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

EVALUATION AND OPTIMIZATION OF COMPUTATIONAL COSTS IN SPEAKER RECOGNITION SYSTEMS

EVALUACE A OPTIMALIZACE VÝPOČETNÍ NÁROČNOSTI V SYSTÉMECH PRO ROZPOZNÁNÍ ŘEČNÍKA

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **SABÍNA GREGUŠOVÁ**
AUTOR PRÁCE

SUPERVISOR **Dr. JOHAN ROHDIN**
VEDOUCÍ PRÁCE

BRNO 2020

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2019/2020

Bachelor's Thesis Specification

23008

Student:	**Gregušová Sabína**
Programme:	Information Technology
Title:	**Evaluation and Optimization of Computational Costs in Speaker Recognition Systems**
Category:	Speech and Natural Language Processing

Assignment:

1. Design an evaluation metric and/or protocol that, in addition to accuracy of speaker verification systems, considers computational costs. Ideally by discussing with companies.
2. Evaluate some existing speaker verification systems with this new metric. See how the ranking of the systems changes when the computational cost is considered.
3. Develop a new system that is optimized for the new metric. For example by:
   * Tune the architecture of existing models (number or layers in neural networks etc.)
   * Develop heuristics for not processing the whole parts of long utterances (select which parts to process in a clever way)
   * Develop heuristics for processing easy data with a "light" system and hard data with a "heavy" system.
4. Compare the proposed approaches.

Recommended literature:

- Tomi Kinnunen and Haizhou Li, *An overview of text-independent speaker recognition: From features to supervectors*, Speech Communication 2009
- Pavel Matějka et al.*, Brno University of Technology in NIST SREs ---Story and longitudinal analysis,* draft in preparation 2019
- David Snyder et al., *Deep neural networks embeddings for text-independent speaker recognition,* in Interspeech 2017
- *David Snyder et al. x-vectors: Robust DNN embeddings for speaker recognition,* in ICASSP 2018,

Requirements for the first semester:

- Finish item 1 above and initial results on item 2.

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

Supervisor:	**Rohdin Johan A., Dr.**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2019
Submission deadline:	May 14, 2020
Approval date:	November 1, 2019

Abstract

The goal of this thesis is to propose an evaluation metric that includes computational costs. Computational costs generally do not pose a problem in research, but it can become problematic in a commercial production system, where speed is essential. The proposed metric extends existing evaluation framework from NIST and adds parameter for time unit and time unit cost. These metrics are applied on real ASV and experiments show the potential for further research and possible use. The experiments focus on reducing the computational cost by posing a limit on maximum length of the utterance, but also limiting number of frames for x-vector extraction. Both optimizations reduced the computational costs and reached favorable results for the new metrics. Finally, experiments' results are compared and each system modification is ranked according to the new metrics.

Abstrakt

Cieľom tejto práce je navrhnúť hodnotiacu metriku, ktorá zahŕňa výpočetné náklady. Všeobecne výpočetné náklady nepredstavujú vo výskume problém, ale môžu byť problematické v komerčnom produkčnom systéme, kedy je rýchlosť dôležitá. Navrhnuté metriky rozširujú existujúci rámec pre hodnotenie od NIST a pridávajú k nim parametre pre časovú jednotku a náklady pre časovú jednotku. Tieto metriky sú aplikované na skutočný ASV a experimenty ukazujú potenciál pre hlbší výskum a možné použitie. Vrámci experimentov bola limitovaná maximálna dĺžka nahrávok, ale aj maximálne dĺžka rámcov pre spracovanie pri extrakcii x-vektorov. Obe optimalizácie znížili celkové výpočetné náklady a dosiahli priaznivé výsledky pre nové metriky. Na záver sú výsledky z experimentov porovnané a jednotlivé modifikácie ohodnotené a zoradené podľa nových metrík.

Keywords

speaker recognition systems, speaker recognition evaluation, speaker recognition optimization, computational costs, evaluation metrics

Kľúčové slová

rozpoznávanie rečníka, evaluácia systémov na rospoznávanie rečníka, optimalizácia systémov na rozpoznanie rečníka, výpočetné náklady, evaluačné metriky

Reference

GREGUŠOVÁ, Sabína. *Evaluation and Optimization of Computational Costs in Speaker Recognition Systems*. Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Dr. Johan Rohdin

Rozšírený abstrakt

V posledných rokoch sa zvýšil záujem o možnosť použitia systémov na rozpoznávanie rečníka aj v proprietárnych (komerčných) softvéroch. Hlavné uplatnenie takýchto systémov je v dnešnej dobe najmä vo forenznej vede, vďaka čomu je možné potvrdiť alebo vyvrátiť identitu poodzrivého práve na základe zvukovej nahrávky získanej počas vyšetrovania. Momentálne sa do popredia dostáva myšlienka využitia týchto systémov aj vo sfére bezpečnosti a autentizácie. Je však potrebné takýto systém zaistiť od možnosti zneužitia inou osobou, ktorá by aktívne menila svoj hlas, aby napodobila niekoho iného, alebo využila nahrávku danej osoby pre falošnú autentizáciu.

Každý systém pre rozpoznávanie rečníka má svoje nedostatky a preto existujú prípady, kedy by podľahol takémuto útoku. Preto je potrebné skúmať takéto situácie a ich následky v komerčnej sfére. Systémy na rozpoznávanie rečníka predstavujú *binárny problém*. Systém porovná hlas hovoriaceho (vstup) s nahrávkou hovoriaceho cieľovej identity, ktorá je prijatá (*enrolled*) v danom systéme. Rozhodnutie systému je teda potvrdenie (*accept*) alebo vyvrátenie (*reject*) tohto hovoriaceho. Z toho vyplýva, že systém môže urobiť 2 možné chyby, a to potvrdiť indetitu podvodníka (*false accept*), alebo vyrátiť identitu skutočného hovoriaceho (*false reject*).

Práve v komerčnej sfére je potrebné priradiť týmto chybám určitú peňažnú hodnotu, "cenu". Rozhodovacia hranica bude vždy kompromisom medzi tým, čo je pre nás dôležitejšie, napr. pre banku je oveľa dôležitejšie vyvrátiť identitu podvodníka, ktorý by mohol ďalej vykonávať transakcie v mene niekoho iného, ako to, že niektorý zákazník bude otrávaný z toho, že ho nevedia identifikovať. Priradenie takejto "ceny" je však problematické, pretože sú to situácie, ktorým sa ťažko umelo priraďuje cena. Pre banku je určite výhodnejšie, aby odhalila podvodníka, no môže sa stať, že bohatý zákazník bude otrávený z toho, že jeho autorizácia je zdĺhavá a rozhodne sa presunúť svoje peniaze do inej banky, ktorá má lepší systém. Preto je potrebné hľadať balans medzi takýmit situáciami a nepribližovať sa do extrémov preferovania jedného rozhodnutia nad druhým.

Práve rozhodnutie systému v týchto okrajových situáciach slúži ako vhodný ukazateľ pre porovnanie výkonu jednotlivých systémov. Okrem prideľovania peňažnej hodnoty dôsledku týchto rozhodnutí je taktiež potrebné prihliadať aj na výpočetný čas. Vo výskume nie je až tak potrebné brať na čas ohľad, pretože cieľom je dospieť k najlepšiemu možnému riešeniu.V komerčnej oblasti je ale aj čas rozhodujúcim faktorom, a preto je potrebné aj ten zakomponovať do celkového hodnotenia takéhoto systému.

V tejto práci sú vytvorené nové metriky, ktoré vo svojich výpočtoch zahŕňajú aj časový údaj. Nové metriky vychádzajú z existujúcich rámcov pre hodnotenie a porovnanie systémov a rozširujú ich o čas. Navrhnuté metriky sú použité v experimentoch na reálnych systémoch. Jedná sa o nástroj Kaldi, ktorý je voľne dostupný pre vedecké účely. Pri experimentoch boli využité dáta z databáze VoxCeleb, ktorá obsahuje obrovskú škálu audio súborov celebrít voľne dostupných na internete. Cieľom experimentov je simulovať stav systému v reálnych aplikáciach, preto je použitý predtrénovaný model a na vyhodnotenie PLDA.

Meraný čas predstavuje iba čas, ktorý program strávi v CPU (nezahŕňa čas potrebný na ostatné operácie, ako na vstup/výstup, alebo čas, kedy bol proces blokovaný iným procesom). Pre objektívne hodnotenie sú rovnaké experimenty vykonané viackrát a finálny časový údaj je priemerom týchto experimentov. Následne sú aplikované navrhnuté metriky pre porovnanie systémov.

Cieľom optimalizácie bolo upraviť existujúci systém tak, aby pre nové metriky dosahoval čo najlepšie výsledky. Zároveň sa pri celkovom hodnotení systémov zohľadnovali aj pôvodné metriky, aby bol nájdený vhodný systém, ktorý produkuje čo najviac správnych výsledkov

a zároveň má znížené výpočetné náklady. Jednou z možností bolo skrátiť dĺžku vstupných súborov (výpovede rečníka). Takáto modifikácia prirodzene vedie na zníženie výpočetných nákladov. Výsledkom tohto experimentu bolo ohodnotenie systému s rôznou maximálnou dĺžkou vstupov.

Druhý experiment sa zameriava na skracovanie počtu rámcov pre x-vektor extrakciu. Keďže sa jedná o výpočetne najnáročnejšiu časť skriptu, je to vhodný objekt optimalizácie. Tieto experimenty úspešne zredukovali celkový výpočetný čas a zachovali dobré ohodnotenie výkonu aj pomocou pôvodných metrík.

Na záver sú oba experimenty porovnané a systémy sú ohodnotené a zoradené podľa výsledkov nových metrík. Druhý experiment mal konzistentne lepšie výsledky – priemerný čas spracovania bol nižší a aj korektnosť výsledkov bola lepšia. Vykonané experimenty dokazujú, že je možné systém optimalizovať tak, aby dosahoval lepší čas a stále produkoval dobré výsledky.

Evaluation and Optimization of Computational Costs in Speaker Recognition Systems

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Dr. Johan Rohdin. All the relevant information sources used for this thesis have been properly cited and included in the bibliography.

. . . .
Sabína Gregušová
May 28, 2020

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Johan Rohdin, for all his valuable advice, guidance and patience with my work. I would also like to thank my friends, family, and my great boyfriend, Filip, for all their great support and patience throughout this difficult process.

Contents

Chapter 1

Introduction

In recent decades, there has been a tremendous spike in the technological field, which was accompanied by deeper interest in using machine learning and classification in practical applications.

The research on speaker recognition systems involves the study of many disciplines, such as *signal processing*, *machine learning* etc. In order to evaluate, access and compare performance of different systems, there has been an continuous effort to develop standardized evaluation methodology.

The goal of this thesis was to conduct a research on existing evaluation and optimization techniques of speaker recognition technology, look into the possibility of taking the computational costs into account, the possibility of its improvement and a development of new metrics for improved assessment of its performance and potential future use. The organization of this thesis is as follows.

Chapter 2 introduces the basic state-of-the-art systems and its fundamental theoretical knowledge.

Chapter 3 presents the existing evaluation frameworks and standards, its mathematics and proposes new protocols and metrics used in the experiments. It is based on the idea that practical application of speaker recognition technology often requires not only accuracy, but speed as well. Therefore, these metrics are developed with the intention of improving the ranking of the systems with the inclusion of processing costs as well.

Chapter 4 applies the proposed metrics on a toolkit Kaldi, which is used for research on speaker recognition licensed under collaborative open source software development. The experiments show, how the inclusion of time essentially changes the evaluation process. It also focuses on developing a new system that is optimized for the new metrics.

Chapter 5 concludes the thesis by presenting the obtained results of different approaches and comparing the proposed concept with the pre-existing evaluation framework.

Chapter 2

Speaker recognition systems

The primary goal of speaker recognition systems is to recognize a speaker from their voice. Each speaker has their unique characteristics, which include the physical features, manner of speaking, choice of vocabulary, accent etc. It is important to realize, that although very close in fields, *speaker recognition* and *speech recognition* have different goals, modeling strategies and practical applications.

There is a wide range of practical uses for these systems. *Forensics* often utilizes speaker recognition systems in order to confirm or deny the identity of a suspected criminal in front of the court from the recordings obtained as evidence (e.g. phone conversation). Voice and speech, as a bio-metric feature, also has potential to be used for *authentication* and *security*. Such systems, however, must be capable of clearly distinguishing between a true voice of a speaker and speaker, who is trying to mimic and impersonate somebody else by consciously changing their high-level voice features. The system also cannot be susceptible to accepting recordings, as this could also lead to breach in the safety and security.

Currently, it is not a good strategy to purely rely on the speaker recognition systems, but it can be used to enhance an already existing security system as another supporting form of bio-metric authentication among others.

2.1 System types and classification

We generally distinguish the speaker recognition systems into 3 separate types according to its goals as:

- **Diarisation:** successfully segment the input signal according to speaker's identity; the goal of this segmentation is to correctly determine „who spoke when“
- **Verification:** either to confirm or refute the identity claim of a speaker by comparing the speaker's utterances with the utterances of the speaker whose identity is being claimed; the target identity template must be already enrolled (1:1 match)
- **Identification:** speaker's utterance is compared with multiple enrolled utterances; in order to correctly verify the speaker's identity, their speaker model must be already enrolled (1:n match)

The speakers' utterances for training and testing can be recorded in various situations, but we can consider two types of speakers:

- **Cooperative speaker:** speakers, who wish to be recognized; speakers are willing to repeat the utterances and speak more clearly if needed, they are often requested to use pre-determined phrases
- **Non-cooperative speaker:** speakers, who do not wish to be recognized; this type of recognition systems is considered to be particularly challenging, as the utterances are often recorded in a more hostile environment, thus making the speaker model more complex

There are also 2 types of speaker recognition systems, which are closely bounded to the types of speech we use in the utterances:

- **Text-dependent:** the speakers have limited word/phrases they can use; oftentimes, there are restricted words (lexicons) that are known beforehand, so the system has more data and information about the possible utterance, it generally performs better than it's counterpart; these utterances are most commonly performed by a *cooperative speaker*
- **Text-independent:** the speakers can freely use any words/phrases; the testing data and production data can be completely different, the system must be able to deal with various manners of speaking, for example in *forensics*, the speaker is usually *non-cooperative* and the conditions of such audio is harsher

It is possible to deduce many potential applications of such technology if we look at this classification. The speaker recognition systems usually tend to be more focused on processing audio from telephony-based conversations, which is generally speaking *text-independent* with *non-cooperative* speakers. Further explanation and examples are discussed in [9], [7] or [6].

2.2 Architecture of speaker recognition systems

The speaker recognition system consists of 3 phases, *development phase*, *enrollment phase* and *verification phase/evaluation phase.* The main goal of the *enrollment phase* is to extract the speaker-specific features and create a speaker model. *Development phase* refers to learning/estimating parameters from scratch (often also called *training phase*). Lastly, *Evaluation phase* is used for evaluating the created model. The term *adaptation* is used for slightly tuning the existing parameters of an existing model. The same data should not be used in multiple phases, as it jeopardizes the system and distorts the results of performance evaluation [14].

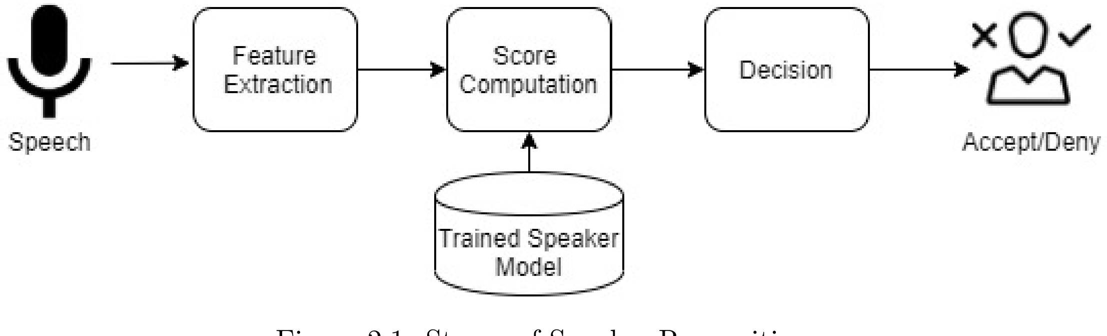

Figure 2.1: Stages of Speaker Recognition

Even when the same speaker utters the same sentences, the extracted features can still slightly vary for each utterance. Therefore, our goal is to create a model, that captures the main characteristics, while it is still capable of dealing with a feature variation. The features are divided into various categories, from high-level features (learned) to short-term spectral and voice source features (physiological). There is no feature category, that is considered the best – the choice depends on computational resources, complexity and practicality. Whatever feature is chosen for the system, there will always be a trade-off [9].

The aim of **feature extraction** is to convert a complex speech waveform to reasonable number of parameters, that can be used for further processing and analysis of the signal. In speaker recognition, we typically want to remove the noise (pre-processing of the data) and only extract the important features of the speech. One of the most widely used approaches in speaker recognition systems are the **mel-frequency cepstrum coefficients (MFCC)**. The MFCC was inspired by the function of human ear. The function rises linearly up to 1000 Hz, but logarithmically above that. The signal is firstly segmented into frames of about 20-30 ms, windowed by a window function that boosts the higher frequencies. Fast Fourier Transform (FFT) is applied to find the frequency components – we transform the time domain speech signal into spectral domain signal. The log of the frequency components is taken, because it is easier to do summation instead of multiplication. Finally, the Discrete cosine transform is applied to get the MFCC. The MFCC are the amplitudes of that spectrum [7].

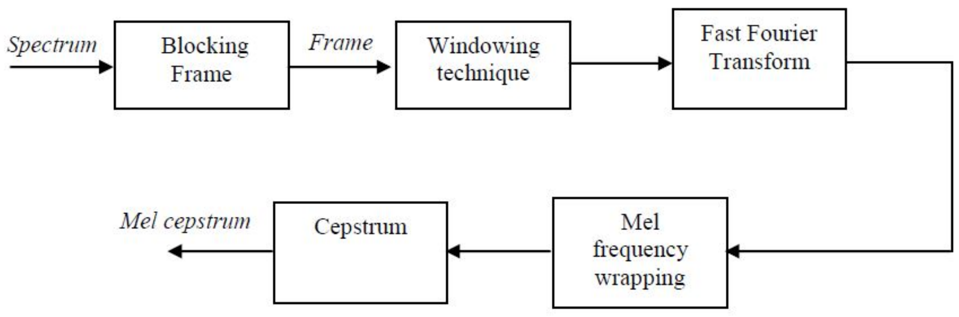

Figure 2.2: MFCC process of feature extraction (from [2])

For *enrollment* of the speaker, it is sufficient to have around 30 seconds of audio per speaker. In the *development (training) phase*, the chosen model is trained with the *training data*. It is essentially trained with a large data-set, which ideally represents the real data distribution. *Training data* are required to contain large number of speakers with multiple recordings per speaker. The model must be then verified with *evaluation data*, which cannot be the same as the *training data*. The choice of the model is based on the system objectives, because each model has its advantages and disadvantages. Models, according to how they are trained, are divided into:

- **Generative models:** the goal is to find the feature distribution within each speaker (GMM, VQ)
- **Discriminative models:** the goal is to find the boundary between the speakers (SVM, NN)

Each speaker is characterized by a speaker model and the most common models are introduced later in this chapter. There are various models and their alternations. Although there has been a recent trend of shifting the focus from the traditional models to neural networks, it is essential to be familiar with the older models as they provide a good basis for understanding the state-of-the-art speaker recognition systems.

In *verification*, the speaker's voice is compared against the speaker model of the claimed identity; whilst in *identification*, the speaker' s voice is compared to all the enrolled speaker models and the best one is selected.

While technologies used in these sub tasks are essentially the same, the thesis focuses on verification for 3 reasons:

- The U.S. National Institute of Standards and Technology (NIST) has conducted several speaker verification evaluations, which means large quantities of data as well as standardized experimental protocols for this task are available.
- Automatic Speaker Verification (ASV) has a large number of applications. In particular within access control, surveillance and forensics. In access control, ASV is used to authorize access to a resource such as a bank account or a building. In surveillance applications, it is used for detecting a wanted criminal in a collection of telephone recordings. In forensics, ASV is used for comparing a voice recording from a crime scene with the voice of a suspect or a victim.
- The modeling of the "claimed identity is false" case poses some additional difficulties over speaker identification because in this case, the utterance could have been spoken by a person that the system has no reference data for

The final decision of the recognition systems is based upon the distance from the reference speaker model, i.e the matching score. Ultimately, it is a binary decision, either to **accept** or **deny** the claimed identity of a speaker.

There is a threshold that determines, what distance is considered as acceptable for finding a match for the speaker (**target trial**). Any distance below the threshold is then considered to be an impostor (**non-target trial**). Real-life data will be overlapping at certain points [6]. In practice, the threshold is set by the evaluation script in the *evaluation phase*. The metrics used for evaluation, such as Equal Error Rate (EER) and Decision Cost Function (DCF), are further discussed in Chapter 3.

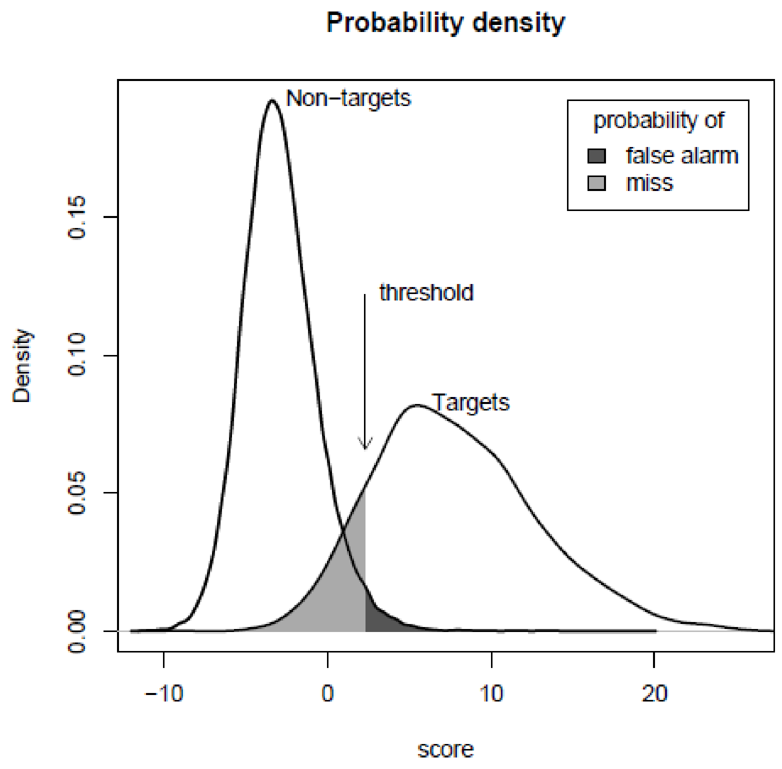

Figure 2.3: Visualization of scores for targets and non-targets and the decision threshold (from [10])

2.3 Gaussian mixture model (GMM)

A probabilistic *generative* model which is often used as a reference model for speaker recognition. It relies on the fact that its means depends on the speaker's identity and the channel effects. The concatenated means are referred to as super-vector. Ideally, we want to extract the features, but remove the channel effect on the data [11].

GMM is composed of a finite mixture of multivariate Gaussian components. Let $\mathcal{N}(x; \mu_c, \Sigma_c)$ be the probability density function (PDF) for a multivariate Gaussian distribution. The probability density function (PDF) of a multivariate Gaussian mixture model is then given as [3]:

$$P(x|\lambda) = \sum_{c=1}^{C} \mathcal{N}(x; \mu_c, \Sigma_c) P_c \tag{2.1}$$

$$\mathcal{N}(x; \mu_c, \Sigma_c) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)} \tag{2.2}$$

$$\lambda = \{P_c, \mu_c, \Sigma_c\} \tag{2.3}$$

$$\sum_{c=1}^{C} = 1 \tag{2.4}$$

where C is the number of Gaussian components in the mixture, μ_c is the mean vector and Σ_c is the co-variance matrix of the c-th Gaussian component; P_c is the *mixing weight*, i e.

the *a priori probability* for the c-th Gaussian component. Usually, we assume that feature vectors are independent, so the GMM likelihood is computed as:

$$P(x|\lambda) = \prod_{c=1}^{C} \mathcal{N}(x; \mu_c, \Sigma_c) \tag{2.5}$$

$$\log P(x|\lambda) = \sum_{c=1}^{C} \log \mathcal{N}(x; \mu_c, \Sigma_c)) \tag{2.6}$$

Training of GMM consist of estimating the $\lambda = \{P_c, \mu_c, \Sigma_c\}$ for each component from the training data. Most commonly used algorithm is *Maximum-likelihood (ML)*. It looks for the model parameters by maximizing the likelihood of GMM from the training data. The ML parameters can also be estimated from the iterative *expectation-maximization (EM)* algorithm, although its likelihood increases with every iteration, it has a slow convergence rate and only converges to local optimal points [9].

Since the process of text-independent speaker verification is a difficult task, GMM can be altered by adding some functions that improve its performance. One example is extending the GMM with one model that represents all the speakers with general, speaker-independent feature characteristics. This model is referred to as **Universal Background model (UBM)**. Such approach has shown the best results compared to the other adaptations [14].

The main disadvantage of using the GMM is that it cannot handle high-dimensional data very well, and with the number of features, the number of training data grows exponentially, so-called „the curse of dimensionality“ [9].

2.4 I-vector, x-vector and PLDA

The i-vector model, firstly introduced by [4], stems from GMM UBM in Section 2.3 and simplifies it. It has been discovered, that the channel factors also contain some information about the speaker. Typically, (probabilistic linear discriminant analysis) PLDA is used as a backend model for removing channel effects and comparing two *i-vectors* or *x-vectors*. The model $\mathbf{M}$ is then represented as follows:

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\mathbf{w} \tag{2.7}$$

where $\mathbf{m}$ is the global mean of UBM super-vector, that is speaker and channel independent. $\mathbf{T}$ is a model parameter (i-vector extractor), $\mathbf{w}$ is a variable with standard normal distribution, and its MAP estimate is called the *i-vector*. The PLDA does the comparison between *i-vectors* or *x-vectors* to obtain the verification score. It assumes the distribution of *x-vectors*, ϕ as [11]:

$$\phi = \overline{\phi} + \mathbf{V}\mathbf{y} + e \tag{2.8}$$

In this distribution, $\overline{\phi}$ is the global mean of observed data, $\mathbf{y}$ being a latent speaker variable with normal prior probability, and $\mathbf{e}$ is a latent channel variable with prior probability of $\mathcal{N}(0, \mathbf{W})$. *I-vectors* are usually pre-processed by centering and are projected by linear discriminant analysis (LDA), that reduces the dimensionality. The representations are then length-normalized and modeled by PLDA. The scores are normalized using adaptive s-norm [16]. *X-vector* shares the same concept as presented here but it is based on neural networks and usually outperforms *i-vectors*.

Huge advantage of using i-vectors or x-vectors is the fact that it maps variable-length utterances to fixed-dimensional embeddings. X-vectors often benefits from data augmentation by increasing the amount of training data, which improves robustness of the whole system. Because of that, x-vectors tend to achieve better performance on the evaluation data-set. X-vectors are extracted from Deep Neural Networks (DNN) introduced in Section 2.5. DNN uses supervised training, and that's why it successfully exploits the data augmentation [16].

2.5 Neural Networks

Neural networks have been playing important role in the speaker recognition technology for the past years. It is beneficial to use Neural Networks, as there is no exact function, that maps speaker-specific features to the speaker identity.

The main 2 kinds of neural networks that have been considered for use in the speaker recognition technology is the *multi-layer perceptron (MLP)* and *neural tree network (NTN)*.

Multi-layer perceptron (MLP) is a neural network classifier, usually consisting of one input layer, one hidden layer and one output layer, which is trained with back-propagation. It is beneficial to use MLP for problems with only limited information about the characteristics of the input. Each addition or deletion of nodes or hidden layers results in different solution of the problem and the optimal solution is usually found by trial and error, which is not very efficient. MLP is a subset of Deep Neural Netowrks (DNN), often referred to only as Neural Networks, which is the most recent trend in speaker recognition technology. The main difference between MLP and DNN is that MLP is always feed-forward, while general DNN can have loops [5], [16].

Neural tree network (NTN) is hierarchical hybrid classifier, that combines feed-forward neural networks (neural network where nodes does not form a cycle) and a binary decision tree. The decision tree then consists of single-layer perceptrons (SLP) and child nodes. The NTN is trained for a target speaker - it iteratively labels nodes with either 0 (non-target speaker) and 1 (target speaker) respectively until all leaf nodes are labeled [8]. The resulting NTN has a 100% performance on the training set, but may not achieve optimal generalization for other data-sets [5].

Chapter 3

Evaluation of speaker recognition systems

Being able to evaluate the performance of various algorithms is essential to ensure meaningful comparison. In the past years, there has been an effort to standardize the evaluation methodology used in speaker verification systems. The National Institute of Standards and Technology (NIST) has provided a standardized evaluation framework in order to correctly access and compare text-independent speaker recognition systems.

Since 1996, NIST has been conducting Speaker Recognition Evaluations (SRE) to measure the performance of current state-of-the-art text-independent speaker recognition systems. These evaluations provide objective comparison and feedback on currently used speaker recognition technology. NIST provides the corpora for training and testing the model [15].

3.1 Verification errors

Let there be a test utterance and enrollment utterance in the trial that are from the same speaker, then we denote it as a **target trial**. If the test utterance does not come from the same speaker, we denote it as **non-target trial**. Since the goal of speaker verification is to determine whether the two utterances come from the same speaker or not, there are two possible errors that can occur:

- **False accept (FA)**: system determines that two recordings from different speakers come from the same speaker, therefore classifies non-target trial as a target trial; it is also referred to as *false positive*
- **False reject (FR)**: system determines that two recordings from the same speaker come from different speakers, therefore classifies target trial as a non-target trial; it is also referred to as *miss* or *false negative*

Correct Result	**System Result**	
	Target	**Non-Target**
Target	True Accept	False Reject
Non-target	False Accept	True Reject

Table 3.1: Possible outcomes from comparing the system result with the correct results

The rate of these errors is measured and calculated from the evaluation data-set. Two different error rates complicates the evaluation process, since it is hard to compare the performance of different systems.

3.2 Detection Error Trade-off (DET)

One of the visual tools for speaker evaluation is undoubtedly the detection error trade-off (DET). The resulting graph of DET is capable of showing the system performance at many different operating points. NIST uses DET for plotting the performance of various systems over many operating points. The system that produces the leftmost DET curve is deemed to have the best performance. The weakness of using DET is, however, that they do not provide a single scalar value that could be used for comparison. Therefore, it cannot be used directly for evaluation [9].

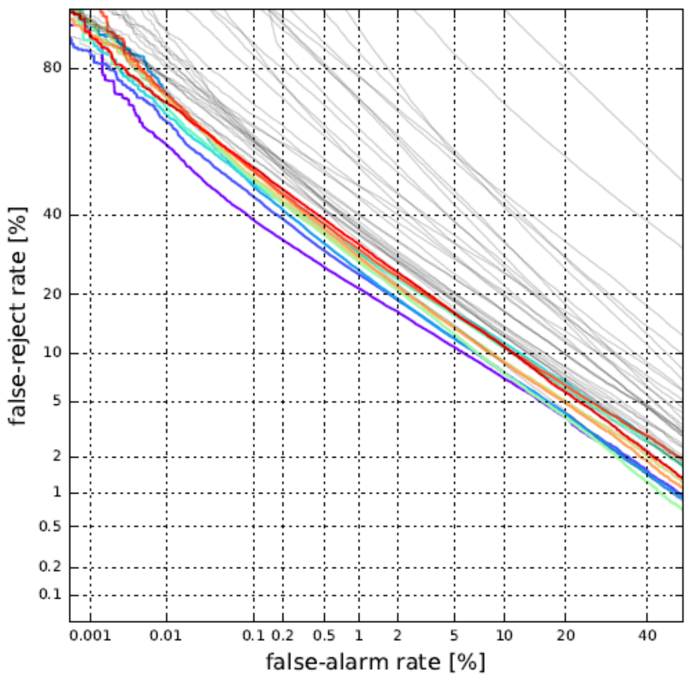

Figure 3.1: Detection error trade-off (DET) graph from NIST SRE 2016 for primary submissions (from [15])

There will always be the trade-off between false accepts and false rejects and it needs to be properly adjusted according to the costs. If the application has very high costs for *false* *accepts*, the threshold can be adjusted so that it only accepts those trials, that produce a very high score, but will, in return, produce a lot of *false rejects* and vice versa.

Therefore, NIST established a detection cost function (DCF) as the primary evaluation metrics for accessing the performance of text-independent speaker verification (Section 3.4). The goal of DCF is to minimize the cost for our decision, which would inherently help us to set a threshold for our decision-making, that is application dependent. DCF is often shown as an operating point on DET.

3.3 Equal Error Rate (EER)

We need to note that trying to strive for a „zero-tolerance“ policy for one type of error will cause the other rate of error to skyrocket, which is not very recommended. At some point in the DET, the probability of *False accept (FA)* equals the probability of *False reject (FR)*; this value is referred to as an *Equal Error Rate (EER)* [10].

EER is expressed as a percentage value of error for given data-set. If the decision threshold is set from EER threshold, then the two types of error become equal. Operating point for EER is shown on the DET curve, such as in Figure 3.2.

3.4 Detection cost function (DCF)

One of the most frequently used evaluation metrics for text-independent speaker recognition systems is the Detection cost function (DCF). DCF is as follows:

$$DCF = C_{FA}R_{FA}(1 - P(target)) + C_{FR}R_{FR}P(target) \tag{3.1}$$

where $P(target)$ is an a priori probability for a target trial as we expect in our application before we observe the trial. C_{FA} is the cost of false accept and C_{FR} is the cost of false reject. Both of these costs must be chosen according to the application, therefore are application dependent. The rates of errors are computed from the evaluation database as:

$$R_{FA} = \frac{\text{Total False Accepts}}{\text{Total Non-target Trials}} \tag{3.2}$$

$$R_{FR} = \frac{\text{Total False Rejects}}{\text{Total Target Trials}} \tag{3.3}$$

which is an estimate of $P(Error|Non)$ and $P(Error|Target)$.

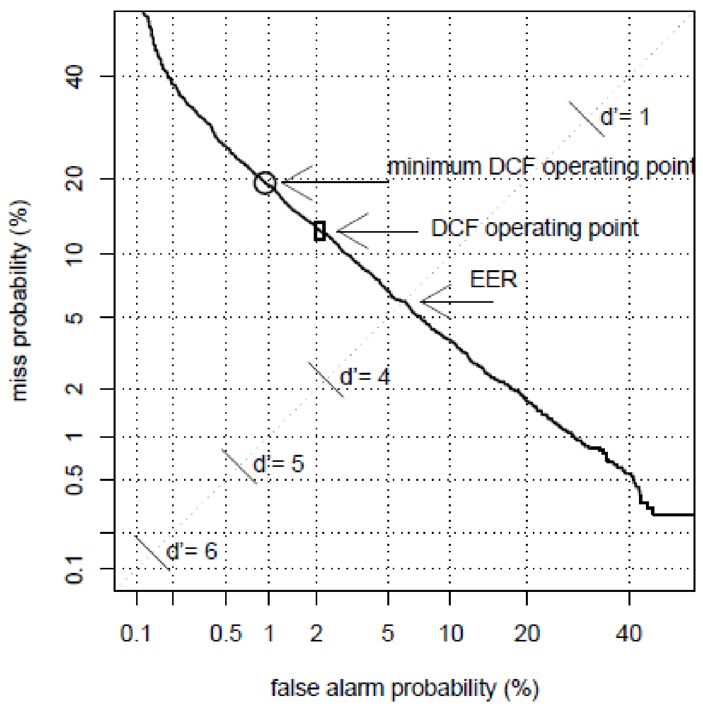

Figure 3.2: Graph of DET with DCF, minDCF and EER (from [10])

Given the cost parameters and a prior of the DCF, we wish to set the threshold so that the expected cost is minimized. Therefore, decision threshold can be set by minimizing the DCF. In NIST 2016 Speaker Recognition Evaluation Plan[1], the minDCF is computed by iterating over rates for false accept and false reject and finding the lowest DCF value. Finally, the the cost is normalized by C_{Default}, which is minimum of following values:

$$C_{\text{Default}} = \min \begin{cases} C_{\text{FalseReject}} \times P_{\text{Target}} \\ C_{\text{FalseAccept}} \times (1 - P_{\text{Target}}) \end{cases}$$

The cost is normalized by dividing it by C_{Default}, which is the best cost that could be obtained without processing the input data (i.e., by either always accepting or always rejecting the segment speaker as matching the target speaker, whichever gives the lower cost).

$P(target|data)$ is the probability of target given data after we observe the trial. A perfect system would output $P(target|data) = 1$ for a *target trial* and $P(target|data) = 0$ for a non-target trial. However, bad system would output $P(target|data) = 1$ only for very easy *target trial*, $P(target|data) = 0$ for very easy *non-target trial* and in most cases, it would output $P(target|data) = 0.5$. Our goal is to minimize the cost, so we will accept the decision that involves the lowest possible cost. In other words, if the expected cost of accepting a trial is smaller than the expected cost of rejecting it, we should accept it. The optimal threshold on the Log Likelihood Ratio (LLR) can be derived as follows:

[1]Available at `https://www.nist.gov/system/files/documents/2016/10/07/sre16_eval_plan_v1.3.pdf`

$$\frac{\text{Expected cost of accepting the trial}}{\text{Expected cost of rejecting the trial}} < 1 \tag{3.4}$$

$$\frac{C_{FA}(1 - P(target|data))}{C_{FR}P(target|data)} < 1 \tag{3.5}$$

We can use the Bayes's rule to rewrite it as:

$$\frac{C_{FA}\frac{P(data|non)P(non)}{P(data)}}{C_{FR}\frac{P(data|target)P(target)}{P(data)}} < 1 \tag{3.6}$$

$$\frac{C_{FA}P(data|non)P(non)}{C_{FR}P(data|target)P(target)} < 1 \tag{3.7}$$

$$\frac{C_{FA}P(non)}{C_{FR}P(target)} < \frac{P(data|target)}{P(data|non)} \tag{3.8}$$

We want the system to output the (logarithmic) likelihood ratio rather than the posterior probability, because we want to be able to set the prior probability $P(target)$. Therefore, the prior probability $P(target)$ should not be included in the model at all. Instead, it should exist as an input parameter [14].

$$\log\frac{C_{FA}P(non)}{C_{FR}P(target)} < \log\frac{P(data|target)}{P(data|non)} \tag{3.9}$$

Since it is a binary decision, if we do not accept the decision, then we will reject the decision according to the above rule.

We can also scale the DCF for optimization and it stays equivalent to the original DCF as:

$$DCF' = P_{EFF}P_{FR} + (1 - P_{EFF})P_{FA} \tag{3.10}$$

$$P_{EFF} = \frac{P(target)C_{FR}}{P(target)C_{FR} + (1 - P(target)C_{FA})} \tag{3.11}$$

By doing so, we can use P_{EFF} *(effective prior)* to find the threshold for the log likelihood ratio, as mentioned in [14]:

$$\tau = -\Big(\log\frac{P(target)}{1 - P(target)} + \log\frac{C_{FR}}{C_{FA}}\Big) \tag{3.12}$$

$$\tau = -\log\frac{P_{EFF}}{1 - P_{EFF}} \tag{3.13}$$

If the speaker verification system outputs the scores that can be interpreted as the log likelihood ratio, the threshold can be set for any P_{EFF}.

3.5 Proposed metrics

The goal is to develop metrics, that would accurately measure and evaluate the performance of the speaker recognition systems, whilst taking the computational costs into account. In general, taking computational costs and time into account is very subjective task, since the output can drastically change depending on the used hardware.

Therefore, the evaluation process will be carried out on the same machine so the results can be compared objectively. During our research, we proposed two metrics that could be used for the evaluation process, that are based on the NIST evaluation framework and extended by the cost of time.

It is important to note, that the above mentioned evaluation strategies are sufficient for the research, that is result-oriented. My goal is, however, to explore the possibility of extending such metrics for use in **speaker recognition systems**, where computational time *(computational cost)* is important as well.

The systems' performance will be judged according to the new metrics and the best one will be deployed with normal decision threshold. New metrics will not affect the decision threshold itself, but it will rather help us choose the best system for given application.

3.6 Time Constraint Protocol (TCP)

The goal of the Time Constraint Protocol (TCP) is to set an upper limit for the computational time. We can safely assume that the threshold for the computational time should be adjusted based on the expected costs of error. For example, a banking system that would use speaker recognition technology in order to verify the identity of the client would have high monetary costs if it accepted an impostor, but rejecting a target speaker would only result in an annoyance of the client with no monetary costs for the bank.

As a base of this protocol, the maximum available time will be denoted as Θ. The variable Θ can be set arbitrarily, but it makes sense to set it more objectively when possible. Many similar applications, such as the banking system mentioned above, use an `Software agent` that communicates with the user. In this particular scenario, the user will have to verify their identity and the system will start processing. The software agent usually informs the user, that the input is being processed. The time it takes to convey this information will become the Θ value, which would ensure, that the system must make the decision by the time the agent has finished the communication.

For better decision making process, we will establish ε, which is a variance from Θ and it was experimentally chosen to be 5% of the Θ value, but can be changed arbitrarily. The requirements are:

$$\Theta > 0 \tag{3.14}$$

$$\varepsilon > 0 \wedge \varepsilon < \Theta \tag{3.15}$$

Θ is the computational cost we strive to achieve. The purpose of the variance ε is to show the range that is very close to Θ. It is the extra time that we are willing to sacrifice, which is also application specific. It is possible to visualize the TCP results on numerical axis with pivotal timestamps and current system position.

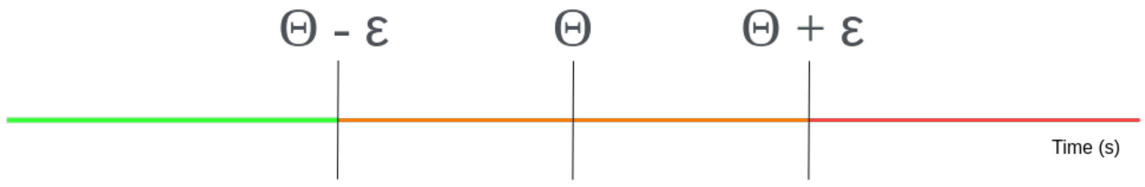

Figure 3.3: Visualization of TCP on time axis

The axis is divided into 4 intervals. The colors represent state of the system in given interval. Multiple systems' results can be plotted onto this axis, with clear visual comparison for evaluation of the system. In order to obtain a scalar value for the evaluation, we can do so by subtracting Θ from the system processing time (T) :

$$T - \Theta = \Delta \tag{3.16}$$

where Δ is used for the evaluation of the constraint fulfillment.

$\Delta > \varepsilon$	Did not fulfill the constraint
$\varepsilon \geq \Delta > 0$	Almost fulfilled the constraint
$0 \geq \Delta > -\varepsilon$	Fulfilled the constraint
$-\varepsilon \geq \Delta$	Fulfilled the constraint very well

The reason for finding Δ instead of comparing T against Θ directly is because we want to able to compare systems and their result of TCP for different Θ and ε values.

3.7 Modified Detection Cost Function (MDCF)

The original Detection Cost Function (DCF) introduced in Section 3.4 creates solid basis for evaluation of speaker recognition systems, but does not take the computational time into account. My goal is to propose a modified DCF (MDCF), that is based on the original DCF, but introduces time cost variable into it.

The modification of the DCF involves appropriately adding the time cost into the function. My proposed function is

$$MDCF = C_{FA}R_{FA}(1 - P(target)) + C_{FR}R_{FR}P(target) + TC_T \tag{3.17}$$

MDCF uses the same input parameters as in the original DCF and introduces 2 new parameters

- T: average time needed to make a final decision
- C_T: cost for one time unit

Product of these parameters is added to the function. Parameter T does not depend on the final decision, so we do not need to measure time differently for accepting and rejecting.

Time T represents the time it takes for the system to process and compare two utterances against each other (because we are doing verification). Units for time must be consistent over all experiments for easier comparison. This thesis will be using seconds as a time unit. Cost C_T is set manually and has the value of 1 by default.

MDCF can be minimized to find minMDCF – the lowest possible cost including the computational time. MDCF is expected to rank systems with lowest DCF and lowest computational time the highest. It will make the biggest difference with ranking systems, that have similar DCF values, but different computational time.

Chapter 4

Experiments

This chapter focuses on experiments, presents their results and compares different approaches. The experiments have been conducted using Kaldi [13].

Kaldi is a toolkit for speech and speaker recognition written in C++, licensed under the Apache License v2.0, with the intended use by the research community.[1] Kaldi provides excellent basis for speaker recognition – it contains many recipes and examples that are open to modification and optimization. The biggest advantage of Kaldi is that it provides a huge base of source files for the speaker recognition systems.

The remainder of this Chapter is organized as follows: Section 4.1 describes the experimental setup. Section 4.2 focuses on measuring the time and calculating the average processing time. Section 4.3 and 4.4 propose different modifications to the existing system in order to improve its performance according to TCP and MDCF. Finally, Section 4.5 compares all the systems' modifications.

4.1 Setup

The goal is to observe a comparable results, whilst taking computational costs into account. Therefore, in order to compare the experiments objectively, it is essential to obtain the results from the same hardware. Because of that, I used my personal school laptop with the following specifications for all the experiments and trials, except for training the Deep Neural Networks (DNN):

Manufacturer	Asus
Edition	Zenbook
Model	UX410UA-GV018T
Processor	Intel Core i5
Clock rate	2.5 GHz
RAM	8 GB DDR4
Operating system	Ubuntu
Version	18.04

Table 4.1: Hardware and software used for the experiments

[1] Available at http://kaldi-asr.org/

Database called VoxCeleb has been used for training and evaluating the used model. VoxCeleb is large scale audio-visual data-set of human speech, extracted from various interviews of celebrities uploaded to YouTube. The data-set has been released in two stages as VoxCeleb1 and VoxCeleb2. For our use, it was efficient enough to use the VoxCeleb1 data-set, which contains **153 516 utterances** from **1251 speakers**. The data-set contains utterances from people of with various ethnicities, accents and ages. The whole data-set is gender balanced, with around 55% of males and 45% females. The important thing is that almost all utterances are from real-life environment that contain some background noise, and also channel noise [12].

For the concrete experiments with the proposed metrics, I used Voxceleb v2 recipe from Kaldi. This recipe is for speaker verification using the VoxCeleb1 data-set (`kaldi/egs/voxceleb/v2`).[2] This recipe uses Deep Neural Networks (DNN) (`nnet3`) source file and pre-trained x-vector model introduced in Section 2.4. I used pre-trained model, because it is very time consuming and hardware intensive to fully train the DNN model and it is not the main goal of this thesis. The used model also contains a PLDA backend for scoring embeddings.[3]

The whole recipe consists of 12 stages, first 8 stages focus on preparing the data, augmenting it, extracting the x-vectors and training the DNN model. I am focusing on all the stages that are used for evaluation.

The goal was to conduct the experiments in the worst case scenario, i. e. with the longest possible computational time. That means clearing the cache memory between the experiments. For time measurement, I used the Unix utility `time` that measures elapsed time for each stage [1]. This command outputs the processing time statistics. It outputs following 3 statistics by default:

- **Real:** “real” time from start of the call until the end of the call, which includes time when other processes were carried out and time, when processes are blocked and waiting for other operations to finish, such as I/O
- **User:** CPU time spent in user-mode code (outside the kernel) within the process
- **Sys:** CPU time spent inside the kernel within the process

Adding *User time* and *Sys time* together then represents the total time for processing used by CPU by the measured process. For the purpose of this paper, the time used for calculations will therefore be the addition of *User time* and *Sys time* and it will be referred to as ***Processing time***.

4.2 Kaldi script evaluation

All stages' time have been measured and recorded in a simple file. The only stage not recorded were the stages 6-8, where the model is supposed to be trained, because we used pre-trained model. This recipe contains a lot of data preparation and data augmentation with various noises required for training. These steps were ignored, as the aim is to observe and measure time for steps that would occur at testing time inside production. Because of that, we will be using VoxCeleb evaluation data-set, which contains $4,874$ utterances. There is total of $37,720$ trials, with $18,860$ (50%) target trials and $18,860$ (50%) non-target

[2]Recipe available at https://github.com/kaldi-asr/kaldi/tree/master/egs/voxceleb/v2
[3]Model is available at https://kaldi-asr.org/models/m7

trials. The script together with VoxCeleb data-set are computational heavy tasks, so the script was run 3 times with the averages represented below:

Stage No.	**Stage Name**	**Total Processing Time (s)**	**Average Time/Utterance (s)**
1	MFCC	159.04667	0.03263
1	VAD	2.95000	0.00061
9	X-vector extraction	3,507.00000	0.71953
11	PLDA Scoring	0.94000	0.00019
Total		3,669.93667	0.75296

Table 4.2: Time measurement for the original Kaldi recipe without any modifications

In order to produce correct results for Average Time/Utterance, we always need to realize what data we're working with. MFCC uses the entire utterances. VAD marks speech and non-speech frames for the entire utterance. However, x-vector extraction is using only the speech frames, not the entire utterances.

If we want to correctly measure the *processing time*, we need to account for the fact that we are testing 2 utterances against each other, so *feature extraction*, *VAD*, *x-vector extraction* is done 2 times, while the *PLDA scoring* is done only once. Making one single decision therefore takes on average:

$$\text{Processing time} = 2 \times (\text{Feature Extraction} + \text{VAD} + \text{x-vector Extraction}) + \text{PLDA} \tag{4.1}$$

$$\text{Processing time} = 2 \times (0.03263 + 0.00061 + 0.71953) + 0.00019 \tag{4.2}$$

$$\text{Processing time} = 1.505731 \text{ (s)} \tag{4.3}$$

From these experiments, it can be concluded that comparing 2 utterances and making a final decision takes approximately 1.5 seconds for this Kaldi recipe. Every conducted experiment uses this math to obtain the average processing time. Following experiments will not show such a detailed breakdown for every time measurement, but will only present the average processing time as a result.

Our optimization goal is to reduce the average *processing time* for making the final decision. It has been proven that x-vector extraction and MFCC extraction take the longest time (almost 99.5% of the total time needed). Therefore, we focus on reducing the computational costs in those stages.

We can also use the original metrics to evaluate the performance of this system. Experiments will present the new metrics alongside EER and minDCF. This Kaldi recipe produces DET curve with Equal Error Rate (EER) of 2.359%. minDCF for $P(tar) = 0.01$ is 0.2504 and minDCF for $P(tar) = 0.001$ is 0.3848 with all costs having the default value of 1.

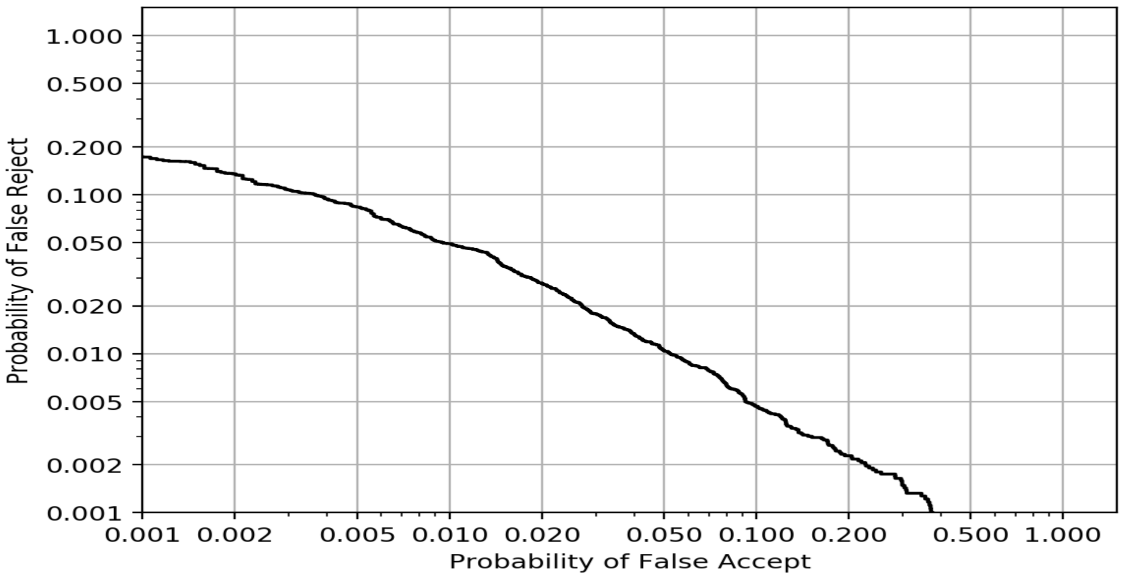

Figure 4.1: DET curve for Kaldi recipe without any modifications

4.3 Utterance length shortening (A)

Very intuitive solution for reducing the computational costs is to shorten the utterance length. However, that will, naturally, make the system more susceptible to errors, because it extracts less information from the utterances. Therefore, it is essential to find a good balance between overall system performance, computational costs and sufficient utterance length.

Overall, we carried out 33 experiments for utterance shortening – 3 experiments for each maximum utterance length. The experiments measured time in the same stages as illustrated in Table 4.2. VoxCeleb1 evaluation data-set was used for these experiments. The original utterances most commonly have $5 - 15$ seconds, and only a few utterances have exceeded the 30 second mark.

The experiments' output show the maximum length for utterances in seconds. The utterances that exceed the maximum limit have been cut to the maximum limit, other utterances have been kept at their original length. The experiments also record, how many files have been modified and what percentage it adds up to. Average *processing time (PT)* is the average time estimate it takes for the system to make a final decision. The original evaluation metrics (EER and DCF) are shown together with the new metrics Modified Detection Cost Function (MDCF) introduced in Section 3.7. Results labeled as (1) use a priori probability $P(tar) = 0.01$ and results labeled as (2) use a priori probability of $P(tar) = 0.001$.

Max Length (s)	**Modified files #**	**Modified files (%)**	**Average PT (s)**
Original	0	0.00	1.50573
60.0	5	0.10	1.52997
45.0	17	0.35	1.49442
30.0	56	1.15	1.48387
15.0	410	8.41	1.44486
12.5	672	13.79	1.39352
10.0	1088	22.32	1.36499
7.5	1856	38.08	1.24434
5.0	3558	73.00	0.85249
3.0	4874	100.00	0.46655
1.0	4874	100.00	0.23953

Table 4.3: Number of modified files for given utterance length and the average processing time

As expected, the average *processing time* is lower for shorter utterances. The most substantial drop in processing time happens under the 10 second mark. The ranking of the systems by MDCF is very different compared to the ranking by DCF. There is a slight difference in ranking the system between MDCF(1) and MDCF(2).

Max Length (s)	**EER (%)**	**minDCF (1)**	**minDCF (2)**	**MDCF (1)**	**MDCF (2)**
Original	2.35900	0.25040	0.38480	1.75613	1.89053
60.0	2.35400	0.25010	0.38490	1.78007	1.91487
45.0	2.35400	0.25140	0.38480	1.74582	1.87922
30.0	2.34900	0.25140	0.38530	1.73527	1.86917
15.0	2.33800	0.25130	0.38700	1.69616	1.83186
12.5	2.34900	0.25780	0.39170	1.65132	1.78522
10.0	2.40200	0.26650	0.39750	1.63149	1.76249
7.5	2.49200	0.28450	0.37330	1.52884	1.61764
5.0	3.20300	0.35820	0.55770	1.21069	1.41019
3.0	5.70000	0.57750	0.67310	1.04405	1.13965
1.0	23.55000	0.99500	0.99500	1.23453	1.23453

Table 4.4: Metrics results for various maximum utterance lengths

We aim to find optimal length of utterances for the shortest processing time and sufficient accuracy. MDCF fulfills this purpose, as it uses the original DCF extended by time. Systems were ranked according to MDCF with time cost for one time unit being 1. The smaller the MDCF the more optimal the system was in both accuracy and speed. The most suitable utterance length is therefore 3 seconds. It is reasonably short, so it reduces the processing time, but at the same time, the system still achieves feasible results.

Rank	**Max Length (s)**	**MDCF**
1	3.0	1.04405
2	5.0	1.21069
3	1.0	1.23453
4	7.5	1.52884
5	10.0	1.63149
6	12.5	1.65132
7	15.0	1.69616
8	30.0	1.73527
9	45.0	1.74582
10	Original	1.75613
11	60.0	1.78007

Table 4.5: Ranking of the system performance for various utterance lengths for $P(target) = 0.01$

Rank	**Max Length (s)**	**MDCF**
1	3.0	1.13965
2	1.0	1.23453
3	5.0	1.41019
4	7.5	1.61764
5	10.0	1.76249
6	12.5	1.78522
7	15.0	1.83186
8	30.0	1.86917
9	45.0	1.87922
10	Original	1.89053
11	60.0	1.91487

Table 4.6: Ranking of the system performance for various utterance lengths for $P(target) = 0.001$

These experiments helped to find the most optimal utterance length for this system that satisfies the new metrics. MDCF proves to be a good metric, because it uses results from minDCF, which is a good indicator for comparing different systems.

Time limit for Time Constraint Protocol (TCP) was set to 1.35 seconds and variance was set to 20%. TCP categorized the results according to the Δ values as follows:

Max Length (s)	**Δ**	**Result**
Original	0.15573	Almost fulfilled the constraint
60.0	0.17997	
45.0	0.14442	
30.0	0.13387	
15.0	0.09486	
12.5	0.04352	
10.0	0.01499	
7.5	-0.10566	Fulfilled the constraint
5.0	-0.49751	Fulfilled the constraint very well
3.0	-0.88345	
1.0	-1.11047	

Table 4.7: Results of the TCP

TCP separated the results into 3 categories according to how well it satisfied the constraint. It is possible to choose any option from given category, but if we look at EER, minDCF(1) and minDCF(2), then the most optimal length is 7.5 seconds. The system satisfies the constraint posed by TCP and achieves EER of 2.492%, minDCF(1) of 0.28450 and minDCF(2) of 0.37330.

TCP can also be visualized in a graph. The colors represent how well the system performed. Black horizontal lines are the borders between intervals. System performance is then shown as a blue dot on a time axis.

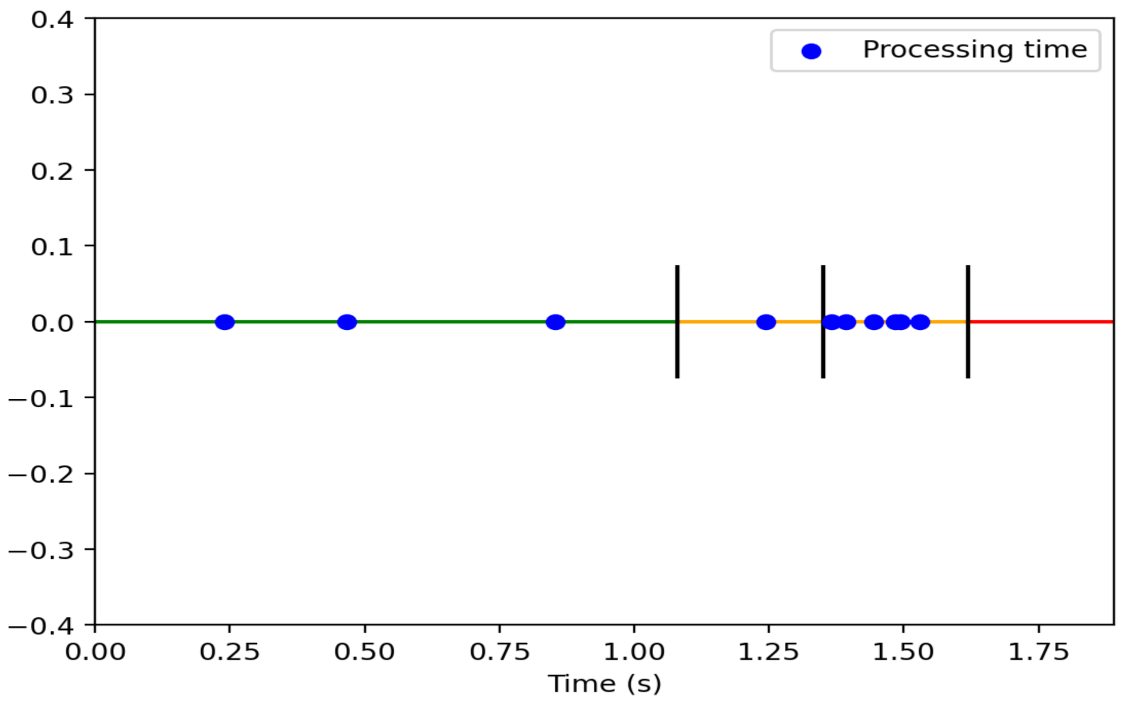

Figure 4.2: Graph of TCP and the system performance

Another possibility is to cut the utterance length after VAD was applied. That would ensure that the information contains as little silence as possible.

4.4 Processing files before x-vector extraction (B)

In Section 4.3, we achieved overall lower average processing time by setting a limit for maximum utterance length. That decreased the computational cost mainly for MFCC, as well as x-vector extraction. We evaluated the Kaldi recipe and found out that processing maximum 3 second utterances is the most beneficial for the system in terms of computational costs by MDCF. However, utterances as short as 3 seconds or less can contain a lot of silence. In case of the user agent system that directly communicates with people, it may take some time for the person to realize that it is time to speak, so the utterance can contain a lot of silence and too little information.

Therefore, the actual amount of speech used to create the x-vectors is different for different utterances. This experiment would cut the utterances in such a way that the amount of speech used to create each x-vector is the same. Processing files after VAD will cause each segment to have equally many speech frames for x-vector calculation, as opposed to simply restricting the length of utterances, which may results in some files having fewer speech frames to work with.

The goal is to limit the maximum number of frames in similar manner as illustrated in Section 4.3, but after VAD. As a result, we expect to observe decline in average processing time for favorable evaluation by the new metrics.

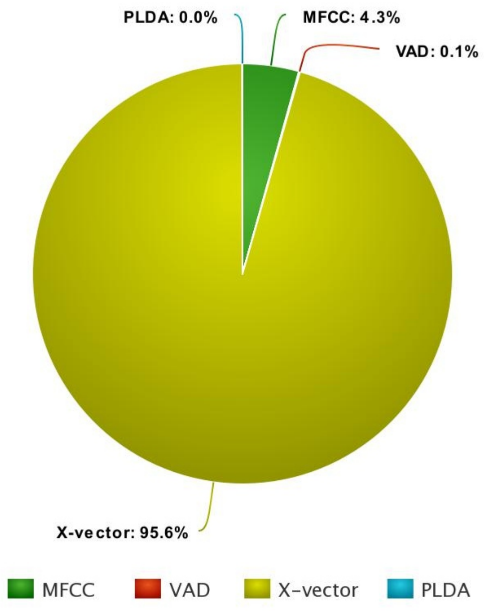

Figure 4.3: Distribution of the average processing time for different stages of the Kaldi recipe

If we look at the distribution of processing time for different stages of Kaldi, we can see that 95.6% consists of x-vector extraction. Therefore, we aim to reduce the processing time for that stage. This experiment will not affect computational time for any other stages but the x-vector extraction, which is the most time-consuming stage. Further explanation about i-vectors and x-vectors can be found in Section 2.4.

We need to cut frames after VAD, which required additional step in the process. This step involves checking the total number of speech frames in given utterance, comparing it with maximum number of frames allowed, calculating the final number of frames and actually segmenting the required number of frames. This takes roughly 2 seconds for all utterances and will be added to the total processing time. Average processing time includes this pre-processing average twice as well (twice because we are comparing 2 utterances), as to simulate the production system environment.

In this recipe, 1 second of audio is represented as 100 frames. Therefore, we will be setting the constraint on maximum number of frames for x-vector extraction. Maximum length in seconds will be denoted as well for approximate comparison to Section 4.3.

This experiment modified less files in each run. Average processing time starts dropping drastically under $1,500$ frames.

Max Frames (#)	**Modified files (#)**	**Modified files (%)**	**Average PT (s)**
-	-	-	1.50573
6,000	3	0.06	1.50749
4,500	15	0.31	1.51286
3,000	53	1.09	1.49781
1,500	373	7.65	1.41059
1,250	593	12.17	1.32523
1,000	1,006	20.64	1.21187
750	1,733	35.56	1.00693
500	3,360	68.94	0.59019
300	4,862	99.75	0.22330
100	4,874	100.00	0.13168

Table 4.8: Number of modified files for given frame length and average processing time

Results of EER, DCF and MDCF are presented in the table below. EER is approximately the same for 750 frames and larger. DCF gives quite a good score, even for smaller number of frames. Consequently, MDCF has very good results, with some scores going under the value of 1.

Max Frames (#)	**EER (%)**	**minDCF (1)**	**minDCF (2)**	**MDCF (1)**	**MDCF (2)**
-	2.35900	0.25040	0.38480	1.75613	1.89053
6,000	2.34889	0.24950	0.37790	1.75699	1.88539
4,500	2.34889	0.24930	0.37790	1.76216	1.89076
3,000	2.34889	0.24960	0.37790	1.74741	1.87571
1,500	2.35949	0.24520	0.37990	1.65579	1.79049
1,250	2.34889	0.25550	0.38270	1.58073	1.70793
1,000	2.40721	0.26560	2.37060	1.47747	1.58247
750	2.46023	0.27900	0.39940	1.28593	1.40633
500	3.00636	0.34170	0.53920	0.93189	1.12939
300	5.42418	0.55910	0.64140	0.78240	0.86470
100	20.64160	0.99330	0.99640	1.12498	1.12808

Table 4.9: Metrics results for various maximum frames lengths

Max Frames (#)	Δ	Result
6,000	0.15749	Almost fulfilled the constraint
Original	0.15573	
4,500	0.16286	
3,000	0.14781	
1,500	0.06059	
1,250	-0.02477	Fulfilled the constraint
1,000	-0.13813	
750	-0.34307	Fulfilled the constraint very well
500	-0.75981	
300	-1.12670	
100	-1.21832	

Table 4.10: Results of the TCP

TCP was satisfied for any number of frames that were 1250 and less. Overall, TCP's constraint was filled more quickly compared to experiment A. TCP has separated the results into 3 categories. The best system according to DCF, that also satisfies the TCP constraint, uses $1,250$ frames (12.5 seconds). It reaches EER of 2.349%, minDCF(1) of 0.25550, and minDCF(2) of 0.38270.

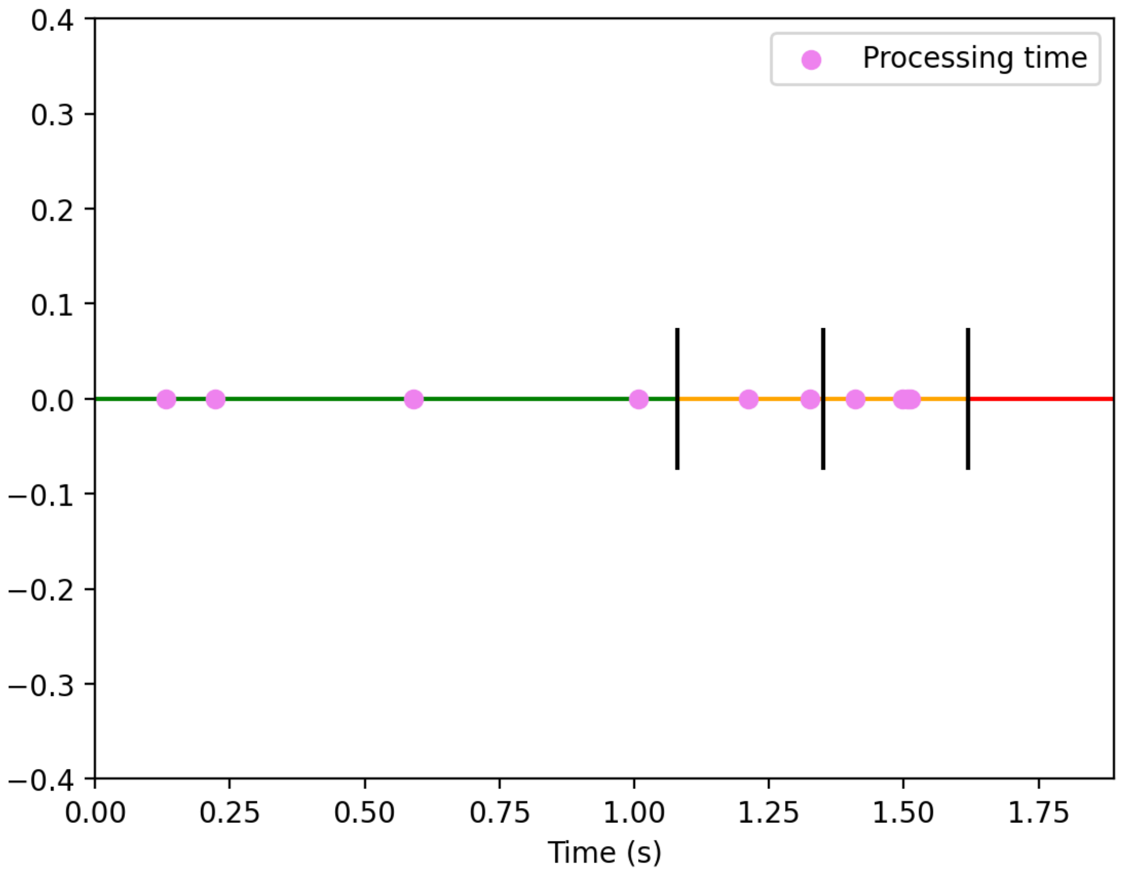

Figure 4.4: Graph of TCP and the system performance

4.5 Results comparison

We conducted 2 experiments that pre-process utterance files at various stages. Firstly we tried to reduce the length of the utterances in order to decrease overall processing time in Section 4.3 (Experiment A). The second experiment focuses on limiting the maximum number of frames extracted in the x-vector extraction stage in Section 4.4 (Experiment B).

By reviewing the experiments, we can see that B achieves smaller average processing time for the same maximum time limit. Every EER (%) is also smaller in experiment B compared to EER produced in experiment A. Results from MDCF also favor Experiment B, with few values going lower than 1, which did not happen at all for experiment A. This is the ranking of all the system modifications by MDCF(1). Maximum limit in seconds has been used for both experiments, with the conversion rate of 1 second = 100 frames for experiment B.

Rank	**Experiment**	**Max Limit (s)**	**minDCF (1)**
1	B	3.0	0.78240
2	B	5.0	0.93189
3	A	3.0	1.04405
4	B	1.0	1.12498
5	A	5.0	1.21069
6	A	1.0	1.23453
7	B	7.5	1.28593
8	B	10.0	1.47747
9	A	7.5	1.52884
10	B	12.5	1.58073
11	A	10.0	1.63149
12	A	12.5	1.65132
13	B	15.0	1.65579
14	A	15.0	1.69616
15	A	30.0	1.73527
16	A	45.0	1.74582
17	B	30.0	1.74741
18	-	Original	1.75613
19	B	60.0	1.75699
20	B	45.0	1.76216
21	A	60.0	1.78007

Table 4.11: Final ranking of all the experiments by MDCF (1)

Experiment B tends to rank higher compared to the same maximum limit results of A. We can conclude that experiment B performed little better for the new metric MDCF.

Both experiments used the time limit of 1.35 seconds with 20% variance. Almost half of all the results managed to fulfill the constraint and no value of Δ was higher than the variance in seconds, ε.

Since experiment B achieved lower average processing time, it also performs very well for TCP. Even though experiment B had a pre-processing stage for x-vector extraction added to it, it still outperforms system in experiment A. The following table firstly separates the results by Δ and then ranks them by minDCF within each category.

Rank	**Exp.**	**Max Limit (s)**	Δ	**minDCF (1)**	**Result**
1	B	7.5	-0.34307	0.27900	Fulfilled the constraint very well
2	B	5.0	-0.75981	0.34170	
3	A	5.0	-0.49751	0.35820	
4	B	3.0	-1.12670	0.55910	
5	A	3.0	-0.88345	0.57750	
6	B	1.0	-1.21832	0.99330	
7	A	1.0	-1.11047	0.99500	
8	B	12.5	-0.02477	0.25550	Fulfilled the constraint
9	B	10.0	-0.13813	0.26560	
10	A	7.5	-0.10566	0.28450	
11	B	15.0	0.06059	0.24520	Almost fulfilled the constraint
12	B	45.0	0.16286	0.24930	
13	B	60.0	0.15749	0.24950	
14	B	30.0	0.14781	0.24960	
15	A	60.0	0.17997	0.25010	
16	-	Original	0.15573	0.25040	
17	A	15.0	0.09486	0.25130	
18	A	30.0	0.13387	0.25140	
19	A	45.0	0.14442	0.25140	
20	A	12.5	0.04352	0.25780	
21	A	10.0	0.01499	0.26650	

Table 4.12: Final ranking of all the experiments by TCP and minDCF

Only 4 results from experiment A fulfilled the constraint imposed by TCP, whilst 6 results from experiment B fulfilled the constraint. Therefore, experiment B has shown better performance for TCP as well. Experiment B also consistently ranks higher for every category. The best result is for experiment B with 750 frames (7.5) seconds. It fulfills the constraint very well and still maintains relatively low minDCF.

This graph shows cumulative results for both experiments, with the original system marked separately.

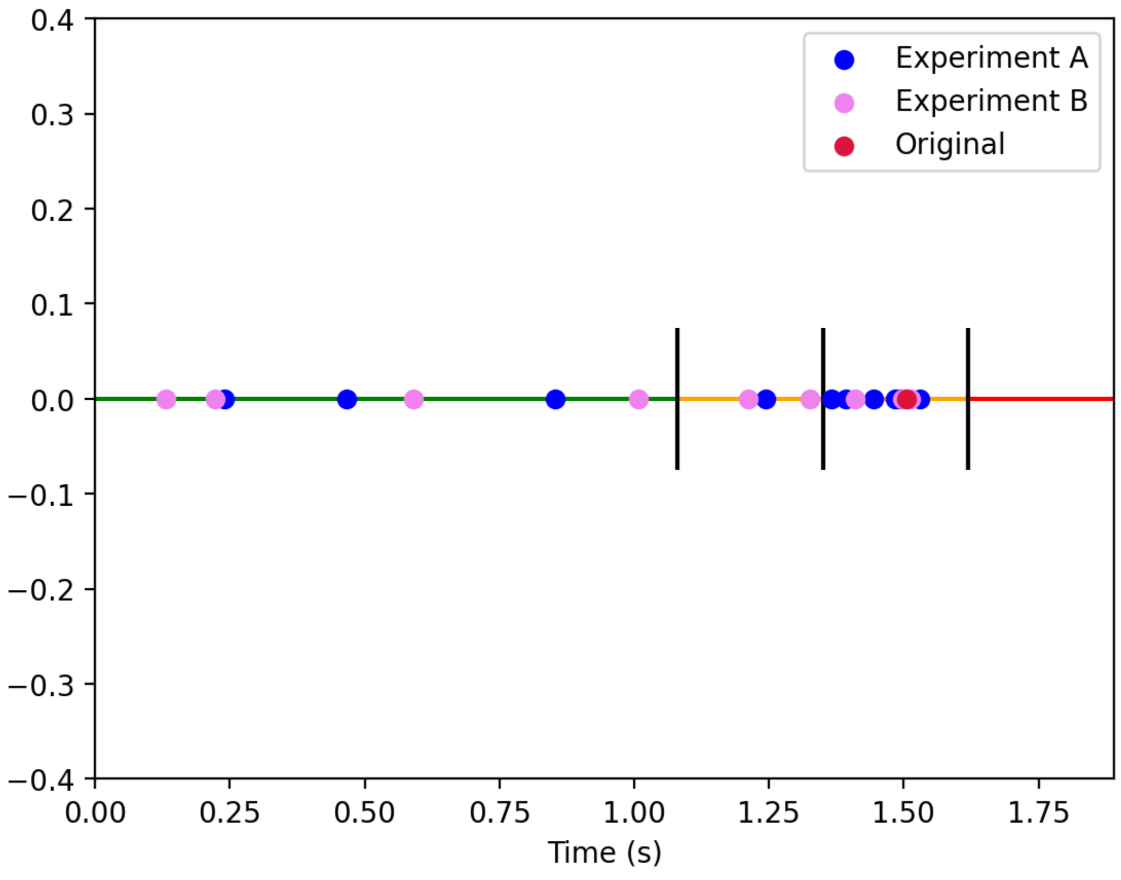

Figure 4.5: Final results of all the experiments by TCP

Experiment B has proven that limiting the frames for x-vector extraction improves the performance more than simply shortening the utterance length as illustrated by experiment A. Another possibility for future research would be combining the experiments together, i.e. shortening the maximum utterance length and them posing a limit for maximum number of frames. Such experiment has potential to reduce the time for MFCC and x-vector extraction and speeding up the x-vector extraction for the equal number of frames.

Chapter 5

Conclusion

The goal of this bachelor thesis was to conduct a research on the existing speaker recognition systems, their evaluation and possible improvements whilst taking computational costs (time) into account. Firstly, the paper presents existing methods and models used in the speaker recognition that are later utilized for the experiments. It presents currently used evaluation metrics and proposes new approaches based on them, with computational cost included.

The paper introduces 2 new metrics for evaluating the system. First metric (MDCF) is based on NIST's DCF and modifies it so that it takes the computation time into account. Second metric (TCP) poses a constraint on maximum computational time and ranks the systems according to how they satisfy the constraint.

Experiments are carried out on a collaborative open source software Kaldi. It was beneficial to use pre-existing system and models, because we wanted to simulate the environment of a system used in production. We focus on measuring the time correctly and establish a new term, *processing time*, that represents the average time needed by the system to make a final decision (accept or deny). Rest of the experiments focuses on developing a system optimized for the new metrics.

Firstly, we focus on setting a limit for the maximum utterance length (Experiment A). It decreases the computational costs, but the system has less information about the speaker to make the decision. The ranking is done by MDCF and the lowest score represents the best possible utterance length. Metric TCP found 4 acceptable utterance lengths that fulfill the time limit of 1.35 seconds.

Because x-vector extraction is the most time-consuming stage, second experiment focuses on posing a limit on the maximum number of frames used in the x-vector extraction stage (Experiment B). X-vector extraction ensures that only voiced frames (frames that contain information about the speaker) are processed, so we are posing a limit on the maximum number of frames used for one utterance. This causes each segment to have equally many speech frames for x-vector extraction. That reduces processing time very well and even outperforms the experiment A.

The system's modifications are ranked, with experiment B achieving better results for smaller maximum number of frames. The experiments have proven that it is possible to optimize the system so that it has lower processing time, but still achieves good results.

Bibliography

[1] *Time(1) - Linux man page.* 4.3th ed. March 2020.

[2] Aroon, A. and Dhonde, S. Speaker Recognition System using Gaussian Mixture Model. *International Journal of Computer Applications.* 2015, vol. 130, no. 14, p. 38–40.

[3] Burget, L. *Klasifikace a rozpoznávání (SUR).* Brno University of Technology, March 2020.

[4] 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.

[5] Farrell, K., Mammone, R. and Assaleh, K. Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing.* 1994, vol. 2, no. 1, p. 194–205.

[6] Furui, S. Speech and Speaker Recognition Evaluation. *Text, Speech and Language Technology Evaluation of Text and Speech Systems.* p. 1–27.

[7] Huang, X., Acero, A. and Hon, H.-W. *Spoken language processing: a guide to theory, algorithm, and system development.* Prentice Education Taiwan, 2005.

[8] Jin, M. and Yoo, C. D. Speaker Verification and Identification. *Behavioral Biometrics for Human Identification.* p. 264–289.

[9] Kinnunen, T. and Li, H. An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication.* 2010, vol. 52, no. 1, p. 12–40.

[10] Leeuwen, D. A. V. and Brümmer, N. An Introduction to Application-Independent Evaluation of Speaker Recognition Systems. *Lecture Notes in Computer Science Speaker Classification I.* 2007, p. 330–353.

[11] Matejka, P., Plchot, O., Glembek, O., Burget, L., Rohdin, J. et al. 13 years of speaker recognition research at BUT, with longitudinal analysis of NIST SRE. *Computer Speech and Language.* december 2019, vol. 63.

[12] 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.

[13] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O. et al. The Kaldi Speech Recognition Toolkit. In: *IEEE 2011 Workshop on Automatic Speech*

Recognition and Understanding. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[14] Rohdin, J. *A study on discriminative training techniques for speaker verification*. 2015. Dissertation. Tokyo Institute of Technology.

[15] Sadjadi, S. O., Kheyrkhah, T., Tong, A., Greenberg, C., Reynolds, D. et al. The 2016 NIST Speaker Recognition Evaluation. *Interspeech 2017*. 2017.

[16] Snyder, D., Garcia Romero, D., Sell, G., Povey, D. and Khudanpur, S. X-Vectors: Robust DNN Embeddings for Speaker Recognition. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018. Available at: https://www.danielpovey.com/files/2018_icassp_xvectors.pdf.