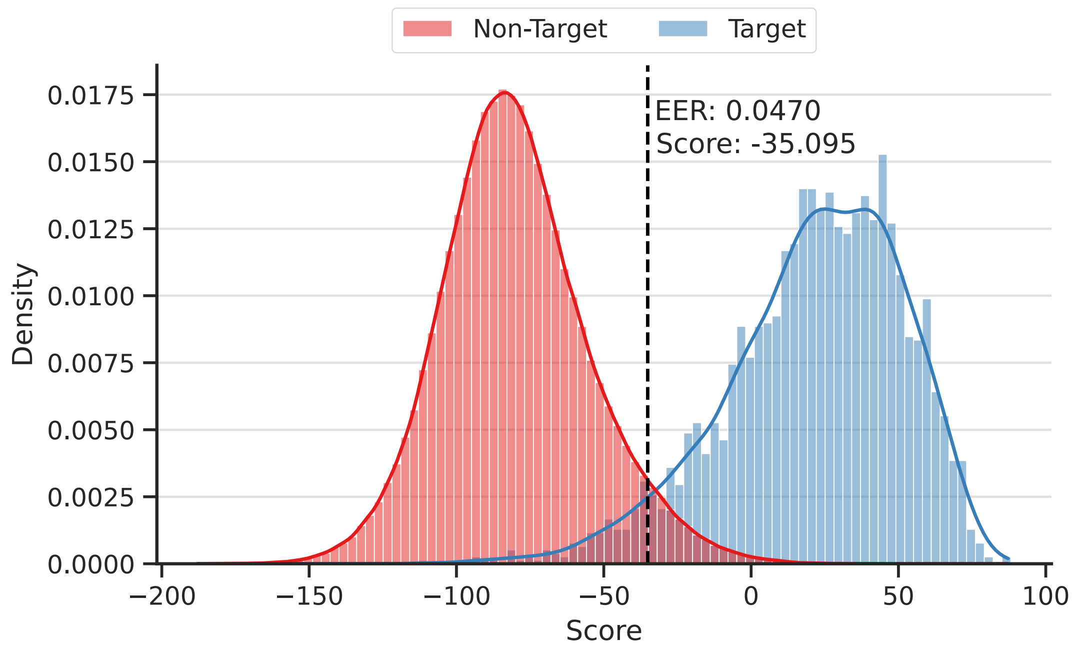


Figure 4.1: *Baselines for SITW dev-core-core condition with target and non-target trials. The thresholds are at scores 0.291 (a) and -35,095 (b) with EER equal to 0.0516 (a) and 0.0470 (b) which represents the value where the probability of miss and false alarm is the same. Scores were calculated using the cosine (a) and GPLDA (b) scoring backend. The contour of the above bins was estimated with KDE.*

EER is equal to 0.0516 for the dev-core-core condition in Figure 4.1a with cosine scoring backend and 0.0470 with GPLDA backend in Figure 4.1b, The goal of the unsupervised evaluation technique is to get as close as possible to the baseline EER without knowing which two pairs of embeddings are the target and which are non-target trials, this information has to be inferred, by clustering or other means, from comparing embeddings and giving them *pseudolabels.*

4.2.2 Embeddings Preprocessing

Raw embeddings with 256 features, as they come in the form of x-vectors explained in Section 2.5, are *not suitable* for clustering and must be preprocessed before being used in another algorithm. Pre-processing is summarized in Figure 4.2, we decided to use the same steps as in Algorithm 1, steps 1.–5. The embeddings are mean normalized with the pre-computed mean vector, which was calculated on the training dataset, and are additionally ℓ^2-normalized. The next step is to reduce the dimensionality with LDA to 128 features. This is useful mainly for two reasons; the reduction of features *speeds up computations* and due to *curse of dimensionality* in Section 2.8.1. (Clustering algorithms work generally better with a smaller number of features per embedding.) Then the steps with mean and ℓ^2-normalization are repeated. This process results in embeddings that have *length equal to 1* and *mean close to 0.* Furthermore, ℓ^2-normalization makes it possible for K-means to work with *cosine similarity*, as proved in Equation (2.24). Finally, we use these embeddings in a clustering algorithm to obtain a pseudo-label for each embedding.

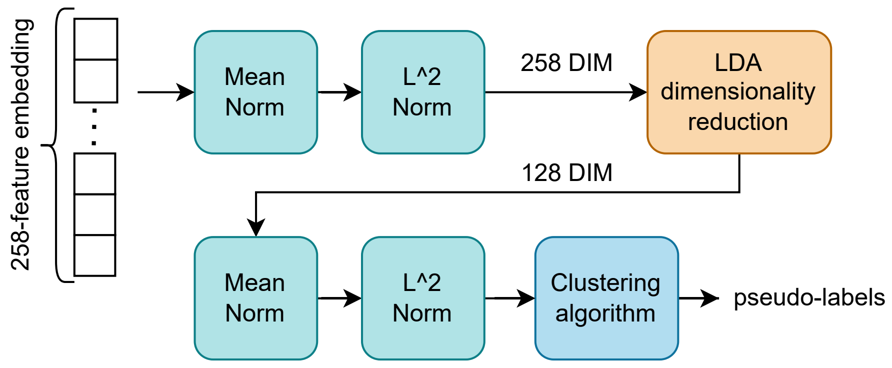


Figure 4.2: *Diagram of a pipeline which was used to obtain pseudo-labels for embeddings with clustering algorithm.*

4.2.3 Clustering and Evaluation

Before evaluating the dataset we have to create *pseudo-labels* which are then used to calculate the EER evaluation metric. For this purpose, we use clustering algorithms to create *clusters* that correspond to these pseudo-labels. *K-means* and *GMM* are a good fit because the embeddings follow a Gaussian distribution. *AHC* with a single and average linkage function is a flexible clustering algorithm and can cluster up to a given threshold.

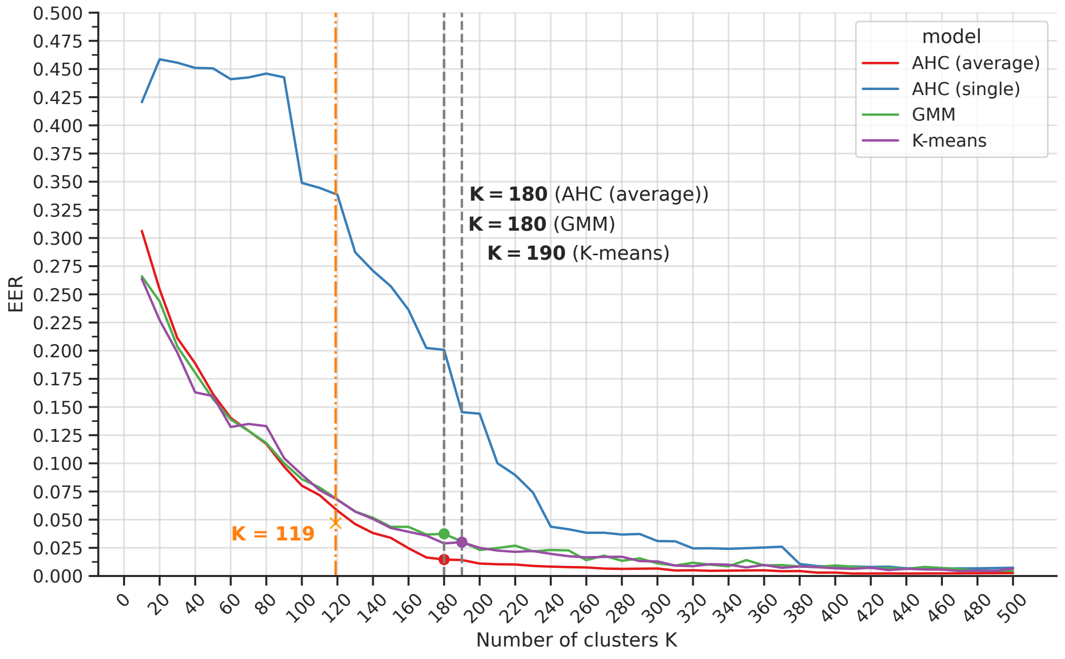
We also run experiments with AHC and a precomputed similarity matrix from the cosine or GPLDA scoring backend; they showed nearly identical results as those of AHC with cosine similarity. Similar results to those obtained with AHC with an average linkage function were also obtained using AHC with a complete linkage. These results are not discussed in further detail.

Testing Criterions for Different Algorithms

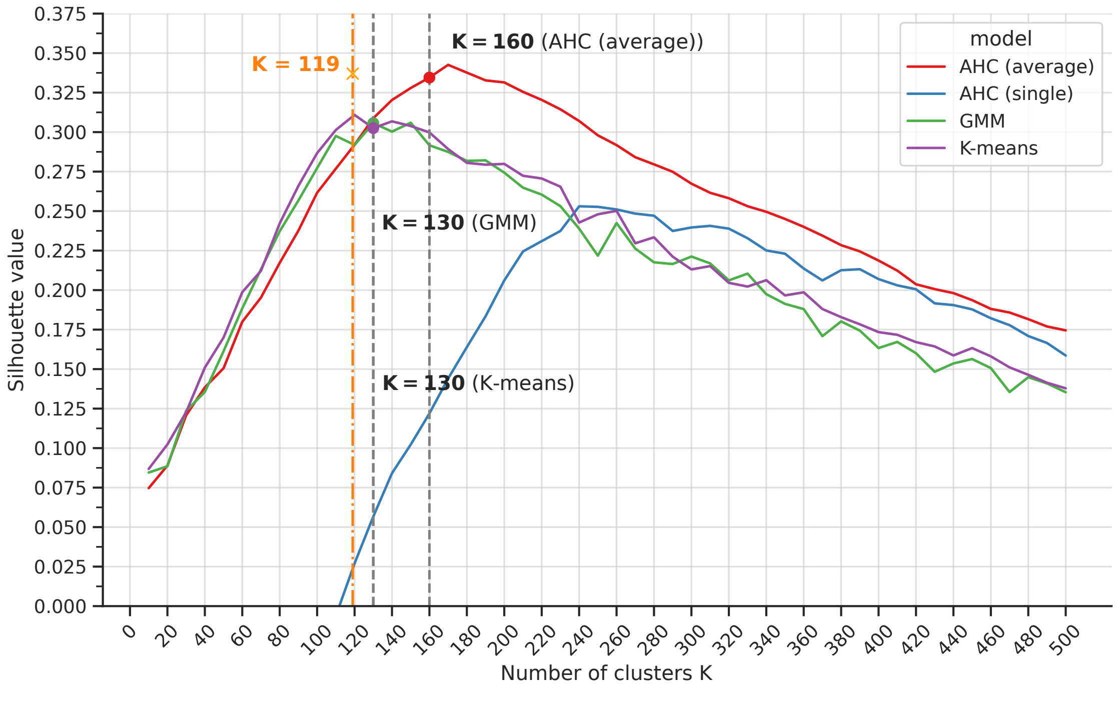
The hardest part of unsupervised evaluation is how to find an optimal number of clusters and overall representative clusters that best describe the data, i.e. they are clustered similarly as in baseline. We used the evaluation metrics discussed in Section 4.1.2 of the experimental setup. Their dependancy on some variable, in our case, the number of clusters, increases or decreases *sharply* up to a breaking point, called an *elbow*. From this point on, the trend becomes steady. Others display a visible maximum, such as the Silhouette value, or minimum. Detection of these knee points was automated with the Kneedle [3] algorithm. The results of the tests are illustrated in Figure 4.3.

The dev-core-core condition of SITW was used for the development purposes of the evaluation method. The algorithms were run with K ranging from 10 to 820 with step 10. For each step, the cirterions were calculated from generated clusters and pseudo-labels. These runs are called *models* because they categorize and model embeddings into a certain number of clusters. As shown in Figure 4.3, the K-means and GMM test runs perform very similarly, resulting in the same K for all criteria. The AHC with average linkage also shows comparable performance, but slightly worse in finding the optimal K. On the contrary, the AHC with single linkage displays a distinct behavior, making it difficult to find the optimal number of clusters, and the gray dashed line that marks the found K is omitted for this algorithm.

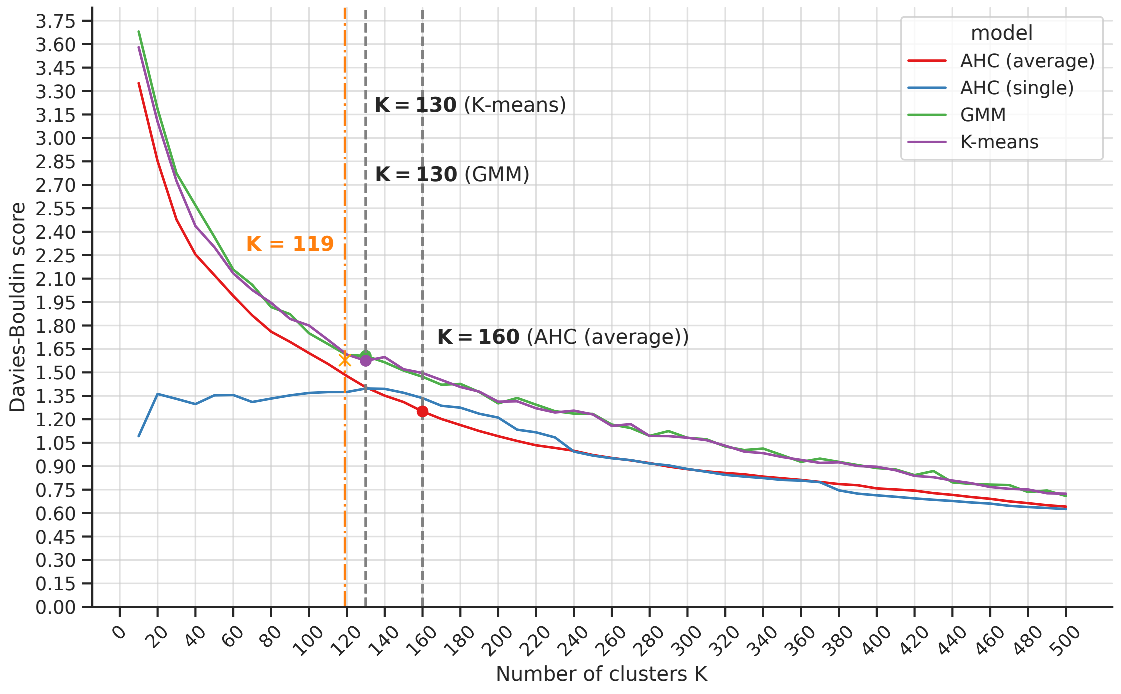
(a) ***EER*** *elbow plot.*



(b) ***Silhouette value elbow plot***



(c) ***DB score*** *elbow plot*



(d) ***CH score*** *elbow plot*

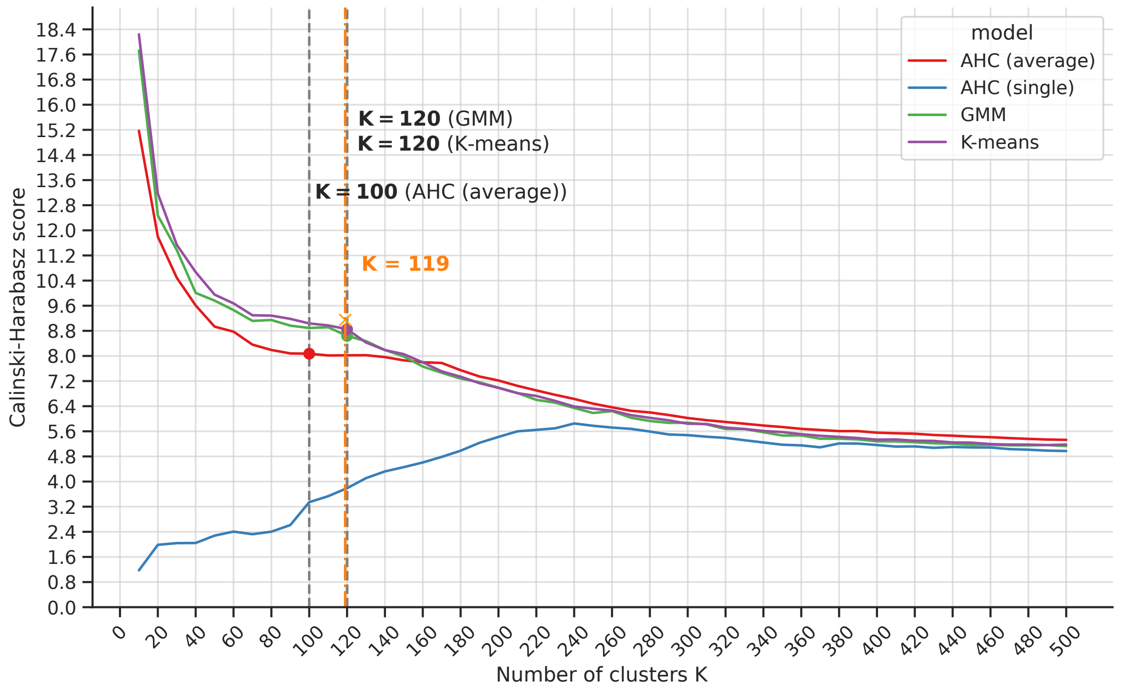


Figure 4.3: *Criteria for finding the number of clusters K with four metrics; EER (a), Silhouette (b), DB (c), CH (d). Each for four models tested on the SITW dataset with condition dev-core-core. Orange dot-dashed line shows real number of clusters, and orange cross shows value of given metric calculated on true labels. Gray dashed lines mark K obtained from each metric and model with the Kneedle algorithm. This condition has 119 speakers and 823 embeddings.*

	K-means	**GMM**	**AHC (average)**	**AHC (single)**	**Baseline values**
Infered number of cluters K	**130**	**130**	160	240	**119**
EER cosine scoring backend	**0.0572**	0.0572	0.0246	0.0438	**0.0516**
EER GPLDA scoring backend	**0.0544**	0.0563	0.0242	0.0438	**0.0470**

Table 4.1: *Summary for each clustering algorithm, K decided by the Silhouette value and Kneedle algorithm with EER calculated from pseudo-labels generated by the algorithm run with infered K. The baseline and optimal values are shown in bold.*

The results from Figure 4.3 are summarized in Table 4.1, optimal K was decided using the elbow of the Silhouette value elbow in Figure 4.3b. We chose K for AHC with a single linkage as the maximum value. The highlighted values are chosen as the optimal among the results from the algorithm runs. Note that the EER in Figure 4.3a for AHC with a single linkage is quite close to the baseline value for GPLDA; however, the model did not show good clustering properties, as shown in the later Figure 4.12 in comparison of cluster sizes. These results are then used in the following figures to explain how the data was clustered.

K-means Inertia

In the K-means algorithm, the *inertia* measure shows how well a dataset is clustered. It is the sum of the squared distances from their closest cluster center. The good model has low inertia and an optimal number of clusters. This metric is used in the elbow method. The result of this approach is shown in Figure 4.4. There is not a very significant elbow, so we decided not to use this method in further experiments.

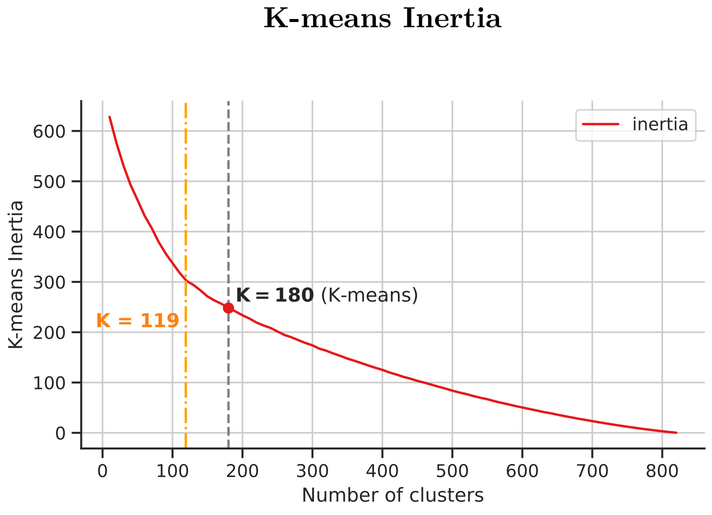


Figure 4.4: *K-means inertia for SITW with condition dev-core-core from 10 to 820 number of clusters K with step 10. The gray dashed line shows K found with Kneedle and a knee point. Orange dot-dashed line shows real number of speakers in the dataset.*

GMM with BIC and AIC

For GMM, we also tested *BIC* and *AIC*[5]. The results of these values can be used in the elbow method; however, for this specific setting, the trend in Figure 4.5 is impossible to use to determine the optimal number of clusters K. For this reason, we did not use this metric in other experiments.

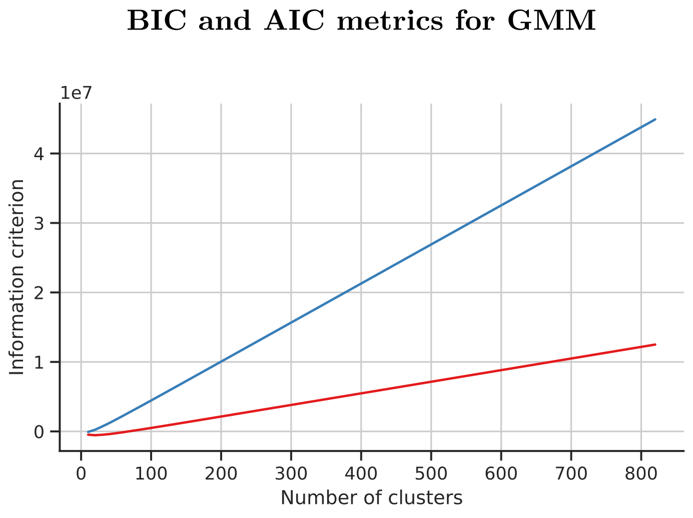


Figure 4.5: *BIC and AIC for GMM tested on SITW dataset with condition dev-core-core from 10 clusters to 820 with step 10.*

Trials Distribution

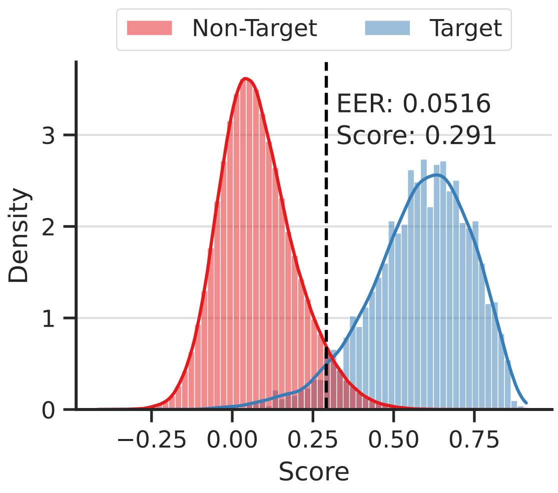
From pseudo-labels we can create target and non-target trials consisting of two embeddings and their score. The same pseudo-labels in the trials are marked as target, and different are non-target. The distribution of these trials obtained from clustering models and their scores is shown in Figure 4.6 compared to the baseline in Figure 4.6a. The distribution for the K-means model in Figure 4.6b and the GMM model in Figure 4.6c are similar to the baseline, there is a large overlap of the miss and FA trials, which contributes to higher EER values.

In contrast, the AHC (average) in Figure 4.6d has a smaller overlap of the miss and FA trials. In general, this model did not assign many embeddings to the same cluster if they have score less than 0.25.

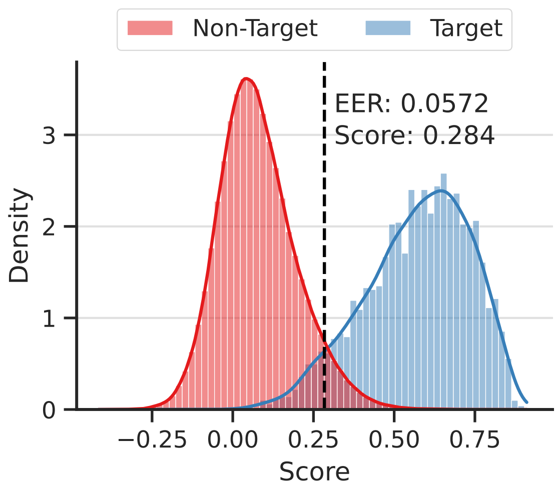
Note that the non-target score distribution does not change in all figures in Figure 4.6. This is caused by a large disproportion between target and non-target scores, which are both normalized independently of each other.

[5]Bayesian and Akaike Information Criterion

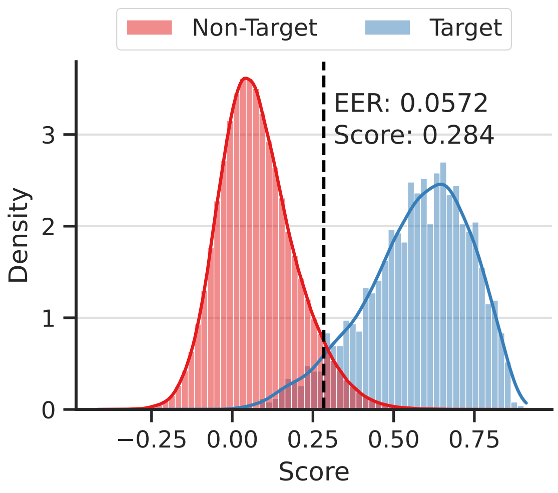
(a) ***SITW baseline*** *with 118 speakers.*



(b) ***K-means*** *with* $K = 130$



(c) ***GMM*** *with* $K = 130$



(d) ***AHC (average)*** *with* $K = 160$

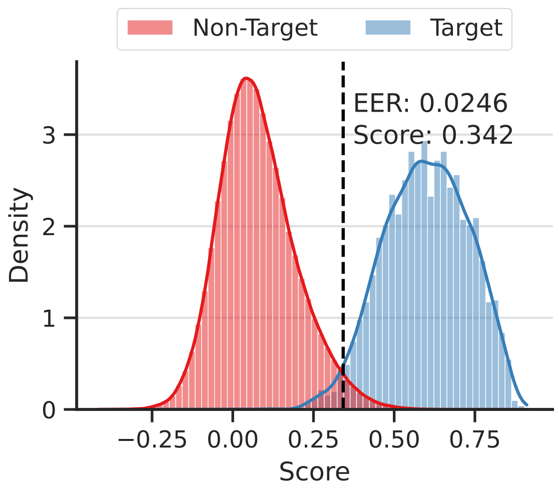


Figure 4.6: *Score distribution for target and non-target trials modeled on condition SITW dev-core-core. Comparison of baseline condition (a) with the trials created by the K-means* $K = 130$ *(b), GMM* $K = 130$ *(c), and AHC (average)* $K = 160$ *(d) models.*

Clusters Size for Different Models

To evaluate results of the clustering algorithms, we use bincounts, with cluster size as bins and number of clusters as heights, to show the distribution of pseudo-speakers after using a specific clustering algorithm. We used the number of clusters K from Table 4.1 and also used $K = 130$ to compare clustering with AHC algorithm with both linkage functions to GMM and the K-means. We compared these figures with the baseline for the SITW dev-core-core condition in Figure 4.7

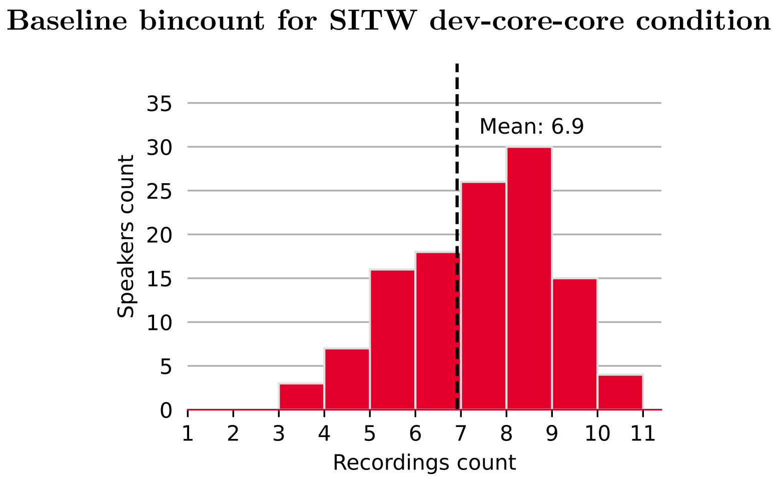


Figure 4.7: *Baseline bincount for SITW with dev-core-core condition which has 119 unique speakers and 823 recordings.*

The bincount of K means in Figure 4.8 follows roughly the same distribution as the baseline in Figure 4.7. The GMM displays similar results. Note that these two clustering algorithms use a *nondeterministic initialization technique* K-means++[6] discussed in Section 2.8.3 and produce different results in different runs. However, the results of different runs do not differ much.

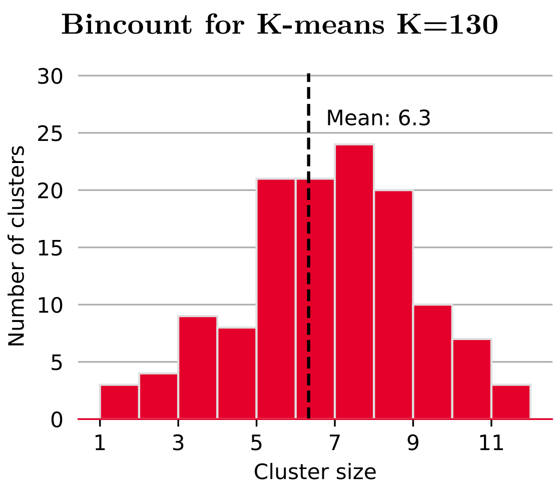


Figure 4.8: *Bincount of recordings for K-means with number of clusters* $K = 130$.

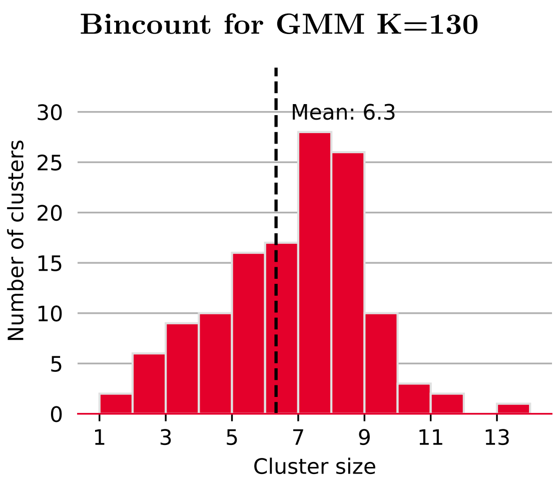
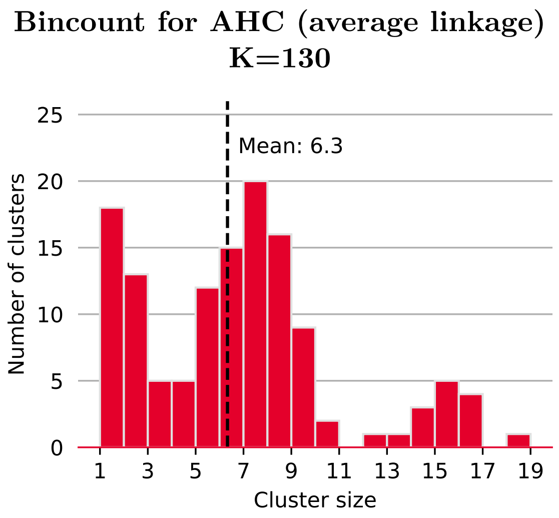


Figure 4.9: *Bincount of recordings for GMM with number of clusters* $K = 130$.

The AHC with the average linkage function bincounts illustrated in Figure 4.10 with 130 clusters and Figure 4.11 with 160 clusters shows that there are many singleton clusters and several large clusters. With increasing K, the number of large clusters decreases because they are divided into smaller clusters. These distributions are quite different from the baseline in Figure 4.7 and also have several large clusters that have more than 11 members.

[6]GMM uses K-means to initialize clusters which in turn uses nondeterministic K-means++.



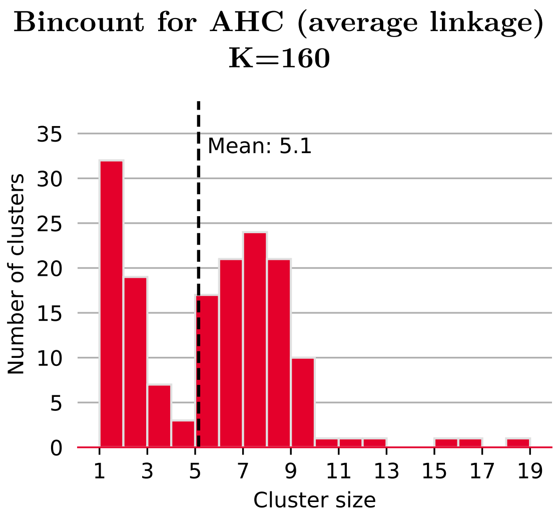


Figure 4.10: *Bincount of recordings for AHC (average linkage) with number of clusters* $K = 130$*.*

Figure 4.11: *Bincount of recordings for AHC (average linkage) with number of clusters* $K = 160$*.*

The AHC with the single linkage function shows the worst behavior. $K = 240$ infered from the maximum of the Silhouette value used in Figure 4.12 displays many singleton clusters and is drastically different from the baseline. Figure 4.13 illustrates *chaining* phenomen introduced in Subsection 2.8.5. As the algorithm worked from the bottom up, it started to link several major clusters by comparing two embeddings/observations from each cluster at a time, resulting in a „chain reaction“ and the creation of three huge clusters highlited in Figure 4.13.

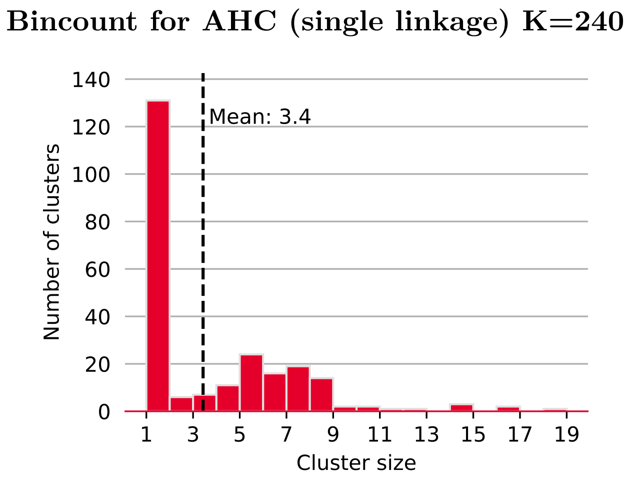


Figure 4.12: *Bincount of recordings for AHC (single linkage) with number of clusters* $K = 240$*. 131 clusters are singletons (their size is 1).*

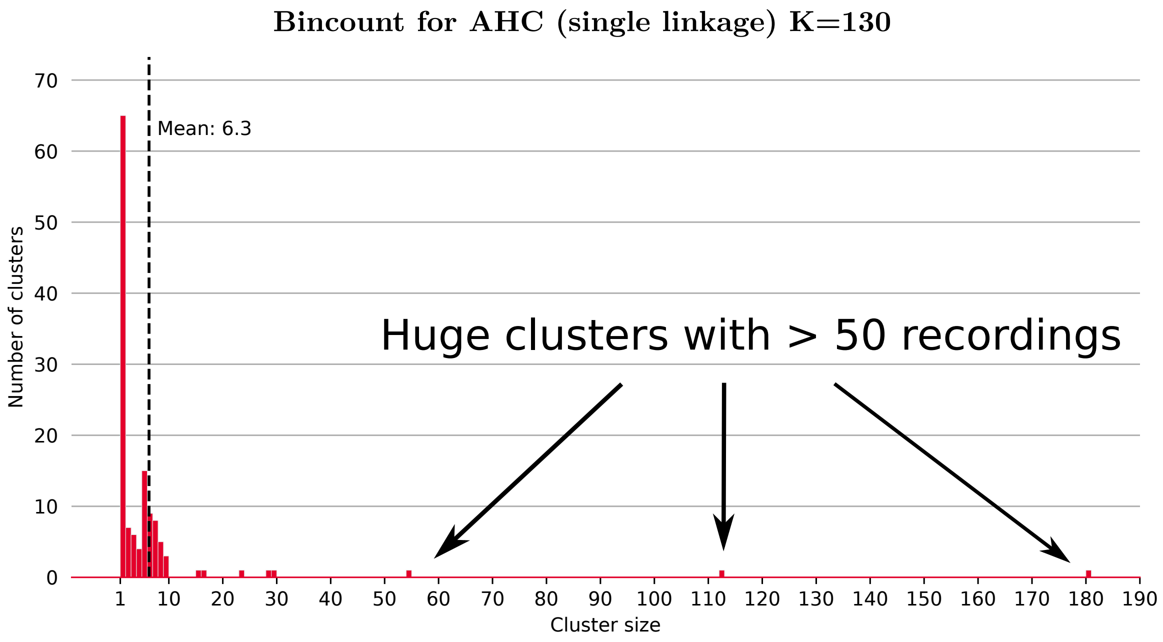


Figure 4.13: *Bincount of recordings for AHC (single linkage) with number of clusters* $K = 130$. *Displaying three huge clusters with size 54, 112, and 181 cluster members and 65 clusters are singletons (size is 1).*

Projection of embeddings by UMAP with true and K-means clusters

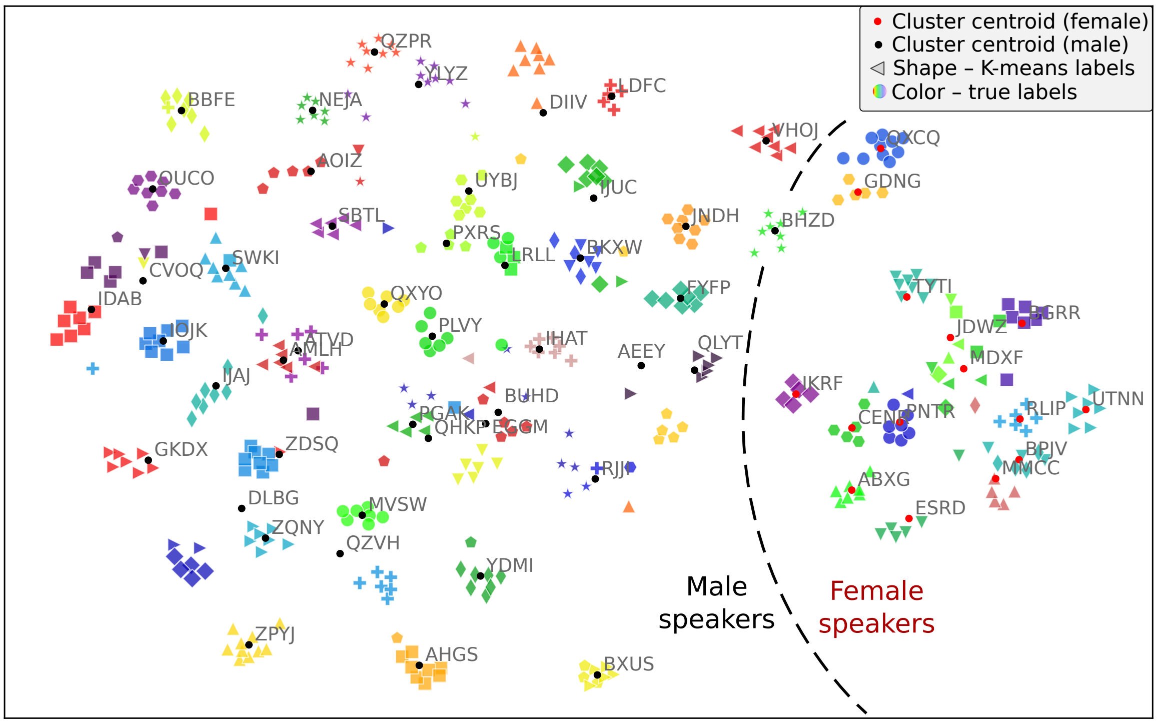


Figure 4.14: *Unsupervised projection of unprocessed embeddings with UMAP*[6] *showing a subset of 60 different speakers (from total 119) separated by a different hue (true labels). Shape represents labels obtained after clustering from K-means with* $K = 130$. *Small black and red dots with text labels are cluster centroids obtained as mean of true speaker label embeddings (mean calculated after 2D UMAP projection). Dashed line separates male and female speakers.*

Projecting Embeddings to Two Dimensions with UMAP

To provide an approximate picture of what clustering algorithms do in multidimensional feature space, we decided to use the dimensionality reduction algorithm called UMAP[6]. Raw embeddings with 256 features were projected with it to two dimensions using an unsupervised approach with the cosine metric as a metric. The resulting projection is shown in Figure 4.14. It displays 60 distinct speakers and their embeddings with different hue and centroids visualized as small dots; some are influenced by outliers and are not in the center of the „blob“ with the same speaker recordings. There is a visible distinction between male and female speakers. Shapes represent cluster assignments (or pseudo-labels) obtained with the K-means algorithm. Many clusters are accurately clustered, as also indicated by the number of correct trials counted in Table 4.2. Only 78 target trials are false.

[6]Uniform Manifold Approximation and Projection (UMAP) was run with cosine metrics and with the spread value set to 28. The total number of 118 speakers was not used because the resulting figure was difficult to read.

		Trials		
		Total trials	Total trials correct	Percentage correct
Trial type	Target	2597	2519	97.0 %
	Non-target	335656	335656	100 %

Table 4.2: *Total number of trials and correct predictions for K-means with $K = 130$.*

4.3 Testing on Other Datasets

For further testing, we decided not to use GMM, as it yields results similar to K-means and is *not scalable*, that is, it is slow for larger datasets. Furthermore, we did not use the AHC with single linkage, as this algorithm has a problem with *chaining*. The test setup for K-means and AHC with average linkage is similar to that in the previous experiment. Each algorithm was run from 10 to 1000 clusters with step 10 and metrics were sweeped for each consecutive run. The elbow point was determined using the Kneedle algorithm. The results for each dataset and their conditions are summarized in Table 4.3. We used these Ks to calculate EER for each dataset, backend, and clustering algorithm model. EER was obtained from generated pseudo-labels from each model, and we compared it to their baseline. These results are shown in Table 4.4. Similarly to previous experiments, to further explain how the clustering algorithms worked on these datasets and conditions, we use figures for all criteria, as well as bincounts with cluster size.

			K obtained from metrics				
Dataset	**Condition**	**Algorithm**	**EER**	**Silh.**	**DB**	**CH**	**Real K**
SITW	dev-core-core	AHC (average)	180	160	160	100	**119**
		K-means	190	**130**	**130**	120	
	eval-core-core	AHC (average)	240	240	290	140	**180**
		K-means	260	**200**	170	120	
SRE16	yue (Cantonese)	AHC (average)	150	140	140	320	**100**
		K-means	160	**120**	**120**	280	
	tgl (Tagalog)	AHC (average)	210	210	210	260	**101**
		K-means	170	170	**130**	140	
VoxCeleb1	eval	AHC (average)	120	**80**	110	210	**40**
		K-means	**110**	40	30	180	

Table 4.3: *Results of finding the optimal K for different datasets, each K was found with Kneedle.*

Dataset	Condition	Backend	Algorithm	EER calculated from K				Baseline EER
				EER	Silh.	DB	CH	
SITW	dev-core-core	cosine	AHC (average)	0.0146	0.0246	0.0246	0.0802	**0.0516**
			K-means	0.0300	**0.0572**	**0.0572**	0.0676	
		gplda	AHC (average)	0.0133	0.0242	0.0242	0.0759	**0.0469**
			K-means	0.0285	**0.0544**	**0.0544**	0.0639	
	eval-core-core	cosine	AHC (average)	0.0282	0.0282	0.0158	0.0873	**0.0508**
			K-means	0.0434	**0.0586**	0.0770	0.1112	
		gplda	AHC (average)	0.0282	0.0282	0.0165	0.0923	**0.0503**
			K-means	0.0398	**0.0607**	0.0777	0.1078	
SRE16	yue	cosine	AHC (average)	0.0161	0.0176	0.0176	0.0062	**0.0281**
			K-means	0.0193	**0.0240**	**0.0240**	0.0126	
		gplda	AHC (average)	0.0172	0.0190	0.0190	0.0068	**0.0295**
			K-means	0.0195	**0.0249**	**0.0249**	0.0131	
	tgl	cosine	AHC (average)	0.0534	0.0534	0.0534	0.0393	**0.1195**
			K-means	0.0563	0.0563	**0.0730**	0.0689	
		gplda	AHC (average)	0.0564	0.0564	0.0564	0.0409	**0.1196**
			K-means	0.0568	0.0568	**0.0743**	0.0706	
VoxCeleb1	eval	cosine	AHC (average)	0.0244	**0.0293**	0.0265	0.0191	**0.0386**
			K-means	**0.0328**	0.0568	0.1247	0.0286	
		gplda	AHC (average)	0.0268	**0.0304**	0.0279	0.0204	**0.0392**
			K-means	**0.0335**	0.0575	0.1225	0.0289	

Table 4.4: *EER calculated with K and pseudo-labels obtained from K-means and AHC with average linkage. Colors represent the difference from the baseline EER; the darker the red, the more negative the difference; the darker the green, the more positive the difference. Lighter hues are closer to the baseline. The numbers in bold are optimal EER values.*

In Table 4.3 we show that the K-means and the Silhouette value work quite well for every test case except SRE16 tgl. DB score in conjunction with the K-means is also a good indicator of the number of clusters. Interestingly, the EER is always around 60 clusters above the real value of K. As mentioned in Section 2.9.2, Kneedle is a general-purpose algorithm to find an elbow point. To be more system-specific, it might be a good step to adjust the final K with bias. On the other hand, the CH score showed that it behaved well only under the specifics of the SITW data set with the condition dev-core-core. Note that all algorithms and criteria did not work well when they were used on the most difficult dataset and condition – SRE16 tgl with EER slightly below 0.12.

Overall, using different scoring backend does not strongly influence the EER obtained by different metrics. Although the EER values for the cosine distance backend were closer to the baseline, this was most likely due to the fact that we used cosine similarity for K-means and AHC.

4.3.1 SITW eval-core-core

The criteria graphs in Figure 4.15 for the eval-core-core condition are similar to those for the dev-core-core condition in Figure 4.3. However, there are two notable distinctions. First, the Silhouette is wider for both models, and the elbow here is not as precise. Second, the elbow of the DB score for AHC (average) is much larger K than the elbow of the K-means model. This is the result of fitting a 7th degree polynomial line, which was influenced by the steep increase at $K = 30$.

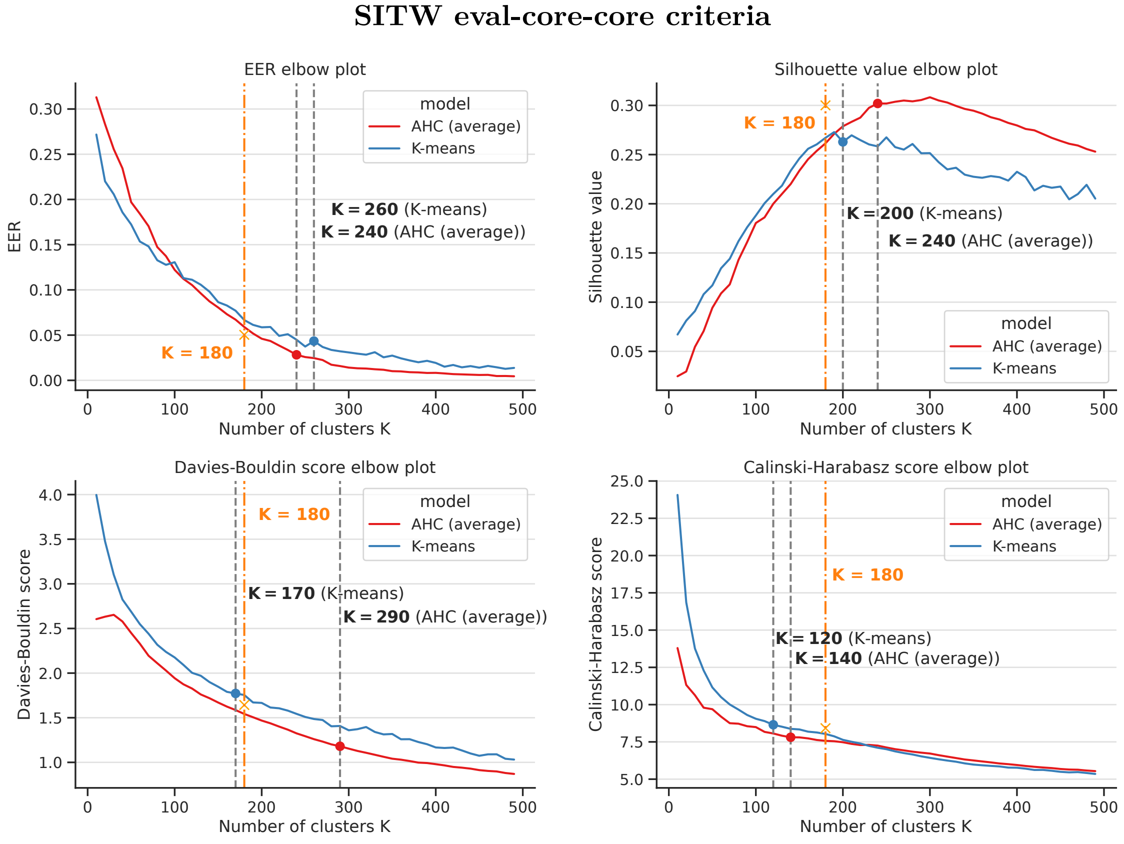


Figure 4.15: *Criteria for dataset SITW eval-core-core. Real number of clusters and metric value are orange. Gray dashed line represents K from each model calculated with Kneedle algorithm. This condition has 180 speakers and 1202 recordings.*

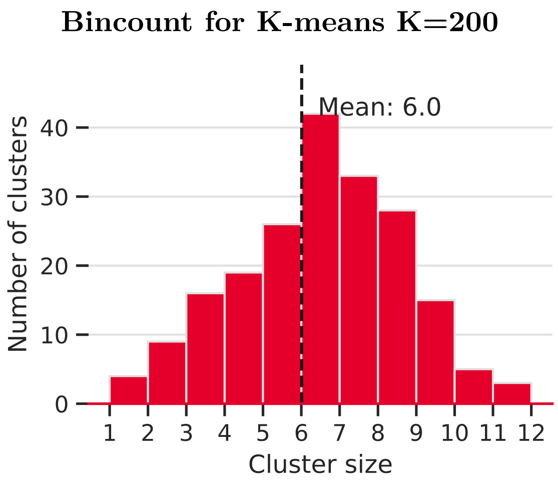


Figure 4.16: *Bincount of recordings for SITW eval-core-core with K-means and number of clusters $K = 200$ chosen by Silhouette value.*

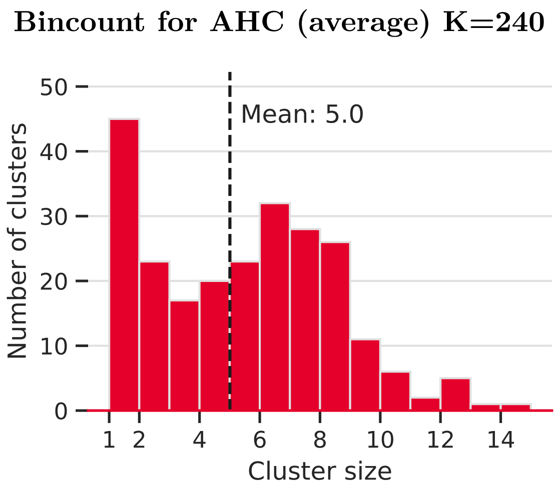


Figure 4.17: *Bincount of recordings for SITW eval-core-core with AHC (average) and number of clusters $K = 240$ chosen by Silhouette value.*

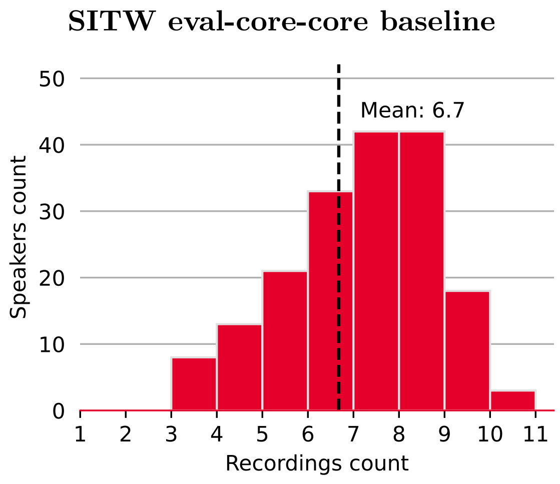


Figure 4.18: *Baseline for SITW eval-core-core with 180 speakers.*

We compared the results for K chosen by the Silhoutte metric with the baseline in Figure 4.18. The K-means model in Figure 4.16 with $K = 200$ is close to the baseline; it also has the same values for the largest number of clusters of sizes 6, 7 and 8. On the contrary, the AHC (average) model with $K = 240$ shown in Figure 4.17 was *comparably worse* and created many *singleton clusters* as in a previous experiment with the dev-core-core condition in the same data set in 4.2.3.

4.3.2 SRE16 yue

The algorithms tested on the SRE16 dataset with yue condition showed *satisfactory* results with visible elbows for the EER, Silhouette, and DB score in Figure 4.19. For the CH metric there is a visible *peak* at $K = 100$, however, it was not detected by the Kneedle algorithm because it does not classify it as an elbow explained in 2.11. Nevertheless, this peak point is probably dataset-specific as it was not shown in other results.

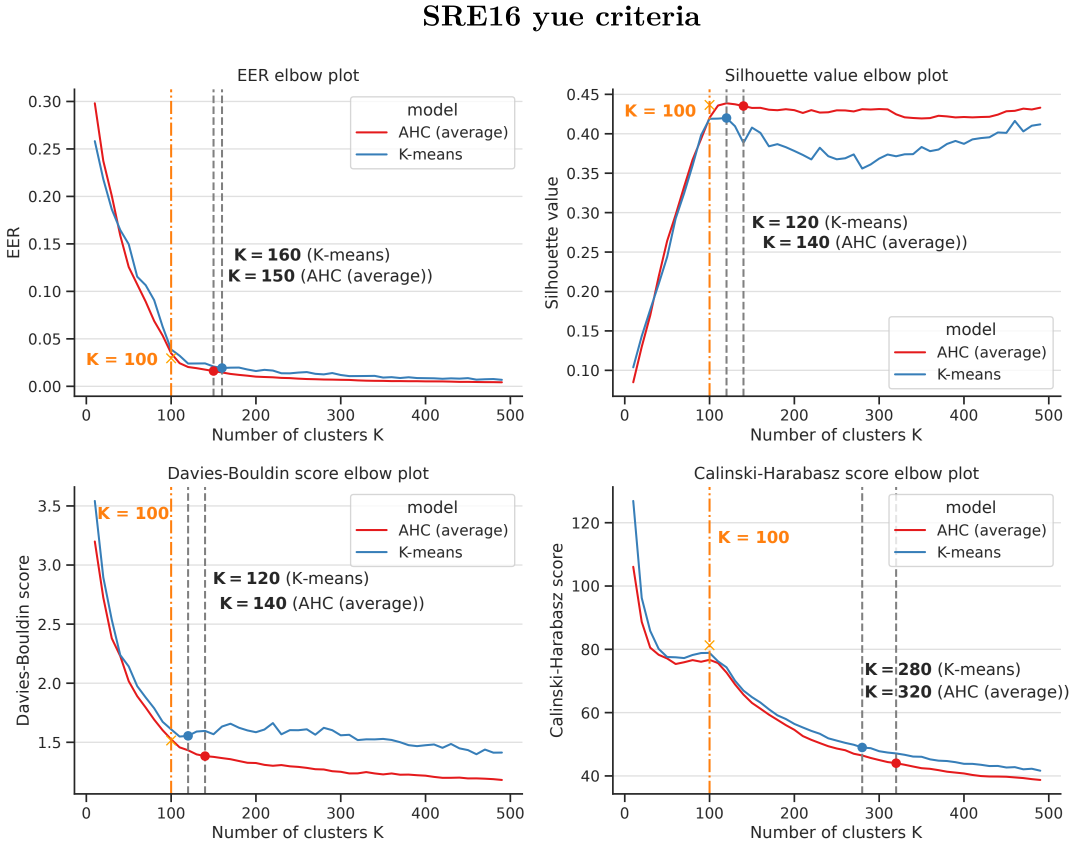


Figure 4.19: *Criteria for dataset SRE16 yue. Real number of clusters and metric value are orange. Gray dashed line represents K from each model calculated with Kneedle algorithm. This condition has 100 speakers and 5152 recordings.*

The K-means model with $K = 120$ in Figure 4.20 shows a distribution similar to the baseline in Figure 4.22, nevertheless, it created more smaller clusters. The model for AHC (average) with $K = 140$ has many singletons as in other experiments.

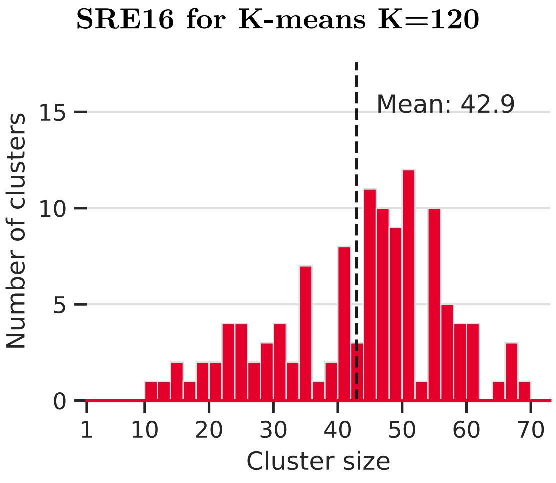


Figure 4.20: *Bincount of recordings for SRE16 yue with K-means and number of clusters $K = 120$ chosen by Silhouette.*

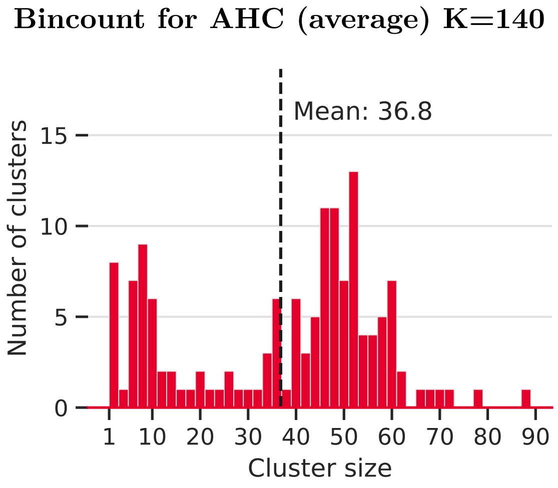


Figure 4.21: *Bincount of recordings for SRE16 yue with AHC (average) and number of clusters $K = 140$ chosen by Silhouette.*

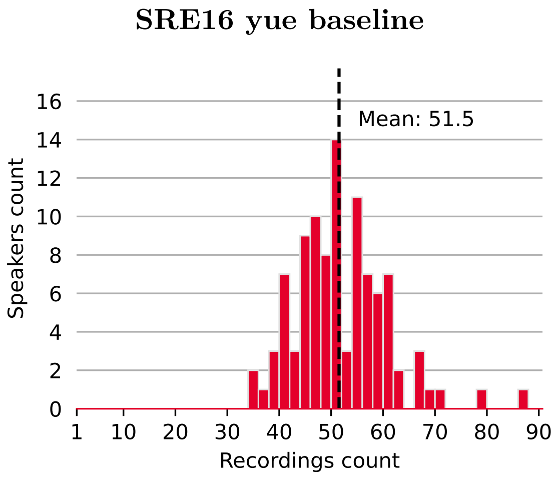


Figure 4.22: *Baseline for SRE16 yue with 180 speakers.*

4.3.3 SRE16 tg

The condition tgl in SRE16 dataset is the most challenging tested. In Table 4.4 it also has the worst EER value and overall results. All the metrics tested in Figure 4.23 show a steep curve that becomes steady after an elbow point. Significant distinction from the other results is that both Silhouette curves do not decrease after an elbow – a sign of a dataset that is difficult to cluster.

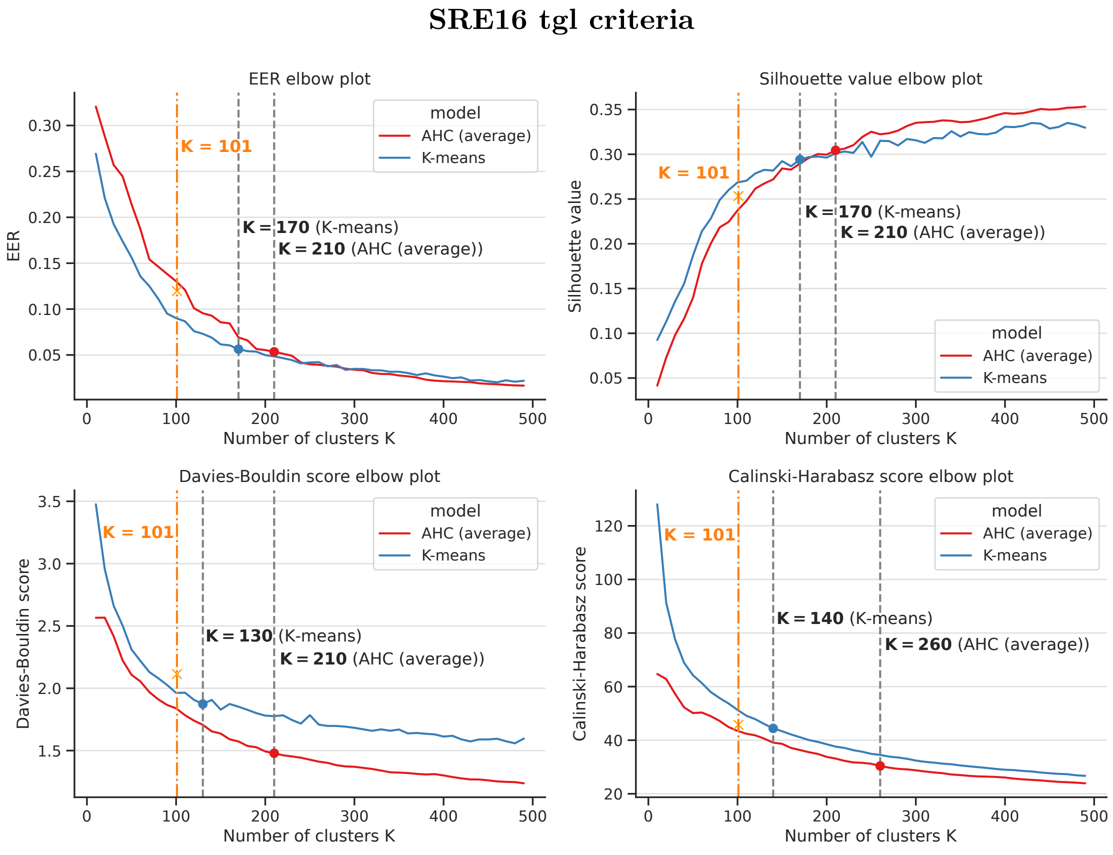


Figure 4.23: *Criteria for dataset SRE16 tgl. Real number of clusters and metric value are orange. Gray dashed line represents K from each model calculated with Kneedle algorithm. This condition has 101 speakers and 4744 recordings.*

Both K-means and AHC with average linkage had problems in clustering this data set. As shown in Figure 4.24 the K-means with $K = 170$ and AHC in Figure 4.25 with $K = 210$ created many smaller clusters than at the baseline in Figure 4.26. It is important to note that this part of NIST SRE16 shows suspicious behavior not only for our experiments, but also from the point of speaker recognition, and was reported by several sites.

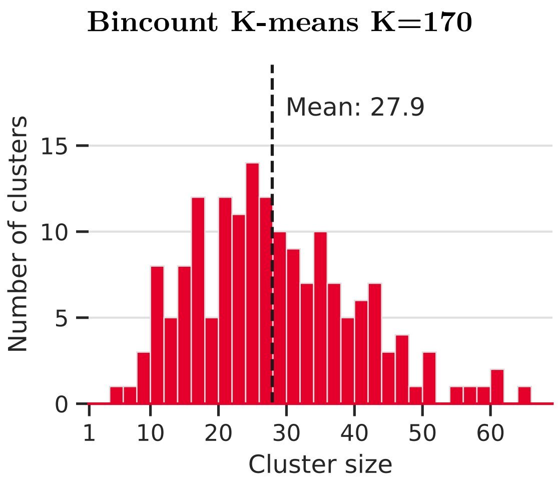


Figure 4.24: *Bincount of recordings for SRE16 tgl with K-means and number of clusters $K = 170$ chosen by DB.*

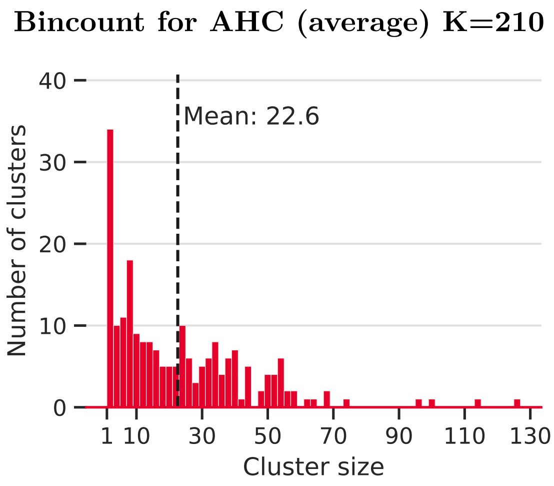


Figure 4.25: *Bincount of recordings for SRE16 tgl with AHC (average) and number of clusters $K = 210$ chosen by Silhouette, EER, and DB.*

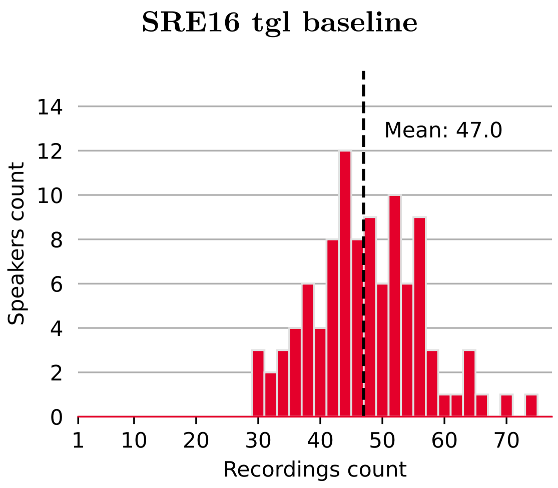


Figure 4.26: *Baseline for SRE16 tgl with 180 speakers.*

4.3.4 VoxCeleb1

Most metrics tested on the VoxCeleb1 dataset showed a visible elbow pattern and a local minimum or maximum, as shown in Figure 4.27. However, finding K close to the real value does not always lead to a good estimation of EER values. In Table 4.4 the baseline EER is approximately 0.039. However, the K-means model with $K = 40$ has EER $= 0.0575$. This error was probably caused by the distribution of VoxCeleb1 speakers and their recording count illustrated in the baseline in Figure 4.31. The K-means model outperformed the AHC with single linkage and with the Silhouette and DB score

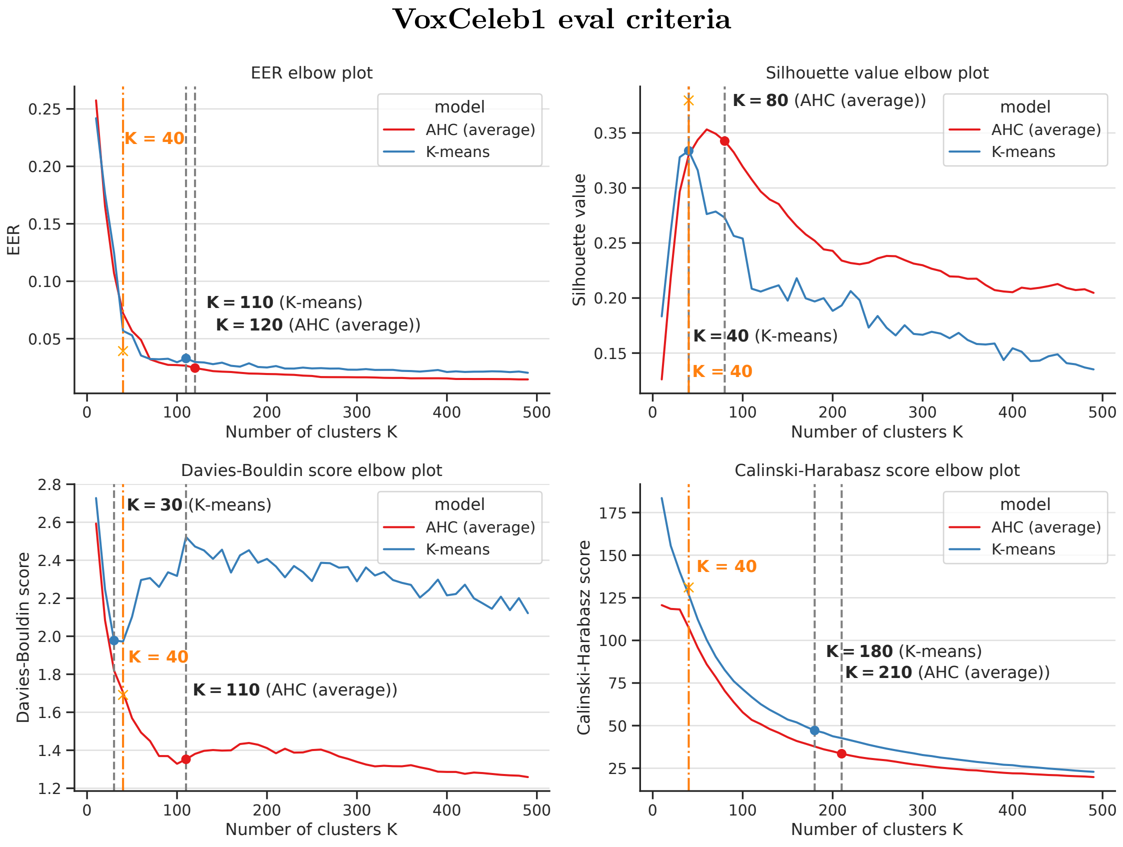


Figure 4.27: *Criteria for dataset VoxCeleb1 eval. Real number of clusters and metric value are orange. Gray dashed line represents K from each model calculated with Kneedle algorithm. This condition has 40 speakers and 4715 recordings.*

The distribution of the size of the clusters is very different from the VoxCeleb1 baseline in Figure 4.31. The K-means in Figure 4.28 with $K = 110$ and AHC with average linkage in Figure 4.29 with $K = 80$ did not model the distribution of the cluster sizes according to the baseline. In addition, the AHC created many singletons.

Furthermore, we added Figure 4.30 with K-means $K = 40$, which is the same K as the baseline. The distribution of the size of the clusters is similar to Figure 4.31. However, as shown in Table 4.4, the EER for this specific K-means model, the difference of the EER from baseline is approximately 0.018. This is not the closest value. However, if we move the elbow point in the EER criteria in Figure 4.27 between $K = 40$ and $K = 110$, to $K = 60$, we are very close to the real value, as illustrated in Table 4.5. To further increase the accuracy, we could use a smaller step for the range 40–110 and then use the elbow method to find the optimal number of clusters K.

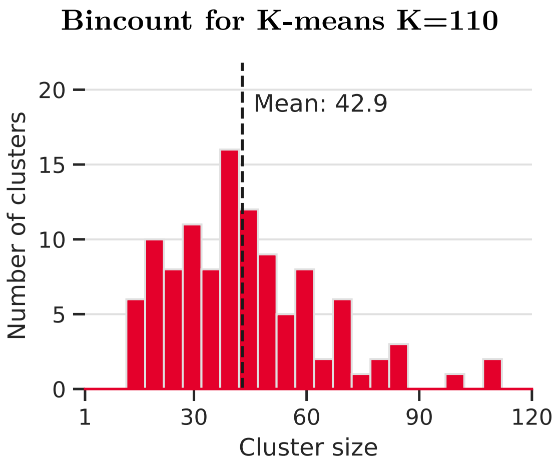


Figure 4.28: *Bincount of recordings for VoxCeleb1 with K-means and number of clusters $K = 110$ chosen by EER.*

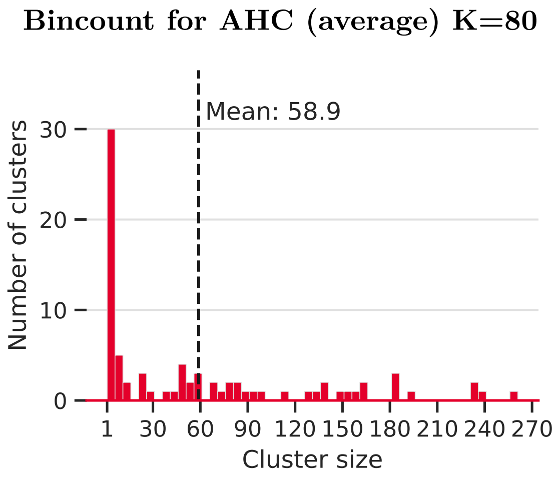


Figure 4.29: *Bincount of recordings for VoxCeleb1 with AHC (average) and number of clusters $K = 80$ chosen by EER.*

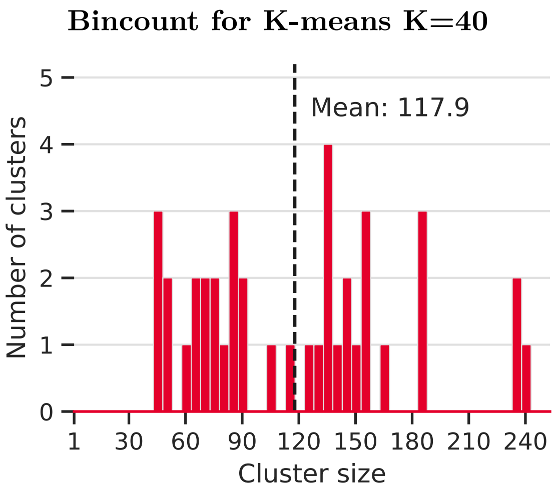


Figure 4.30: *Bincount of recordings for VoxCeleb1 with K-means and number of clusters $K = 40$ chosen by EER.*

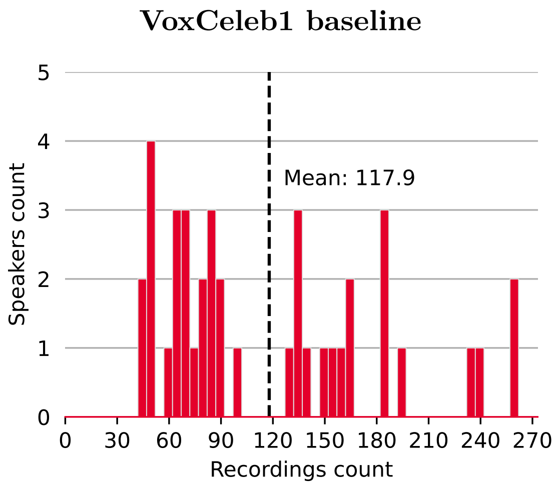


Figure 4.31: *Baseline for VoxCeleb1 with 40 speakers.*

Number of clusters K	40	50	**60**	70	80	90	100	110
EER	0.0568	0.0529	**0.0353**	0.0324	0.0321	0.0325	0.0295	0.0328

Table 4.5: *EER for Voxceleb1 dataset using K-means and cosine backend with different K from 40 to 110 showing steep change in EER from $K = 50$ to $K = 60$.*

Chapter 5

Conclusion

5.1 Summary

This work aimed to explore, implement, and test a technique for estimating results for speaker recognition on data without speaker labels. The speaker recognition system used in this thesis is based on the x-vector architecture. Nevertheless, the methods tested are *system-invariant*. The speaker recognition system processes audio data with an embedding extractor, creating a single discriminative embedding for each speech recording. The basic idea of this work is to create a speaker embedding for each recording in the dataset, then group them into a specific number of clusters K and evaluate the system with these cluster labels as if they were the reference.

Firstly, the so-called *pseudo-labels* have to be created by comparing each embedding and grouping them into different *clusters* representing these pseudo-labels. K-means, Gaussian mixture models (GMM) and agglomerative hierarchical clustering (AHC) were used for this purpose. The most challenging part is to find an optimal number of clusters K without knowing them beforehand. Four metrics were used to evaluate the clustering algorithms and estimate the correct K value – EER, Silhouette index, Calinski-Harabasz (CH), and Davies-Bouldin (DB) scores. These metrics have a peak or an elbow that indicates the optimal K. The Kneedle algorithm [3] was used to detect these points automatically.

Secondly, the pseudo-labels generated by the clustering algorithms were used to calculate Equal Error Rate (EER) – standard measure for comparison and evaluation of biometric systems. The EER is then compared to the baseline with reference speaker labels.

This system was developed and tuned on one part of the SITW database (dev-core-core) and tested on another part (eval-core-core). The best method achieved an estimate of 5.72 % EER with the reference EER equal to 5.15 % on SITW dev-core-core. Similar results were observed on the SITW eval-core-core, where the estimated EER is equal to 5.86 % and the reference 5.08 %. The difference between estimated and reference EER is 0.57 % for the SITW dev-core-core and 0.78 % for the SITW eval-core-core.

Two other tests on unseen datasets (NIST SRE16 and VoxCeleb1) were performed to verify the robustness of the proposed method. The estimated EERs were 2.40 % and 3.28 % for NIST SRE16 and VoxCeleb1, respectively. Reference EERs for these datasets are 2.8 % and 3.86 %, respectively.

Generally, the developed testing process had an estimated error of around ± 1 % in all test databases, which is an excellent result for an unsupervised technique. It is important to note that part of the NIST SRE16 (Tagalog) shows suspicious behavior not only for our experiments but also from the point of speaker recognition, and was reported by several sites.

Overall, the Silhouette value with the cosine distance metric is dominantly the best technique. The elbow method for EER would need some adjustments, as it overestimates the number of clusters K in every test case. DB and CH scores show some interesting results but are dataset-specific; thus, they can only be used in pairs with previous metrics to provide additional clues for an optimal estimate of K.

As the results show, the K-means clustering algorithm is the *best* one. AHC (single linkage) function suffer from the *chaining phenomena*, and the AHC (average linkage) function creates many *singleton clusters*. Generally, the estimation of correct EER and K was easier for datasets with speakers, where each of them has a roughly similar amount of audio recordings, and the dataset has overall good clustering properties without many outliers.

For the automatic control of audio quality and recording quality in the evaluation set, we propose to preselect representative recordings which are close to the production environment. The idea is that there are large amounts of unsupervised data, some of which are of poor quality. The recordings should have at least 3 seconds of *speech signal length* as used in production [1]. Another proposed metric for quality estimation is the *signal-to-noise ratio* (SNR), where the recordings should be close to the average SNR and do not contain many outliers. These two main metrics could be extended with *perceptual evaluation of speech quality* (PESQ) [33]. Furthermore, it is important to create a testing set with speakers who have more than one recording to create enough target trials because without them there will not be enough false alarms (FA) that are important for calculating the EER. This can be done by sampling the trials with higher scores.

5.2 Future work

For the future work, there are several ways to improve the effectiveness of the developed technique. The estimations with the EER and elbow method were slightly inaccurate, overestimating the number of clusters in each experiment. This could be adjusted with the system-specific elbow estimation method. Furthermore, it would be interesting to try different evaluation metrics, or another clustering algorithm, such as the X-means [4] or the G-means [5], which estimates the optimal K by itself.

Another unexplored area is how much noise and other artifacts negatively interfere with the unsupervised evaluation. To investigate this area, we propose experiments with adding noise to recordings and measuring the impact of different SNR values. The length of the speech signal is also an important aspect that influences speech recognition and should be tested to see how the results of unsupervised evaluation change with different lengths of speech.

Bibliography

[1] Phonexia. *Speaker identification (SID)*. Brno: [b.n.], 2019. Accessed: 2022-01-23. Available at: https://web.archive.org/web/20220123224559/https://partner.phonexia.com/kb/sp/speech-platform/spe/technologies-available-spe/speaker-identification-sid/.

[2] Snyder, D., Garcia Romero, D., Sell, G., Povey, D. and Khudanpur, S. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, p. 5329–5333. DOI: 10.1109/ICASSP.2018.8461375.

[3] Satopaa, V., Albrecht, J., Irwin, D. and Raghavan, B. Finding a „Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In: *2011 31st International Conference on Distributed Computing Systems Workshops*. 2011, p. 166–171. DOI: 10.1109/ICDCSW.2011.20. ISBN 978-1-4577-0384-3.

[4] Pelleg, D. and Moore, A. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. *Machine Learning, p.* january 2002.

[5] Hamerly, G. and Elkan, C. Learning the K in K-Means. *Advances in Neural Information Processing Systems*. march 2004, vol. 17.

[6] Phonexia. *Voice verify*. Brno: [b.n.], 2021. Accessed: 2021-12-04. Available at: https://web.archive.org/web/20211204200111/https://www.phonexia.com/en/product/voice-verify/.

[7] Matějka, P., Plchot, O., Glembek, O., Burget, L., Rohdin, A. J. et al. 13 years of speaker recognition research at BUT, with longitudinal analysis of NIST SRE. *Computer Speech and Language*. 2020, vol. 2020, no. 63, p. 1–15. DOI: 10.1016/j.csl.2019.101035. ISSN 0885-2308. Available at: https://www.fit.vut.cz/research/publication/12211.

[8] Beigi, H. *Fundamentals of Speaker Recognition*. December 2011. ISBN 978-0-387-77591-3.

[9] Moore, B. *An Introduction to the Psychology of Hearing: Sixth Edition*. Leiden, The Netherlands: Brill, 2013. ISBN 978-90-04-25242-4. Available at: https://brill.com/view/title/24210.

[10] Ravanelli, M. and Bengio, Y. *Speaker Recognition from Raw Waveform with SincNet*. arXiv, 2018. DOI: 10.48550/ARXIV.1808.00158. Available at: https://arxiv.org/abs/1808.00158.

[11] Tiwari, V. MFCC and its applications in speaker recognition. *International journal on emerging technologies.* Citeseer. 2010, vol. 1, no. 1, p. 19–22. ISSN 0975-8364.

[12] Stevens, S. S., Volkmann, J. and Newman, E. B. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America.* 1937, vol. 8, no. 3, p. 185–190. DOI: 10.1121/1.1915893. Available at: https://doi.org/10.1121/1.1915893.

[13] Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P. and Ouellet, P. Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing.* 2011, vol. 19, no. 4, p. 788–798. DOI: 10.1109/TASL.2010.2064307.

[14] LeCun, Y., Bengio, Y. and Hinton, G. Deep Learning. *Nature.* may 2015, vol. 521, p. 436–44. DOI: 10.1038/nature14539.

[15] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J. and Khudanpur, S. Recurrent neural network based language model. In:. January 2010, vol. 2, p. 1045–1048.

[16] Wang, Y., Deng, X., Pu, S. and Huang, Z. *Residual Convolutional CTC Networks for Automatic Speech Recognition.* arXiv, 2017. DOI: 10.48550/ARXIV.1702.07793. Available at: https://arxiv.org/abs/1702.07793.

[17] Joon Son Chung, A. Z. VoxCeleb2: Deep Speaker Recognition. In: *Interspeech 2018.* ISCA, Sep 2018. DOI: 10.21437/interspeech.2018-1929. Available at: https://doi.org/10.21437%2Finterspeech.2018-1929.

[18] Tchistiakova, S. Time Delay Neural Network. *Time Delay Neural Network Blog post.* Nov 2019. Accessed: 2021-01-29. Available at: https://kaleidoescape.github.io/tdnn/.

[19] Xie, W., Nagrani, A., Chung, J. S. and Zisserman, A. Utterance-level Aggregation for Speaker Recognition in the Wild. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2019, p. 5791–5795. DOI: 10.1109/ICASSP.2019.8683120.

[20] Snyder, D., Ghahremani, P., Povey, D., Garcia Romero, D., Carmiel, Y. et al. Deep neural network-based speaker embeddings for end-to-end speaker verification. In: *2016 IEEE Spoken Language Technology Workshop (SLT).* 2016, p. 165–170. DOI: 10.1109/SLT.2016.7846260.

[21] Yu, Y.-Q., Fan, L. and Li, W.-J. Ensemble Additive Margin Softmax for Speaker Verification. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2019, p. 6046–6050. DOI: 10.1109/ICASSP.2019.8683649.

[22] He, K., Zhang, X., Ren, S. and Sun, J. *Deep Residual Learning for Image Recognition.* arXiv, 2015. DOI: 10.48550/ARXIV.1512.03385. Available at: https://arxiv.org/abs/1512.03385.

[23] Hastie, T., Tibshirani, R. and Friedman, J. *The Elements of Statistical Learning.* Secondth ed. New York, NY, USA: Springer New York Inc., 2009. Springer Series in Statistics. ISBN 978-0387848570.

[24] Li, S. Z. and Jain, A., ed. *Fisher Criterion.* Boston, MA: Springer US, 2009. 549–549 p. ISBN 978-0-387-73003-5. Available at: https://doi.org/10.1007/978-0-387-73003-5_585.

[25] Tharwat, A., Gaber, T., Ibrahim, A. and Hassanien, A. E. Linear discriminant analysis: A detailed tutorial. *Ai Communications.* may 2017, vol. 30, p. 169–190,. DOI: 10.3233/AIC-170729.

[26] Gareth, J., Daniela, W., Trevor, H. and Robert, T. *An introduction to statistical learning: with applications in R.* Spinger, 2013. ISBN 978-1461471370.

[27] Ioffe, S. Probabilistic Linear Discriminant Analysis. In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part IV.* Berlin, Heidelberg: Springer-Verlag, 2006, p. 531–542. ECCV'06. DOI: 10.1007/11744085_41. ISBN 3540338381. Available at: https://doi.org/10.1007/11744085_41.

[28] Ramoji, S., Krishnan, P., Singh, P. and Ganapathy, S. *Pairwise Discriminative Neural PLDA for Speaker Verification.* arXiv, 2020. DOI: 10.48550/ARXIV.2001.07034. Available at: https://arxiv.org/abs/2001.07034.

[29] Kenny, P., Stafylakis, T., Ouellet, P., Alam, M. J. and Dumouchel, P. PLDA for speaker verification with utterances of arbitrary duration. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing.* 2013, p. 7649–7653. DOI: 10.1109/ICASSP.2013.6639151.

[30] Garcia Romero, D., Zhou, X. and Espy Wilson, C. Y. Multicondition training of Gaussian PLDA models in i-vector space for noise and reverberation robust speaker recognition. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2012, p. 4257–4260. DOI: 10.1109/ICASSP.2012.6288859.

[31] Vogt, R., Sridharan, S. and Mason, M. Making Confident Speaker Verification Decisions With Minimal Speech. *IEEE Transactions on Audio, Speech, and Language Processing.* 2010, vol. 18, no. 6, p. 1182–1192. DOI: 10.1109/TASL.2009.2031505.

[32] Plapous, C., Marro, C. and Scalart, P. Improved Signal-to-Noise Ratio Estimation for Speech Enhancement. *IEEE Transactions on Audio, Speech, and Language Processing.* 2006, vol. 14, no. 6, p. 2098–2108. DOI: 10.1109/TASL.2006.872621.

[33] Rix, A., Beerends, J., Hollier, M. and Hekstra, A. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221).* 2001, vol. 2, p. 749–752 vol.2. DOI: 10.1109/ICASSP.2001.941023.

[34] *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs.* Standard ITU-T Recommendation P.862. International Telecommunication Union, 2001. Accessed: 2022-02-15. Available at: https://www.itu.int/rec/T-REC-P.862.

[35] Hughes, G. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory.* 1968, vol. 14, no. 1, p. 55–63. DOI: 10.1109/TIT.1968.1054102.

[36] Gorban, A. N. and Tyukin, I. Y. Blessing of dimensionality: mathematical foundations of the statistical physics of data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences.* 2018, vol. 376, no. 2118, p. 20170237. DOI: 10.1098/rsta.2017.0237. Available at: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2017.0237.

[37] Abdi, H. and Williams, L. Principal Component Analysis. *Wiley Interdisciplinary Reviews: Computational Statistics.* july 2010, vol. 2, p. 433 – 459. DOI: 10.1002/wics.101.

[38] Ross, D., Lim, J., Lin, R.-S. and Yang, M.-H. Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision.* may 2008, vol. 77, p. 125–141. DOI: 10.1007/s11263-007-0075-7.

[39] Wang, S., Sun, Y. and Bao, Z. On the Efficiency of K-Means Clustering: Evaluation, Optimization, and Algorithm Selection. *Proc. VLDB Endow.* VLDB Endowment. oct 2020, vol. 14, no. 2, p. 163–175. DOI: 10.14778/3425879.3425887. ISSN 2150-8097. Available at: https://doi.org/10.14778/3425879.3425887.

[40] Hamerly, G. and Elkan, C. Alternatives to the k-means algorithm that find better clusterings. In:. January 2002, p. 600–607. DOI: 10.1145/584792.584890.

[41] Arthur, D. and Vassilvitskii, S. K-Means++: The Advantages of Careful Seeding. In:. January 2007, vol. 8, p. 1027–1035. DOI: 10.1145/1283383.1283494.

[42] Aurélien, G. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* 2nd ed. O'Reilly Media, Inc., 2019. ISBN 978-1492032649.

[43] Sculley, D. Web-scale k-means clustering. In:. January 2010, p. 1177–1178. DOI: 10.1145/1772690.1772862.

[44] Cheng, J.-M. and Wang, H.-C. A method of estimating the equal error rate for automatic speaker verification. In: *2004 International Symposium on Chinese Spoken Language Processing.* 2004, p. 285–288. DOI: 10.1109/CHINSL.2004.1409642.

[45] Liu, F. and Deng, Y. Determine the Number of Unknown Targets in Open World Based on Elbow Method. *IEEE Transactions on Fuzzy Systems.* 2021, vol. 29, no. 5, p. 986–995. DOI: 10.1109/TFUZZ.2020.2966182.

[46] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics.* 1987, vol. 20, p. 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. ISSN 0377-0427. Available at: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[47] Halkidi, M., Batistakis, Y. and Vazirgiannis, M. On Clustering Validation Techniques. *Journal of Intelligent Information Systems.* october 2001, vol. 17. DOI: 10.1023/A:1012801612483.

[48] Caliński, T. and JA, H. A Dendrite Method for Cluster Analysis. *Communications in Statistics - Theory and Methods*. january 1974, vol. 3, p. 1–27. DOI: 10.1080/03610927408827101.

[49] Davies, D. L. and Bouldin, D. W. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1979, PAMI-1, no. 2, p. 224–227. DOI: 10.1109/TPAMI.1979.4766909.

[50] McLaren, M., Ferrer, L., Castan, D. and Lawson, A. The Speakers in the Wild (SITW) Speaker Recognition Database. In: *Proc. Interspeech 2016*. 2016, p. 818–822. DOI: 10.21437/Interspeech.2016-1129.

[51] Seyed, Kheyrkhah, T., Tong, A., Greenberg, C., Olson, D. et al. The 2016 NIST Speaker Recognition Evaluation. In:. Interspeech 2017, Stockholm, -1, 2017-08-20 2017. Available at: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=922849.

[52] Nagrani, A., Chung, J. S., Xie, W. and Zisserman, A. Voxceleb: Large-scale speaker verification in the wild. *Computer Speech and Language*. 2020, vol. 60, p. 101027. DOI: https://doi.org/10.1016/j.csl.2019.101027. ISSN 0885-2308. Available at: https://www.sciencedirect.com/science/article/pii/S0885230819302712.

[53] Sadjadi, O. *NIST SRE CTS Superset: A large-scale dataset for telephony speaker recognition*. NIST SRE website, 2021-08-16 04:08:00 2021. Accessed: 2022-05-05. Available at: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=933116.

[54] Godfrey, J. J., Holliman, E. C. and McDaniel, J. SWITCHBOARD: Telephone Speech Corpus for Research and Development. In: *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*. USA: IEEE Computer Society, 1992, p. 517–520. ICASSP'92. ISBN 0780305329.

Appendix A

Used Software and Libraries

Python v3.9.12 was used with these libraries and their corresponding versions:

- sklearn v1.0.2
- numpy v1.20.3
- scipy v1.7.2
- pandas v1.3.4
- numba v0.54.1
- kneed v0.7.0
- gap-stat v2.0.1
- matplotlib v3.4.3
- seaborn v0.11.2
- jupyter v1.0.0