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

DETECTION OF PRE-RECORDED MESSAGES IN SPEECH

DETEKCE PŘED-NAHRANÝCH ÚSEKŮ V ŘEČI

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **DOMINIK BOBOŠ**
AUTOR PRÁCE

SUPERVISOR **Doc. Dr. Ing. JAN ČERNOCKÝ**
VEDOUCÍ PRÁCE

BRNO 2020

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2020/2021

Bachelor's Thesis Specification

Student: **Boboš Dominik**
Programme: Information Technology
Title: **Detection of Pre-Recorded Messages in Speech**
Category: Speech and Natural Language Processing

Assignment:

1. Get acquainted with speech processing and music analysis techniques leading to the search of identical sequences in data.
2. Create a simulated data-set from a standard speech database and real announcements from telephone operators.
3. Design and implement a system for search of re-occurring sequences using for example signal analysis, phoneme recognition, MIR techniques or speaker identification/diarization.
4. Evaluate on the created data, suggest improvements and implement them.
5. Create a short 30s video documenting your work.

Recommended literature:

- Jansen, Aren, Kenneth Church, and Hynek Hermansky. "Towards spoken term discovery at scale with zero resources." in Proc. Interspeech 2010.
- according to supervisor's recommendation

Requirements for the first semester:

- Items 1, 2 and significant advance in items 3 and 4.

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

Supervisor: **Černocký Jan, doc. Dr. Ing.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2020
Submission deadline: May 12, 2021
Approval date: May 3, 2021

Abstract

Recognition of pre-recorded messages in speech is useful for any follow-up speech data mining. This thesis summarises the theory of searching similar utterances in speech and efficient approaches to compare two sequences. To investigate identification of redundant information in audio, it is necessary to have a large amount of data with the exact phrases repeated multiple times. We generated a dataset by mixing pre-recorded messages into phone calls with variations in speed, volume and repetitions. Our system tackles "known messages" and "unknown messages" scenarios by using approaches like clustering or detection in chunks. Dynamic time warping, approximate string matching and recurrent quantification analysis are compared, and finally, all mentioned techniques are combined to obtain a precise and efficient system.

Abstrakt

Rozpoznání před-nahraných zpráv v řeči (tzv. "plechové huby") je užitečné pro jakékoliv následující dolování informací v řečových datech. Tato práce shrnuje teorii hledání podobných promluv v řeči a efektivní přístupy k porovnání dvou sekvencí. Ke zkoumání identifikace opakujících se informací v audiu je nutné mít velké množství dat s přesně se opakujícími úseky. Takovou datovou sadu jsme vygenerovali smícháním předem nahraných zpráv s telefonními hovory se změnami rychlosti, hlasitosti a opakování. Náš systém řeší scénáře „známých zpráv" a „neznámých zpráv" pomocí shlukování nebo detekce v blocích. Porovnali jsme techniky dynamického borcení času (DTW), přibližné shody řetězců a rekurentní kvantifikační analýzy, a nakonec jsme všechny uvedené techniky zkombinovali a získali tak přesný a efektivně pracující systém.

Keywords

detection of re-occurring sequences in audio, segmental dynamic time warping, recurrence quantification analysis, fuzzy string matching, bottleneck features, phoneme posteriors, Mel-frequency cepstral coefficients features

Klíčová slova

detekce opakujících se sekvencí v nahrávkách, segmentální dynamické borcení času, analýza rekurentní kvantifikace, přibližná shoda řetězců, bottleneck příznaky, fonémové pravděpodobnosti, příznaky Mel-frekvenčních kepstrálních koeficientů

Reference

BOBOŠ, Dominik. *Detection of Pre-Recorded Messages in Speech.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Dr. Ing. Jan Černocký

Detection of Pre-Recorded Messages in Speech

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Doc. Dr. Ing. Jan Černocký. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Dominik Boboš
May 9, 2021

Acknowledgements

I would like to thank my supervisor Jan Černocký for his guidance, willingness and lots of valuable suggestions. I would also like to thank members of BUT Speech@FIT group for sharing their know-how, models and code, which helps me throughout my work.

Contents

List of Acronyms

DET	Detection Error Trade-off
DTW	Dynamic Time Warping
EER	Equal Error Rate
FSM	Fuzzy String Matching
LCMA	Length-Constrained Minimum Average
LD	Levenshtein Distance
MFCC	Mel Frequency Cepstral Coefficients
MinDCF	Minimum Detection Cost Function
NN	Neural network
NMI	Normalized Mutual Information
RI	Rand Index
ROC	Receiver Operating Characteristic
RQA	Recurrence Quantification Analysis
S-DTW	Segmental Dynamic Time Warping
STD	Spoken Term Detection
QbE	Query-by-Example
QbT	Query-by-Text

Chapter 1

Introduction

Storing a large amount of speech data comes with a lot of disadvantages. On the one hand, the low-level issues as running out of free space. On the other hand, high-level as automatic processing or listening through unnecessary or repetitive data.

The main objective is to minimise wasting time by automatic processing or listening to redundant information in speech data. The case of such information could be the pre-recorded operator messages (for instance "Thank you for calling, please leave a message."). Detection could improve productivity for many professions, such as law enforcement agency (LEA). For these professions, labelling the parts of speech samples as not relevant could save their time and help to focus on more interesting data. Audio data can be analysed by many approaches, from the most commonly used as filtering, downsampling to analysing phoneme strings or similar advanced techniques.

For detecting similar or identical parts in a large dataset, more methods must be combined. First, we need to decide whether to classify redundant data based on the previous knowledge of pre-recorded messages or we do not have such an option. Second, it is essential to choose the methodology for tagging found pre-recorded operator messages. Either mark an exact time in the phone call or set binary labels whether the given recording contains pre-recorded speech. The requirement for the system is to have a fast and accurate solution with a minimum hardware load. The language resources play their role in the process as well. With low-resource languages, it is not possible to use such techniques as automatic speech recognition (ASR).

When the pre-recorded messages are known, it is possible to create a model for searching similarities in the dataset. Despite having a functioning model on some data, a new unknown message could appear at any time and get missed.

The opposite is to have zero knowledge about the messages. This method does not require a model, makes it more versatile and removes the necessity of having a database with operator messages. Detection of terms at scale with zero resources is dependent on the quality of audio files in the dataset. The same message could appear in many phone calls, but never be the same in length, volume, eventually speed or pitch. The more occurrences of the message, the more accurate the system we will get. A disadvantage of this approach is a zero accuracy when the message appears only once or only a few times.

1.1 Tasks

The main task is to classify whether a recording contains the pre-recorded message or not. This thesis presents two main scenarios how to achieve this task:

1. **Known messages scenario** – This scenario is used to create *reference clusters*. The cluster analysis is accomplished by provided labelled pre-recorded messages. Recordings are compared to reference clusters. This approach is used in experiments in sections 8.2 and 8.3.

2. **Unknown messages scenario** – Here the system does not know the pre-recorded messages and has to infer them as repeated parts of calls. This approach can be divided into the following tasks:

   a) Detection based on *a recording itself* – The process does not require any additional information, and the pre-recorded message is detected in itself. This approach is used in the system 5.2.

   b) Detection based on *all files* – Recordings are compared to all recordings in the set or by a chunk of the set. The new chunk is randomly chosen with every recording. This approach is used in several systems: in sections 5.1, 5.3, and 7.2.1.

   c) Detection based on *a list of candidates* – Audio files are compared to the list of candidates. This approach requires an analysis to create a candidate list first. Systems in sections 7.1.1 and 7.1.2 use this method.

   d) Detection based on *clusters* – Recordings are compared to the created clusters. The cluster analysis requires list of candidates. This approach is used by experiments in sections 8.2 and 8.3.

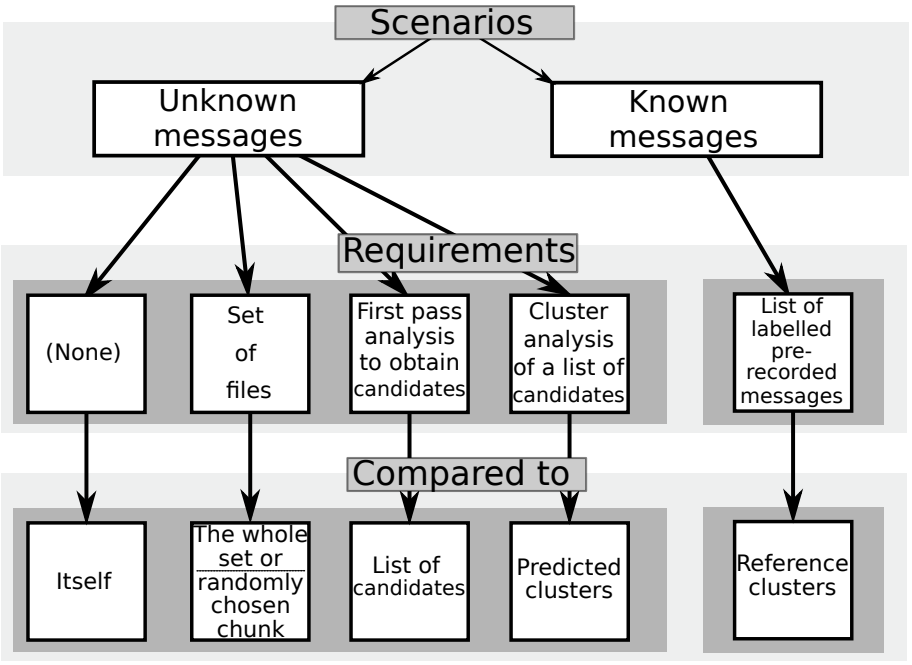


Figure 1.1: The scheme of described tasks of an evaluation process.

1.2 Organisation

To give an overview of current recognition systems, Chapter 2 will describe the basic theory associated with solving similar tasks. Chapter 3 will introduce used dataset and its generation. Used methods will be covered in Chapter 4. Implementation and applied optimization techniques will be described in chapters 5 and 6. The rest is dedicated to performed experiments and their results – Chapters 7 and 8. Finally, the conclusions are drawn in Chapter 9.

Chapter 2

State-of-the-art

Speech recognition has a wide range of applications in security systems, automotive, healthcare or many other fields. This chapter will introduce standard methods for speech recognition. The studies describe different views for solving similar tasks such as term recognition or detection of identical parts in speech.

Speech is a continuously varying signal – the proper digital processing algorithm has to be selected for the automatic speech recognition system. For isolated words detection and continuous speech recognition, various methodologies exist which evolved over the years. One of the most used is *Hidden Markov Models* (HMM) which has been used in many systems [19]. For simple tasks, the use of standard *Dynamic Time Warping* is enough. Nowadays, usage of *Deep Neural Networks* (DNN) or *Recurrent Neural Networks* (RNN) are on a rise [24], caused by better overall performance and accuracy on even less training data [16]. The choice of method depends on the many factors. Either the method is language-dependent or language-independent, speaker-dependent or speaker-independent, or there is no preliminary information. The goal is to stick to no preliminary information techniques as much as possible – the algorithms like Dynamic Time Warping (DTW).

2.1 Feature extraction and feature matching

The following methods aim at spoken term detection with no training data or dictionaries. To obtain the required information from the speech recording, features have to be extracted. The feature vectors are then analysed to make a decision [1].

2.1.1 Mel-frequency Cepstral Coefficients features

Mel-frequency Cepstral Coefficients (MFCC) are based on human hearing perceptions that is less selective for frequencies over 1KHz, thus it is more concerned with lower frequencies and their variations. For simple isolated word detection, MFCC and DTW techniques are satisfactory and efficient [17]. A combination of various features (such as bottleneck features 2.3) is to be adapted for high-reliability [19]. The process of MFCC extraction requires preprocessing the input speech to digital waveform first. The wave is segmented into 20 – 25 ms long frames at a rate of 10 ms by a windowing function (Hamming window). Fast Fourier Transform is then applied to the frames. Mel for a given frequency in Hertz is calculated by:

$$F_{MEL} = 2595 \times \log_{10}[1 + f/700]. \tag{2.1}$$

Magnitudes of FFT coefficients $X(k)$ are then multiplied by triangular Mel filters as follows:

$$s(m) = \sum_{p=1}^{P} \Big[|X(p)|^2 H_m(p) \Big], \tag{2.2}$$

where $m \in 1 \leq m \leq M$ and M is total number of triangular Mel weighted filters. $H_m(p)$ is the weight given to the p-th energy spectrum bin contributing to the $m - th$ output band. P is the number of points used to compute the FFT. Finally, the Discrete Cosine Transform (DCT) is computed on log of filter bank energies as shown:

$$c(n) = \sum_{m=1}^{M} \log\big(s(m)\big) cos\Big[n\Big(m - \frac{1}{2}\Big)\frac{\pi}{M}\Big], \tag{2.3}$$

where, $c(n)$ are the cepstral coefficients, $n = 1, 2 \ldots N$ (N is the number of MFCCs), s_m is weighted triangular Mel filter [1] [26].
The whole process is shown in Figure 2.1.

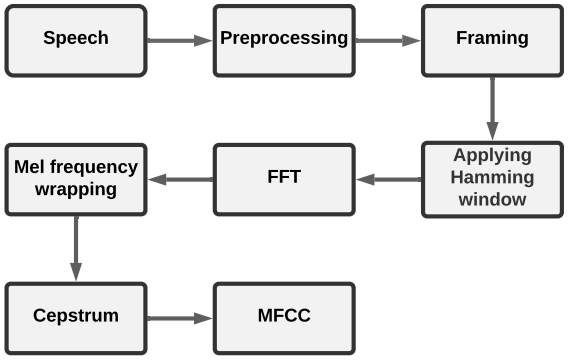


Figure 2.1: MFCC flow diagram. Taken and vectorised from [1]

2.1.2 DTW feature matching

Dynamic time warping (DTW) is an algorithm used to find the shortest distance and compare two time series data when the time indices are not synchronised [1]. The comparison between linear matching and DTW distance matching is shown in Figure 2.2.

The basic DTW is calculated as follows. Assume two time series $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_r]$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t]$ of feature vectors of lengths r and t. First an r-by-t local distance matrix is constructed where each element of the matrix contains distance $d(\boldsymbol{x}_i, \boldsymbol{y}_j)$ between the two feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$. The distance is calculated by the Euclidean distance:

$$d(\boldsymbol{x}_i, \boldsymbol{y}_j) = \sqrt{\sum_{k=1}^{F} \big(\boldsymbol{x}_i(k) - \boldsymbol{y}_j(k)\big)^2}, \tag{2.4}$$

where $\boldsymbol{x}_i(k)$ and $\boldsymbol{y}_j(k)$ are the corresponding elements of feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$. F is the number of dimensions of a feature. [17].

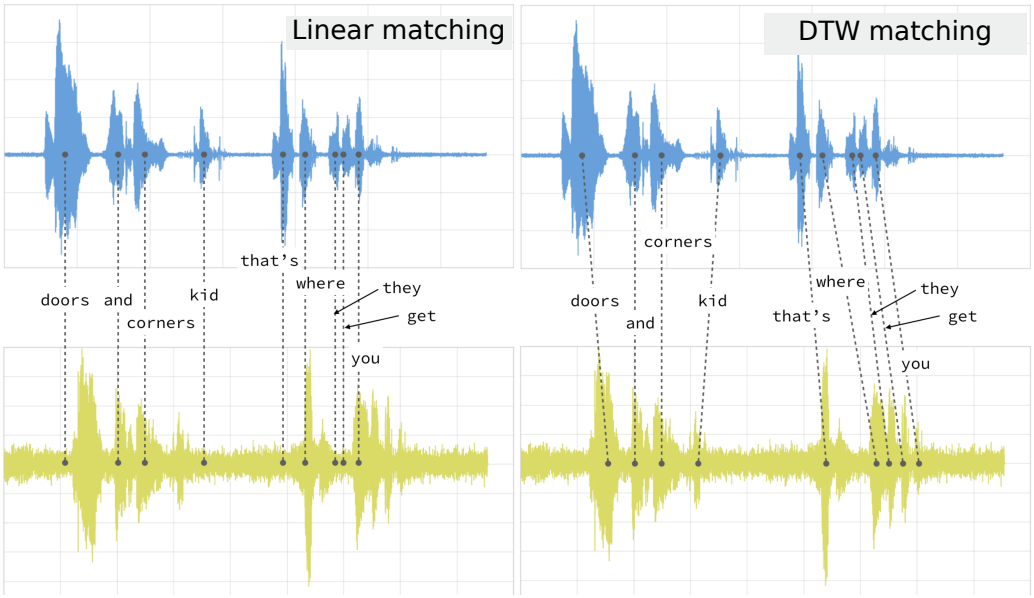


Figure 2.2: Comparison between linear matching and DTW matching on two speech recordings. Taken from [25].

The accumulated distance is computed as follow:

$$D(\boldsymbol{x}_i, \boldsymbol{y}_j) = \min \begin{Bmatrix} D(\boldsymbol{x}_i - 1, \boldsymbol{y}_j) + d(\boldsymbol{x}_i, \boldsymbol{y}_j) \times w_1, \\ D(\boldsymbol{x}_i, \boldsymbol{y}_j - 1) + d(\boldsymbol{x}_i, \boldsymbol{y}_j) \times w_1, \\ D(\boldsymbol{x}_i - 1, \boldsymbol{y}_j - 1) + d(\boldsymbol{x}_i, \boldsymbol{y}_j) \times w_2 \end{Bmatrix}, \tag{2.5}$$

where, i and j are matrix elements corresponding to the alignment between the feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, w_1 and w_2 are symmetric weights, where typically $w_1 = 1$ and $w_2 = 2$ (or $\sqrt{2}$) [38]. Example of local distance and accumulated distance is shown in Figure 2.3.

As compared recordings can vary in lengths, it is important to normalise the final DTW distance by using *normalisation factor*. Normalisation factor is in general computed as $Norm = \sum_{q=1}^{Q} W(q)$, for a symmetric weight function it is $Norm = r + t$ [38].

Final normalised DTW distance $Dist(\boldsymbol{X}, \boldsymbol{Y})$ is described as:

$$Dist(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{Norm} D(\boldsymbol{x}_r, \boldsymbol{y}_t). \tag{2.6}$$

Optimal alignment path can be obtained by backtracking – choose path with minimum distances in the accumulated distance matrix, starting from the $D(\boldsymbol{x}_r, \boldsymbol{y}_t)$ and ending in $D(\boldsymbol{x}_1, \boldsymbol{y}_1)$.

2.1.3 Segmental DTW

One of the solutions is the *S-DTW* (Segmental DTW) algorithm. The global DTW optimal alignment is a suitable way to measure similarity when the compared utterances are isolated words. However, if the utterances happen to consist of multiple sequences, the distances and path may not be useful [10].

The SDTW modification of the base DTW helps to prevent overall distortion of the warping path. Consider two utterances, `i)` "Brno University of Technology is located

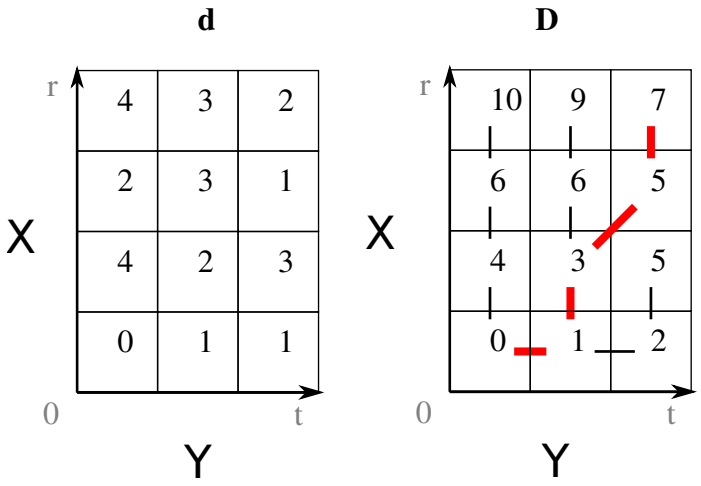


Figure 2.3: Example of two sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$. The local distance matrix $\boldsymbol{d}$ and accumulated distance matrix $\boldsymbol{D}$. DTW distance $Dist(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{3+4}7 = 1$. DTW alignment path is obtained by backtracking – $\boldsymbol{x}(k) = [1\,2\,2\,3\,3]$, $\boldsymbol{y}(k) = [1\,1\,2\,3\,4]$. Taken from: [38].

strategically close to the three capital cities.", `ii)` "One of the best technology for speaker diarisation is developed by the team at Brno University of Technology.". Both share the same phrase "Brno University of Technology" but its position differs– this leads to a chaotic DTW warping path and unsatisfactory results as standard DTW is searching only for optimal global alignment.

The principle of the S-DTW algorithm is to use other diagonals of an optimal alignment path for searching than the main diagonal. It consists of two main components: `i)` setting *global constraints* to restrict space a warping path can take and producing multiple alignment paths by changing starting and ending points in the same two input sequences, and `ii)` *path trimming procedure* which excludes largely distorted regions of an alignment path by *length-constrained minimum average LCMA* [21].

In the first step, it is important to set global constraint. The tighter is the allowed space, the more accurate result we can get, but at the price of higher computational cost. Consider two sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ and the corresponding warp path P_n. The criterion for P_n to accomplish the restriction is to satisfy condition:

$$|(i_k - i_1) - (j_k - j_1)| \leq R, \tag{2.7}$$

where R is the maximum allowed de-synchronisation, (i_1, j_1) is the origin of the path, and (i_k, j_k) is the k-th point of the path. From equation 2.7 results the width $2R+1$. Depending on the size of sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ as well as the size of R, the ending point may not reach the ending point of $\boldsymbol{X}$ or $\boldsymbol{Y}$.

This restriction determines the total number of diagonal regions. Each neighbour diagonal is overlapped by 50% as they are moved by R and the width is $2R + 1$. An example of the constraint on the warping path is shown in Figure 2.4.

The second step – *path refinement* is to select the best part from the warping paths from the first step. The task is to refine the warp by excluding the parts with high distortion. For this purpose, *length-constrained minimum average* ($LCMA$) is used first, followed by extension of the path fragment to include thresholded neighbouring points.

Let $\boldsymbol{S}$ be a sequence of positive real numbers $\boldsymbol{S} = \langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N \rangle$ and $\boldsymbol{L}$ a length constraint parameter. Then the LCMA subsequence – *LCMA(**S**, **L**)* is a consecutive subsequence of

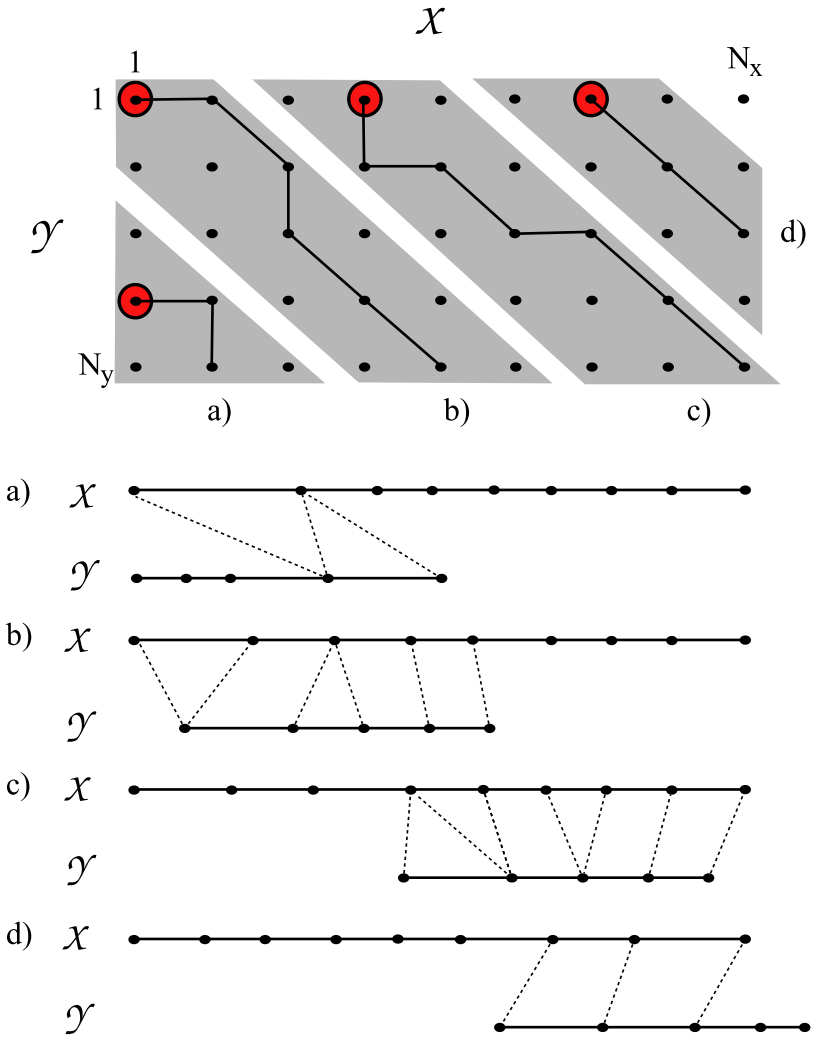


Figure 2.4: S-DTW warping paths for sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ and the maximum allowed desynchronisation $R = 1$. The starting point is marked with a red dot, and for each warping path from the grid, the corresponding alignment is shown. Source: [21].

$\boldsymbol{S}$ with a length bigger than $\boldsymbol{L}$ that is minimising the average of the values of $\boldsymbol{S}$ [21]. The constraint $\boldsymbol{L}$ helps in computing the minimum average subsequence – without it, the result would be just the smallest single elements in the sequence. Thus, the bigger the size of $\boldsymbol{L}$, the longer sequences SDTW will return. Small values lead to finding shorter sub-words in the sequence, while longer return words or even phrases. The final result of SDTW for two utterances is shown in Figure 2.5.

2.2 Optimisation across a large dataset

According to Aren Jansen's work, the hardest task is to similarities across a large dataset in an effective and fast way. The search for repeated parts across a large dataset is inherently an $O(n^2)$ problem [10].

While the S-DTW algorithm is an effective way to find repeated intervals, it is still using a lot of the processing time. Thanks to the implementation of a coarse first-pass that relies

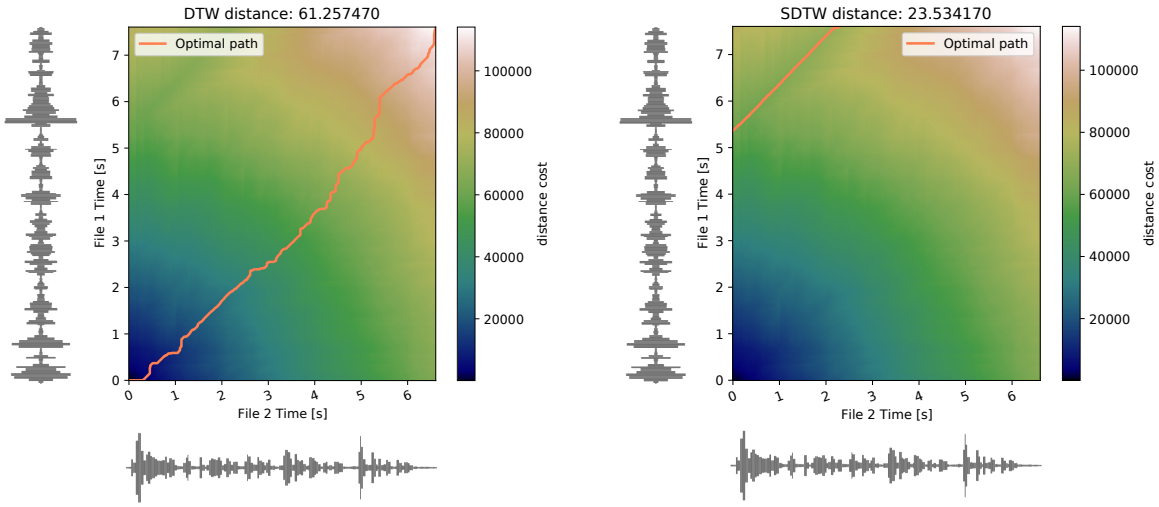


Figure 2.5: Alignment path of the two utterances. Base DTW warping path on the left, S-DTW modification on the right. File 1 represents sentence "One of the best technology for speaker diarisation is developed by the team at Brno University of Technology.", file 2 "Brno University of Technology is located strategically close to the three capital cities.". The pattern "Brno University of Technology" is found via S-DTW and thus results in a small distance compared to base DTW. Euclidean distance and standard MFCC acoustic features are used.

on representational and dot-plot sparsity, the constants improve significantly. By limiting the ultimate S-DTW search space, an orders-of-magnitude speedup is achieved [10].

2.2.1 Efficient Line Segment Detection

The solution to this optimisation problem is based on a graphical method of dotplots for comparing sequences. To obtain a dotplot for audio, the audio signals must be represented by acoustic feature vector (or phoneme posteriors) time series and a real-valued similarity between pairs of feature vectors must be used. For the given time series, the Gram matrix may be defined. The matrix is a symmetric similarity function between feature vectors from the time series. The cosine similarity is used in the matrix:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\left[1 + \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{||\boldsymbol{x}||\,||\boldsymbol{y}||}\right], \tag{2.8}$$

where, $K(\boldsymbol{x}, \boldsymbol{y})$ is the element of the Gram matrix, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are its feature vectors. Thus a value of 1 occurs when $\boldsymbol{x}$ and $\boldsymbol{y}$ are in the same direction, 0.5 when they are orthogonal [10]. The example of a dotplot computed as above is shown in Figure 2.6.

Aren Jansen's work [10] presents five steps to detect diagonal lines segments efficiently:

1. Create binary matrix G' from the dotplot matrix G by applying a similarity threshold δ, where $G(i,j) = 1$ if $G(i,j) > \delta$, $G(i,j) = 0$ if $G(i,j) < \delta$.

2. Apply to G' a diagonal μ-percentile filter of length Δ with 45° orientation relative to the x-axis, thus only diagonal line segments of suitable length and density will pass.

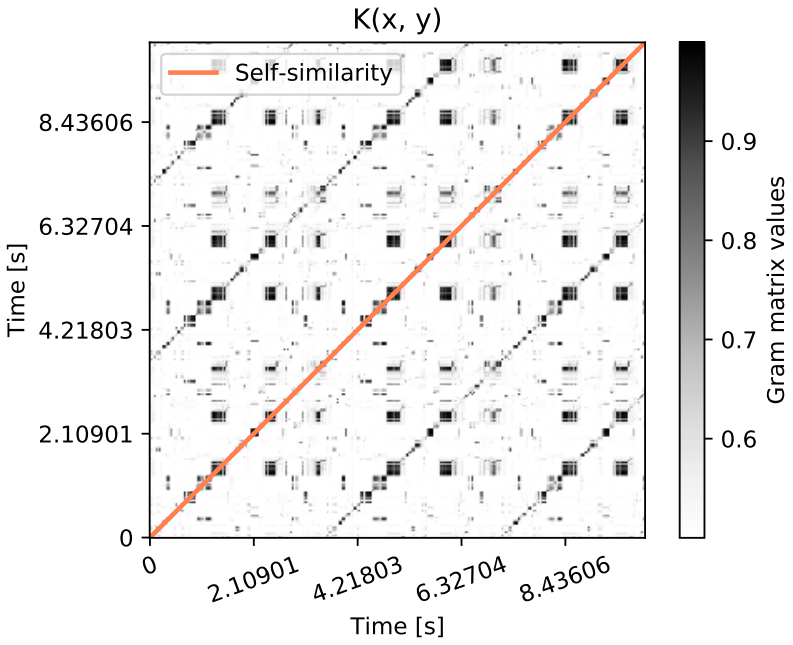


Figure 2.6: Posteriogram dotplot computed on the telephone operator file "If You would like to make a call please hang up and try again" which is repeated 2.88 times, thus it is a 10.06 seconds long recording.

3. Apply to G' a Gaussian filter of width σ with orientation orthogonal to the 45° line segments, thus 45° lines relative to the x-axis of the image. This operation provides variation in speaking rate thanks to allowing deviation from the target similarity lines. The results after these three steps are shown in Figure 2.7

4. Apply to a filtered G' a one-dimensional Hough transform, where r is varied and θ is $-45°$. This step creates a projection onto the line $y = -x$.

5. Define rays from the created peaks from step 4 to search for line segments. Since the peak in the middle corresponds to self-similarity and the Hough transform is symmetric, only peaks on the left from the self-similarity peak are considered.

Using the steps above, the $O(n^2)$ space constraints are decreased, since the thresholded dotplots tend to be exceedingly sparse. However, this approach is not sufficient as a natural prosodic variation of the same term among repeated utterances can break our assumption of the line segment [10]. However, in re-occurring sequences like pre-recorded messages, the prosodic variation stays the same – the messages differ only in speed. S-DTW algorithm from [20] was designed for a more intensive search for curves in dotplots.

2.2.2 Recurrence quantification analysis

Recurrence quantification analysis (RQA) is a method of nonlinear data analysis. RQA calculate the value of path alignments by dynamic programming [30]. RQA provides objective quantification of important aspects revealed by the plot – recurrence matrix (RM). Points in a RM that form diagonal line segments are considered to be deterministic (apart from the isolated points) [37]. The method is similar to DTW. However, instead of finding the minimised alignment path, in the RQA analysis, the longest alignment paths are selected.

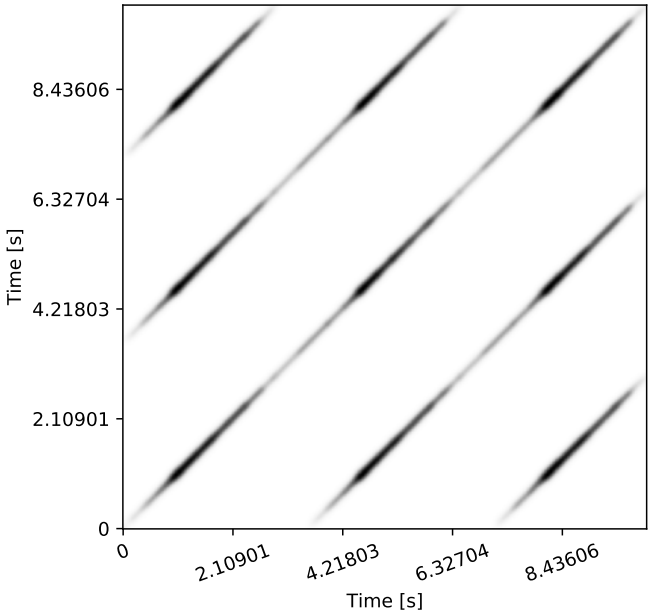


Figure 2.7: Dotplot from 2.6 after steps 1-3 are applied. The constants were set to: $\delta = 0.7$, $\mu = 70$, $\Delta = 100$, $\sigma = 5$.

As mentioned in [12], RQA quantifies the structure of RM by several metrics:

- *Recurrence Rate* – provides the ratio of the number of recurrent points to the square of the length of the time series.
- *Determinism* – gives the percentage of recurrent points that fall on diagonal lines, while ignoring the main diagonal.
- *Number of Lines* – count of the sequences of length at least 2.
- *Average Line Length* – measures the average length found in RM.
- *Maximum Line Length* – captures the length of the longest diagonal sequence detected in RM. This measure provides information about the stability and a theoretical connection to the larger dynamical systems literature.
- *Entropy* – presents a measure of the stability of the structures in RM. In the case of a completely ordered system, entropy reaches its minimum (maximum in the case of randomness).

2.2.3 Two-pass approach

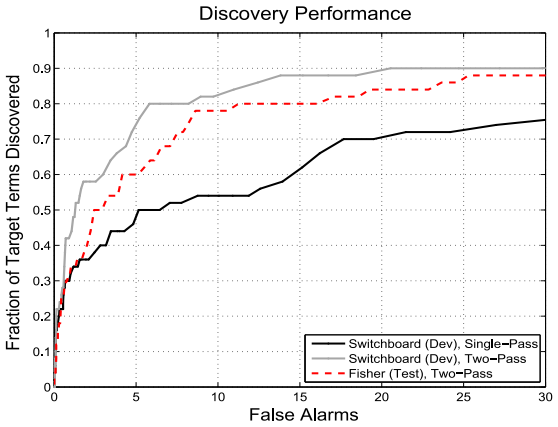
If the curve is acceptably similar to a diagonal, repeated spoken utterances are identified by S-DTW even though the spoken terms have different speaking rates and stress patterns. However, this bigger accuracy comes at a significant computational cost. The solution is to use a two-pass approach. This strategy includes using the optimised image filter method first and then application of S-DTW for each detected line segment [10]. The second pass strategy includes additional S-DTW parameters as follows:

- Set the highest allowed deviation R of the optimal warp path from the diagonal line.
- Take into account the first-pass line segment as the optimal warp path passes through it. Start from the segment midpoint and compute S-DTW both forward and backward in time. Stop growing the match when the path integral of $\left[1 - K(\boldsymbol{x}_i, \boldsymbol{x}_j)\right]$ exceeds a given dissimilarity threshold B.
- Remove all initial or final points in the optimal warp path that have lower similarity than given threshold T.

In [10], the target term is chosen by using the transcription of the speech recordings. The system achieves the following ROC curve for the set variables as in Figure 2.8. The two-pass approach clearly shows improved performance.

Parameter	Single-Pass	Two-Pass
Posteriorgram threshold, ρ	0.1	0.1
Percentile filter threshold, μ	40%	55%
Dotplot threshold, δ	0.65	0.0
Percentile filter length, Δ	0.5 s	0.5 s
Gaussian filter width, σ	30 ms	30 ms
Max S-DTW deviation, R	–	10
Dissimilarity budget, B	–	15
Path trimming threshold, T	–	0.6

a)



b)

Figure 2.8: Parameter values a) used for the single-pass and two-pass versions of the discovery. ROC curves b) for the development (Switchboard data) and test (Fisher English corpus) tasks. Taken from [10].

2.3 Spoken term detection

Spoken term detection (STD) is a common approach to speech search systems. STD can be described as a search of a given phrase in speech data and showing the list of detections with timestamps.

STD is a similar task as searching the re-occurring pre-recorded messages. They both search for occurrences of queries in speech. However, the difference between the tasks is that the same pre-recorded message preserves same prosodic features everywhere, it varies in speed only, what is not the case in STD. Additionally, in the unknown messages scenario, the pre-recorded message – query is not given and must be identified first.

The query can be entered either by text (Query-by-Text, QbT STD) or by spoken term (Query-by-Example, QbE STD). QbT systems require lots of resources like transcribed data, phone sets or additional linguistic resources. Thus it is language-dependent and available only for the most widespread languages. On contrary, QbE STD systems can be language-independent as the query input is in the spoken form. So it is a viable option

for low-resource languages. QbE STD systems search in phone posteriors or any suitable features [5].

For searching the pre-recorded operator message occurrences in a large dataset, the QbE STD approach is obviously more appropriate.

Both techniques can be combined together. The scenario consists of entering the text query first by QbT STD system and as the searching goes, the detected hits are used as queries for the QbE STD systems. The lists of detections from both systems are then merged, bringing new detections or more precise hits of already searched terms. The shown technique is known as *relevance-feedback* mentioned in [3].

Fapso's work [5] mentioned three main methods of matching techniques for QbE STD:

- Template matching of features by using dynamic time warping (DTW)
- Sequential statistical modeling of features by using *Gaussian mixture model* (GMM) and *Hidden Markov model* (HMM)
- Lattice matching by using weighted finite state transducers (WFST)

Audio Pre-processing. Audio pre-processing is an essential step for decent features. For processing the digital signal, *voice activity detection* (VAD) is applied in the first place. VAD is used to remove non-speech parts in given audio to get a proper estimation of the speaker normalisation parameter in the following step.

Next, vocal tract length normalisation and speaker mean normalisation are applied to the speech recordings. Finally, the features are extracted from the processed audio. We can choose either 3-state phone posteriors or *bottleneck features*, which is the last hidden compression layer of a neural network (NN) trained for phone recognition [5].

Template Matching of features. Distance matrix of given metrics is computed between query and utterance feature vectors. Afterwards, the shortest DTW paths crossing the whole query and any part of an utterance are chosen from the distance matrix [8].

Sequential statistical modelling. By using the HMM/GMM techniques, the query model is trained on query features, while the background model is trained on features of all terms in the dataset. Utterance features and their likelihood ratio in each frame are matched against the created query model and the background model which returns the confidence score of the query in the frame [33].

Lattice matching. Previous techniques used just phone posteriors features when searching for query hits. For lattice matching, phone lattices or phone confusion networks are created from the phone posteriors.

2.4 Fuzzy string matching

Fuzzy string matching (FSM) is an important scientific task that has attracted much interest as key information can be expressed by symbolic sequences. When comparing two strings, it is usually straightforward – comparing character by character and declaring them as equal or not. However, a small spelling mistake could ruin the search. There are many use cases when it is desirable to know how similar one string is to another, such as text retrieval,

signal processing, and computational biology. For instance, "I ate a fresh green apple." is similar to "He eats fresh green apples." at the first sight, but not for the computer.

Levenshtein distance

The decent solution for quantifying the similarity is the *Levenshtein Distance* (LD) (also known as *"edit distance"*). It compares strings by several edit operations, such as deletion, insertion, and substitution of individual symbols. LD can be defined as the minimum cost of converting one string into another by using a sequence of edit operations [36].

The edits could be described as transmission errors. Assume a string $\boldsymbol{X}_s \in H$ is the input to a noisy channel and $\boldsymbol{Y}$ is the output string. The transmission errors is described as follows:

- $\boldsymbol{Y}$ has a *substitution* error if $\boldsymbol{X}_s = \alpha a \beta$ and $\boldsymbol{Y} = \alpha b \beta$.
- $\boldsymbol{Y}$ has a *deletion* error if $\boldsymbol{X}_s = \alpha a \beta$ and $\boldsymbol{Y} = \alpha \beta$.
- $\boldsymbol{Y}$ has an *insertion* error if $\boldsymbol{X}_s = \alpha \beta$ and $\boldsymbol{Y} = \alpha a \beta$ [11].

The Levenshtein distance $LD(\boldsymbol{X}, \boldsymbol{Y})$ between two strings $\boldsymbol{X}$, $\boldsymbol{Y}$, each of length $|\boldsymbol{X}|$ and $|\boldsymbol{Y}|$ is defined as:

$$LD(\boldsymbol{X}, \boldsymbol{Y}) = \begin{cases} |\boldsymbol{X}| & \text{if } |\boldsymbol{Y}| = 0, \\ |\boldsymbol{Y}| & \text{if } |\boldsymbol{X}| = 0, \\ LD(tail(\boldsymbol{X}), tail(\boldsymbol{Y})) & \text{if } \boldsymbol{X}[0] = \boldsymbol{Y}[0], \\ 1 + \min \begin{Bmatrix} LD(tail(\boldsymbol{X}), \boldsymbol{Y}), \\ LD(\boldsymbol{X}, tail(\boldsymbol{Y})), \\ LD(tail(\boldsymbol{X}), tail(\boldsymbol{Y})) \end{Bmatrix} & \text{otherwise,} \end{cases} \tag{2.9}$$

where the *tail* of string $\boldsymbol{X}$ or $\boldsymbol{Y}$ includes all characters except the first one. The First element is the minimum corresponding to deletion edit, the second corresponds to insertion and the last one to substitution [9].

2.5 Evaluation of detection

The choice of the right evaluation metric is the key to get a reliable system. We present several metrics for STD and hence pre-recorded message detection. A detection list with a confidence score, start time and end time provided by the system is needed for evaluation and computing the given metrics.

Several scenarios happen in the decision making process of the system. The system marks the given segment as containing the pre-recorded telephone operator message or not. This may lead to either right decision or not. When the recording with the message is marked by the system correctly, we talk about "*hit*", in the opposite scenario it is called "*miss*". The system may mark the recording without the operator message incorrectly, in this case it is a"*false alarm*". Figure 2.9 shows the cases mentioned above.

2.5.1 DET

Detection error trade-off (DET) is a common technique to show system performances for various operating points. Miss probability P_M and false alarm probability P_{FA} are shown

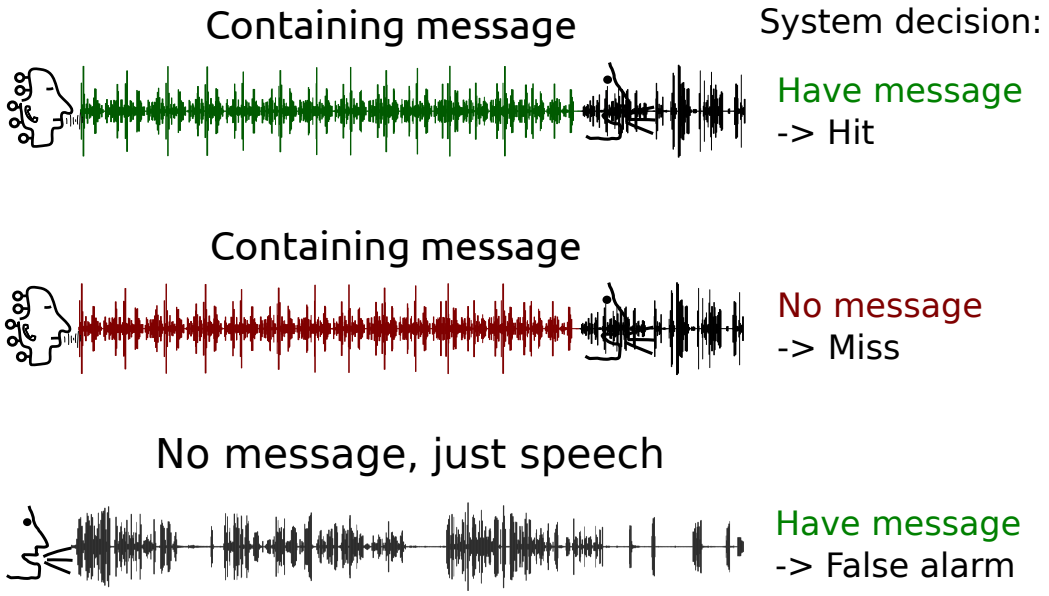


Figure 2.9: Three possible system decisions.

on axes in the DET curve, which provides information on both error types. The used scale for both axes is normal deviate scale so that the DET curves are linear for most detection systems. The closer the DET curve is to the lower bottom corner, the better performance the system provides [14].

For a given threshold θ DET is defined as a dependency of miss probability $p_M(\theta)$ and false alarm probability $p_{FA}(\theta)$. DET curve does not give a single value, but provides a more complex graph of system's performance for various operating points [5]. The DET scheme is shown in Figure 2.10.

2.5.2 ROC

A *receiver operating characteristic* (ROC) curve is a graphical plot similar to the DET curve showing the diagnostic ability of a binary classifier system as discrimination threshold is changing. An ideal cut-off value is almost always a trade-off between hits and false alarms.

The ROC curve shows a graphical illustration of trade-offs at each cut-off for any diagnostic test that uses a threshold or any continuous variable. The better system performance is reflected in the ROC curve by getting it closer to the top left corner. The area under the ROC curve (AUC) is calculated to measure the system performance, where $AUC \in [0, 1]$, $AUC \in \mathbb{R}$. Thus, the best cut-off value provides both the highest hits score (to 100%) and the lowest false alarms score (to 0%) [4]. The ROC scheme is shown in Figure 2.10.

2.5.3 EER

Equal error rate (EER) represents the percentage for the given threshold θ_{EER} where the number of missed detections are equal to the same amount of false alarms. The lower the equal error rate value is, the higher the accuracy of the system [5].

As the EER is a pooled metric, just one global threshold is applied for all queries. The metric is defined as:

$$EER = \frac{\sum_{Q\in\Delta} N_{target}(Q) - N_{hit}(Q, \theta_{EER})}{\sum_{Q\in\Delta} N_{target}(Q)}, \tag{2.10}$$

where Q is the query and Δ is the set of all queries. The number of false alarms is given by expression $\sum_{Q\in\Delta} N_{target}(Q) - N_{hit}(Q, \theta_{EER})$. The scheme depicting the ERR line is shown in Figure 2.10.

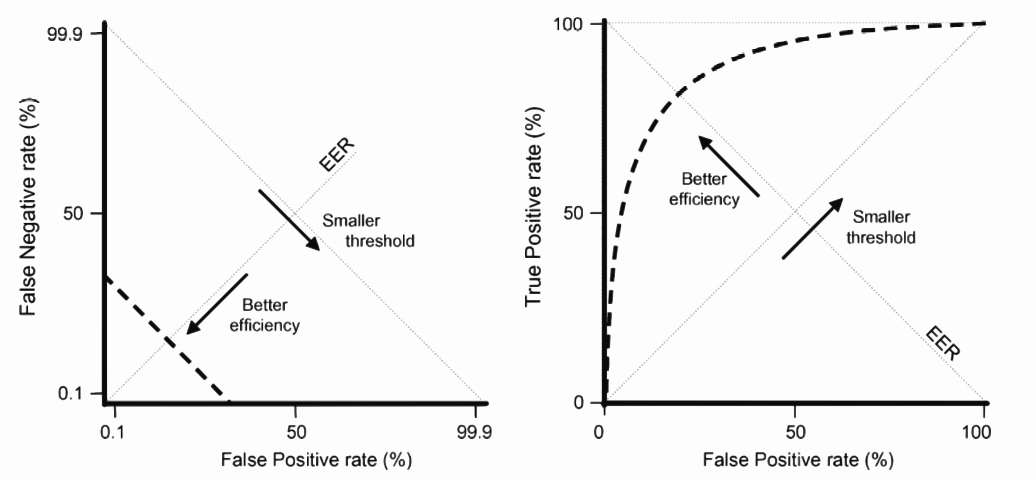


Figure 2.10: Scheme of DET (left) and ROC (right) curves showing the ideal systems as well as EER line. Taken and vectorised from [27].

2.5.4 Minimum DCF

Minimum detection cost function $MinC_{det}$ is a metric similar to equal error rate (EER). $MinC_{det}$ is not a measure of calibration, but of discrimination. It is defined as a weighted sum of the probabilities of miss $\alpha(\theta)$ and false alarm $\beta(\theta)$ at a given threshold θ [18]:

$$CF_{det}(t) = C_{miss} \times \alpha(\theta) \times P_{target} + C_{false\,alarm} \times \beta(\theta) \times (1 - P_{target}), \tag{2.11}$$

The parameters C_{miss} and $C_{false\,alarm}$ are the relative costs of detection errors, and the parameter P_{target} is the a-priori probability of the specified model speaker [35]. Unlike EER it is dependent on the particular application-dependent parameters of C_{det}. In the means of the NIST Speaker Recognition Evaluation, it is usual to indicate C_{det}^{min} on DET-curves [32].

2.6 Evaluation of clustering

Clusters are necessary components in the classification process, and thence an evaluation of clustering is essential to obtain better performance. In this section, a few evaluation metrics of clustering are presented. In *external evaluation*, clustering results are evaluated based on known class labels – known as *reference clustering*. For example, let assume the predicted cluster $P = [0, 0, 1, 2, 2, 0]$. To evaluate the performance of P, we use reference clustering $R = [0, 0, 1, 1, 2, 2]$. Several measures are adapted from variants used to evaluate

classification tasks. Instead of counting how many times a class is correctly assigned to a single data point – *hits*, metrics for pair counting assess whether each pair of data points that is truly in the same cluster is predicted to be in the same cluster [23].

Purity (Section 2.6.1) is a simple and transparent evaluation measure. The *Rand index* (Section 2.6.2) penalises both false positive and false negative decisions during clustering. *Normalised mutual information* (Section 2.6.3) can be information-theoretically interpreted [13].

2.6.1 Purity

Purity is a measure showing "pureness" of the dominant class in the cluster in proportion to the other classes. To compute purity, let $\Omega = \omega_1, \omega_2, \ldots, \omega_k$ be the set of clusters and $C = c_1, c_2, \ldots, c_j$ be the set of all classes. For each cluster of Ω a class from C, which is the most frequent in the cluster, is assigned. Then the accuracy is computed by counting the number of correctly assigned classes and dividing by the total number N of all data points [13]:

$$purity(\Omega, C) = \frac{1}{N} \sum_{k} \max_{j} |\omega_k \cap c_j|. \tag{2.12}$$

The main drawback of purity is giving a high score even for poor clusters, as purity does not penalise having more clusters. Score of one can be accomplished by putting each data point into its own cluster. Using purity is not suitable to trade-off the quality of the clustering against the number of clusters. A measure that allows us to monitor this trade-off is *normalised mutual information* (NMI) (Section 2.6.3). A trivial example of purity is presented in Figure 2.11. The Purity score of one represents the best performance, zero the worst.

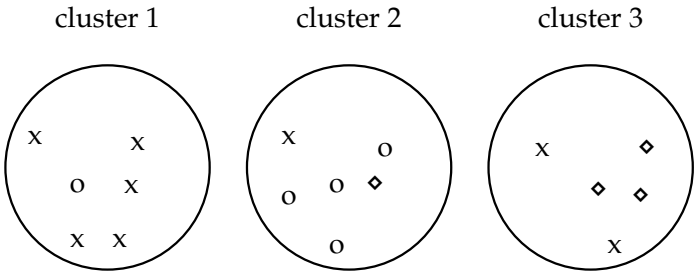


Figure 2.11: Purity – a measure scoring cleanliness of the clusters. For the given clusters, purity is computed as $(1/17) \times (5 + 4 + 3) \approx 0.71$. Taken from [13].

2.6.2 Rand index

Rand index (RI) shows a similarity between the reference clusters and the clusters returned from the algorithm. Two cases are measured: i) a *true positive* (TP) decision assigns two similar data points to the same cluster and ii) a *true negative* (TN) decision assigns two different data point to different clusters. Thus, two types of errors can occur: i) a false positive (FP) decision assigns two different data points to the same cluster, ii) a false negative (FN) decision assigns two similar data points to two different clusters. The result of Rand index is the percentage of correct decisions. Rand index score of one represents the best performance, zero the worst.

Rand index is computed as follows:

$$RandIndex = \frac{TP + TN}{TP + FP + FN + TN}, \tag{2.13}$$

where TP is the count of true positive cases, TN is the true negative cases, FP is the number of the false positive cases and FN is the false negative cases. For the example in Figure 2.11 the RI is computed as $(20+72)/(20+20+24+72) \approx 0.68$. The main drawback of Rand index is that FP and FN are weighted equally.

2.6.3 NMI

Normalised mutual information (NMI) solves the main drawback of the purity measure. NMI quantifies how much information is shared between the reference clustering and the predicted clusters, and has a reduced bias for varying cluster numbers [13]. Let $\Omega = \omega_1, \omega_2, \ldots, \omega_k$ be the set of clusters and $C = c_1, c_2, \ldots, c_j$ be the set of all classes, then NMI is computed as:

$$NMI(\Omega, C) = \frac{MI(\Omega, C)}{\frac{1}{2}\big(H(\Omega) + H(C)\big)}, \tag{2.14}$$

where MI is mutual information calculated from equation 2.15 and H is an entropy computed from equation 2.16.

Mutual information is calculating the probability of being in a cluster ω_k, class c_j and its intersection for each data point, where N is the total count of all data points:

$$MI(\Omega, C) = \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}. \tag{2.15}$$

Let $X = x_1, x_2, \ldots, x_i$ be the set of the monitored phenomenon, and N is the total count of points in X, then entropy H is computed as:

$$H(X) = -\sum_i \frac{|x_i|}{N} log \frac{|x_i|}{N}. \tag{2.16}$$

Without normalising NMI by the entropy in the denominator, the mutual information has the same drawback as purity – it does not penalise large cardinalities. As NMI is a normalised measure, it is possible to compare clusters with different cluster count [13]. NMI score of one represents the best performance, zero the worst.

Chapter 3

Dataset

This chapter will describe the dataset used to evaluate the implemented methods. The dataset that completely fits our needs was not found, and it is hard or nearly impossible to obtain real, either from the investigator team or commercial data. Therefore, the goal is to get as close as possible to a real situation. I accomplished that by mixing the collected pre-recorded telephone operator messages into standard publicly available phone calls.

3.1 Switchboard dataset

DARPA Switchboard Telephone Conversation corpus – the second release from August 1997 is used as a base for the simulated dataset. The Switchboard-1 Telephone Speech Corpus was originally collected by Texas Instruments in 1990-1, under DARPA sponsorship. The first release of the corpus was published by the National Institute of Standards and Technology (NIST) and distributed by the Linguistic Data Consortium (LDC) in 1992-3. In the second release, assembled and published by the LDC, all known errors affecting the original publication of speech files have been corrected[1].

The total duration of the telephone conversations is **517.13** hours with a total of **4870** conversations. The files are in 16-bit wave format with a sample rate of **8000 Hz**. Duration of a call is from 5 to 10 minutes. length. The recorded phone calls are mono, each file has channel A from the called person perspective and channel B from the caller side, both with the same duration. This approach explains why the silent part occurs in the recorded conversation – a person from the opposite side is always quiet. This approach makes a more significant impact on the recogniser because of existing deviations in the recordings[2]. Each file is labelled as follows: `swXXXXX-Y` where `XXXXX` is the record ID and `Y` is either A or B, which labels different speaker. The same labels go for the corresponding transcription of the conversation.

As the recordings are longer than the typical average call length, the files are trimmed into smaller parts. For that purposes, the open-source Python library **Pydub**[3] is used. The recordings were cut as shown in table 3.1. The goal was to cut them equally in the sense of the duration – each category, is in terms of the total length, approximately 60 hours long.

This process extends the original labelling of the file, as it adds information about the length and the part corresponding to the original recording. This changed the original

[1] LDC Switchboard-1 Release 2: https://catalog.ldc.upenn.edu/LDC97S62
[2] However, a phenomenon like *cross-talks* can occur
[3] Pydub (used under MIT license): https://github.com/jiaaro/pydub

`swXXXXX-Y` format to `swXXXXX-Y_Z_D`, where `Z` is the part of the original `XXXXX-Y` and `D` is the duration of the trimmed recording in seconds – the 180-second-long recording trimmed to 3 parts will be labelled as follows: `sw00000-A_1_60`, `sw00000-A_2_60` and `sw00000-A_3_60`.

files length [s]	**count**
180	1,409
120	2,476
90	3,391
60	3,532
45	4,639
30	5,940
20	10,843
15	10,120
<15 (leftovers)	4,870
total	*47,220*

Table 3.1: The lengths and the total counts of the files from the Switchboard corpus after trimming.

3.2 Telephone operator pre-recorded messages

To create a simulated dataset for this thesis, the telephone operator pre-recorded messages were collected first: a total of **26 unique recordings** either downloaded from the internet or recorded from real telephone conversations. The messages in English (*15* in total) are downloaded from *Soundsnap*[4] under Soundsnap license, or *Storyblocks*[5] under Royalty-Free license agreement. The messages (in both English and Czech or English and Slovak – *11* in total) were recorded from the actual calls via *Cube ACR*[6] app using *Nokia 7.2* with Android 10. The pre-recorded messages were split into three categories:

- Samples of *category A* are messages that should appear only at the beginning of the call. (example: "Thank you for calling, please leave a message").
- *Category B* messages could appear anywhere through the call. (example: "Please hold for a very important message").
- *Category C* messages should appear both at the beginning and the end of the call. (example: "Sorry the number you are calling just not answer at the moment, please try again later").

Messages are spoken by male or female speakers. Some of the messages carry the same content but differ in gender. The initial silence is removed by open source software *Audacity*[7] and messages are normalised to same volume as the phone calls from the Switchboard corpus. The message is labelled in the following format: `XYZ`, where `X` is the category (A, B or C), `Y` stands for language (0 – English, 1 – Czech+English, 2 – Slovak+English) and "Z" is the number (0 – 9) of the file.

[4] Soundsnap license: https://www.soundsnap.com/licence
[5] Royalty-Free license agreement: https://www.soundsnap.com/licence
[6] Cube ACR on Google Play: https://play.google.com/store/apps/details?id=com.catalinagroup.callrecorder
[7] Audacity: https://www.audacityteam.org/

3.3 Mixing pre-recorded messages into Switchboard

The next step is to mix the telephone operator messages with the phone calls from the Switchboard corpus. First, approximately 10% of the trimmed phone calls were chosen – 4260 calls. Also, 200 recordings of zero length calls were chosen – in total **4460** calls.

To get close to realistic data, each pre-recorded message is mixed into calls with varying speed, volume gain and varied repetition. The task is accomplished using *Pydub* library. Based on the advice of my supervisor, I selected the length of a mixed pre-recorded message to be around one minute. The changes of mixed pre-recorded messages are: in gain between -6dB – +6dB, speed between 0.9 – 1.1 of the original speed, repetition between 0.8 – 30.0. The position of the mixed message in the phone call depends on the category of the pre-recorded message. Type A is at the beginning of the Switchboard recording, B is somewhere in the middle and type C is at the end and the beginning of the conversation.

The total count of the mixed pre-recorded messages is **4460** files of a total length of **150.66 hours** (of which 99.23 hours is the length of clear messages without the phone calls conversation). Mixed recordings use extended labelling with info as follows:

`swXXXXX-Y_Z_W__AAA_ST(BB.BB)L(CC.CC)G(DD.DD)R(EE.EE)S(FF.FF)`, where

- Part before two underscores is the voice call recording where the pre-recorded message is mixed.
- `AAA` – ID of the used pre-recorded message.
- `ST(BB.BB)` – `BB.BB` marks position where the message starts in the file in seconds.
- `L(CC.CC)` – `CC.CC` is the length of the message in seconds.
- `G(DD.DD)` – `(-)DD.DD` is the volume gain [dB] – values can be negative.
- `R(EE.EE)` – `EE.EE` represents how many times the message is repeated.
- `S(FF.FF)` – `FF.FF` is the speed of the message compared to the original.

The prepared recordings were split into training and evaluation sets and their subsets used for development – total numbers are presented in table 3.2. Each of the sets contains representatives of every pre-recorded telephone operator message – just in a different ratio. With intention of creating a universal dataset for varied techniques of machine learning (like training a NN), "Train" and "Sub train" sets are created but not used in evaluations.

Database	Raw phone calls [hours]	Mixed calls with messages [hours]	Total [hours]	Total (raw + mixed) [count]
Train	339.31	117.63	**456.94**	32377+3576=**35953**
Eval	126.62	37.10	**163.72**	10583+884=**11467**
Sub train	16.83	30.62	**47.45**	697+697=**1394**
Sub eval	4.31	6.48	**10.79**	187+187=**374**

Table 3.2: Total lengths [hours] and counts of audio files in individual sets.

Chapter 4

Used tools

All the created systems and additional scripts were written in *Python 3.8* with *Numpy 1.20.1* [7] and *SciPy 1.4.1* [34] on a machine running *Ubuntu 20.04 LTS* with 16 GB of RAM and Intel Core i7-8565U CPU. All of the graphic content of the code output is done by *Matplotlib 3.3.2* [2].

4.1 Feature extraction tools

Feature extraction is essential for speech processing. For the work, I used standard MFCC features, phoneme posteriors and bottleneck features.

Phoneme recogniser

Phoneme recogniser based on long temporal context [29] developed at Brno University of Technology (BUT)[1] is used for extracting 3-state phoneme posteriors from audio wave file and producing phoneme strings (and phoneme lattices via an additional script). I used PHN_CZ_SPDAT_LCRC_N1500 system (Czech 8 kHz models), the phoneme recogniser works with 25 ms frames with 10 ms shifting and provides 138 dimensions array – 3 states for each of the 46 classes. Its core is written in C language, using mathematical library ATLAS/BLAS. Each audio wave file is first processed, and the posterior feature vector is then saved in HTK file format or to a text file in case of phoneme string.

Bottleneck feature extractor

For extracting bottleneck feature vectors, we used the BUT/Phonexia Python package[2] [6]. The script extracts bottleneck features and phoneme posteriors from audio files. From three possible choices of the pre-trained neural networks, the one trained on Fisher English – "FisherMono" was chosen. The extracted bottleneck feature vector is 80-dimensional. The processed array is stored in HTK file format.

[1] Phoneme recogniser based on long temporal context: https://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context

[2] BUT/Phonexia Bottleneck feature extractor: https://speech.fit.vutbr.cz/software/but-phonexia-bottleneck-feature-extractor

MFCC feature extractor

The standard 13 MFCC features are extracted from an audio wave file, with the default settings – 25 ms window frames with 10 ms shifting and with 512 being the size of FFT. Extraction is done by using python_speech_features[3] package in version 0.6.

4.2 Algorithm tools

Fast DTW

The dynamic time warping algorithm is used for template matching of two time series of feature vectors. As the standard DTW comes with a higher computational cost, Fast-DTW [28] with available Python package FastDTW 0.3.4[4] provides a decent solution.

By using FastDTW, faster computation is achieved as well as less demand on the system resources. The longer the time series, the faster the calculation is compared to basic DTW. However, on shorter time series, the standard DTW provides faster and more accurate computation. Thus, it is necessary to decide which approach is more suitable for the given task.

Librosa library

Librosa is a Python package for music and audio analysis. It provides the essential building blocks to create audio information retrieval systems [15]. For the thesis, Librosa in version 0.8.0[5] is used. The library is used for loading an audio file, computing recurrence matrix and RQA analysis (in Section 2.2.2). Because of the limitations of Fast DTW on shorter sequences, Librosa is chosen for calculating DTW distance and alignment path, as it has better performance and more accurate results on shorter time series.

FuzzyWuzzy

FuzzyWuzzy[6] is a Python package used for approximate string matching, also known as fuzzy string matching. This tool uses Levenshtein distance to calculate the differences between sequences while providing a simple-to-use interface. The implementation offers several types of string comparison, such as partial ratio, which finds similarities in sub-strings.

S-DTW

Segmental DTW is a modification of a standard DTW as explained in 2.1.3. An implementation according to [21] is available on GitHub[7]. I have edited the code to work with different distance metrics, and the output of the script is modified to return also a warping path.

[3] Python package python_speech_features: https://python-speech-features.readthedocs.io/en/latest/
[4] FastDTW on PyPI: https://pypi.org/project/fastdtw/
[5] Librosa 0.8.0 on Zenodo: https://zenodo.org/badge/6309729.svg
[6] FuzzyWuzzy 0.18.0 on PyPI: https://pypi.org/project/fuzzywuzzy/
[7] S-DTW implementation by gray0302 on GitHub: https://github.com/gray0302/seg-dtw

VAD

Voice activation detection (VAD) is used to classify whether a given piece of audio data is active or inactive. The used py-webrtcvad[8] for VAD developed by Google for the WebRTC project is reportedly one of the best available VADs, being fast and free. The input for VAD is an audio wave file with at least of 8000 Hz sample rate. The utility is set to recommended options with "aggressiveness" of level 2 and frame size of 10ms.

4.3 Evaluation Algorithms

Evaluation metrics are used to compare performance of different systems and its components. Existing tools were used:

Scikit-learn

Scikit-learn[9] provides simple and efficient tools for predictive data analysis. It is an open source project built on NumPy, SciPy, and matplotlib[10] (used for visualising) [22]. This package is used for computing clustering purity, Rand index, normalised mutual information, and for finding the best threshold of the created systems.

Pytel

Pytel is a Python package written by BUT Speech@FIT group members. Pytel is used to comparing the performance of created systems by using DET curves, EER and MinDCF. The package is chosen because of its simplicity and intuitiveness.

[8] py-webrtcvad on GitHub (used under MIT license): https://github.com/wiseman/py-webrtcvad
[9] Scikit-learn 0.24.1 on PyPI: https://pypi.org/project/scikit-learn/
[10] Matplotlib 3.3.2 on Zenodo: https://zenodo.org/record/4030140#.YHjTH1XaiV4

Chapter 5

Baselines

This chapter provides details about the baselines systems used for detecting the pre-recorded messages. Evaluation techniques use two approaches:

- Evaluation by a recording itself – described in Section 1.2.a). This method is used in RQA approach 5.2.
- Evaluation by all files from the set or random chunks of the set – described in Section 1.2.b). This method is used in DTW 5.1 and FSM approaches 5.3.

The results provide binary decision only – either the audio contains the message or not.

Evaluation of the systems is not compared to any existing solution as no public system is known for such a topic.

5.1 DTW approach

DTW aims at is based on finding the similarity between two time series of feature vectors by using the optimal warping path and the DTW distance, more in subsection 2.1.2. DTW gives optimal results when comparing similarly long time series with isolated words. However, in our case, the conditions are not that ideal. The recordings are of various lengths. Also, the similar parts are repeated unevenly, and the position of a pre-recorded message could be anywhere in the file. It follows that the first important step is to find those candidates and compare them. Thus, using the raw DTW distance as the decision-makers will not provide satisfying results. Librosa DTW is not suitable for long time series, therefore FastDTW is used for computing DTW distance and alignment path.

Searching for candidates

To find the candidates for determining similarity, the DTW warping path is computed first and then analysed. The algorithm finds the similarities by looking back to the previous steps the warping path has taken. In a DTW warping path, 3 moves are possible: i) *diagonal*, ii) *horizontal* and iii) *vertical*.

Three types of moves that a two-step pair can get:

- The *"good trend"* (GT) type happens when the current step is in diagonal direction. The direction of the previous step does not matter in this case.

- The *"false trend"* (FT) type happens when the two-step pair consists of two horizontal moves or two vertical moves.
- The *"neutral trend"* (NT) type happens when the two-step pair consists of one vertical and one horizontal moves or one is in the diagonal direction.

The explanation of the moves type is shown in Figure 5.1.

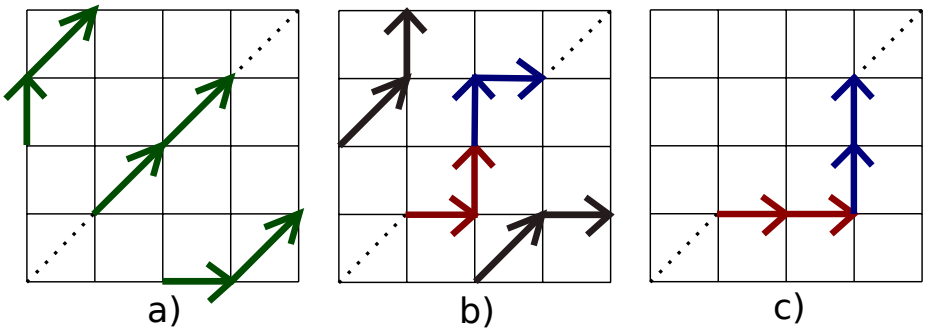


Figure 5.1: Three possible types of a two-step pair in the DTW warping path. Type a) represents a good trend – the current step is diagonal, thus all the green pairs are considered a good trend. Type b) represents the neutral trend – the red and blue pairs are going in both horizontal and vertical direction and the black pairs are of the neutral trend type as well. Type c) represents two possible options of the false trend – either two horizontal moves (red) or two vertical moves (blue).

The idea of the experimental function for finding candidates is to start detecting the similarity when the diagonal move occurs. Then each step is evaluated and the corresponding point is added to the list.

FT type increments the false-trend variable and the constant false trend variable. GT type increments the good-trend variable and resets the variable responsible for monitoring the constant false-trend types. NT type resets only the constant false-trend variable.

The detection ends when the constant false trend is larger than the given threshold σ. Then the quality of the detected part of a sequence is evaluated. First, the length of the list needs to be longer than δ frames, which means $\delta/100$ seconds. The sequences longer than δ are scored by the ratio of the triangle created by the warping path, y-axis and x-axis. If the ratio falls in the given interval τ, the detected line is a good candidate. The whole process repeats until the end of the warping path. Figure 5.2 shows ratio computing. In the presented baseline system, the variables are set accordingly: $\sigma = 25$, $\delta = 200$, and $\tau \in \langle 0.9, 1.1 \rangle$. Formal description of searching for candidates is presented in algorithm 5.1.

Classifying the recordings

The proposed solution for the baseline DTW system is to compare every feature vector with all others. By using a brute-force approach, it is immediately an $O(n^2)$ problem. However, a little trick is performed to get a better constant. The optimisation is achieved by comparing the currently processed file with the number of all recordings reduced by 1/8. For every next file, a new shuffled 1/8 bunch of all files is used. For example, let assume a set of 2000 files. With every evaluated file, a new chunk of 250 files is randomly chosen from

Algorithm 5.1: Formal description of searching for candidates.

Input: DTW alignment path
Result: List of candidates in alignment path

```
previous_point ← None
direction, previous_direction ← None
candidates_list, temp_list ← empty.List()
/* gt=good trend, ft=false trend, cf=constant false                    */
gt, ft, cf ← 0
for point in DTW alignment path do
    if previous_point is not None then
        direction ← getDirection(previous_point, point)
    if temp_list is Empty and direction is DIAGONAL then
        if gt > 5 then temp_list.append(point)
        gt++
        cf, ft ← 0
    if temp_list is not Empty then
        trend ← getTrend(previous_direction, direction)
        if trend is GOOD TREND then gt++ cf ← 0
        if trend is NEUTRAL TREND then cf ← 0
        if trend is FALSE TREND then cf++ ft++
        temp_list.append(point)
    if cf ≥ σ then
        if length(temp_list) ≥ δ then
            if countRatio(temp_list) in τ then candidates_list.append(temp_list)
        gf, ft, cf ← 0
        temp_list.clear()
    previous_direction ← direction
    previous_point ← point
if length(temp_list) ≥ δ then
    if countRatio(temp_list) in τ then candidates_list.append(temp_list)
```

the whole set. As the idea of detecting the pre-recorded messages is to find parts repeated several times, at least one should appear in the reduced bunch. This helps to reduce the processing time by almost 87.5%.

The algorithm runs in two loops – the main loop contains files for classification, – the nested loop compares all the files from the reduced bunch to the one in the main loop. First, a DTW is computed for the raw time series. The warping path is analysed afterwards. The candidates are compared by DTW again. The pair with the minimum DTW distance is compared to threshold ϕ, and the final decision is made. In the case of an empty list from the warping path analysis function, the original DTW distance is compared to threshold ϕ to produce a hard decision. A Euclidean distance is used as a distance matrix for creating a DTW matrix, due to more sparse values, which helps to create a smoother threshold. Standard multiplicative weights for moves are used – $\sqrt{2}$ for a diagonal direction and 1 for both vertical and horizontal directions. Formal description of the process of evaluation is presented in algorithm 5.2.

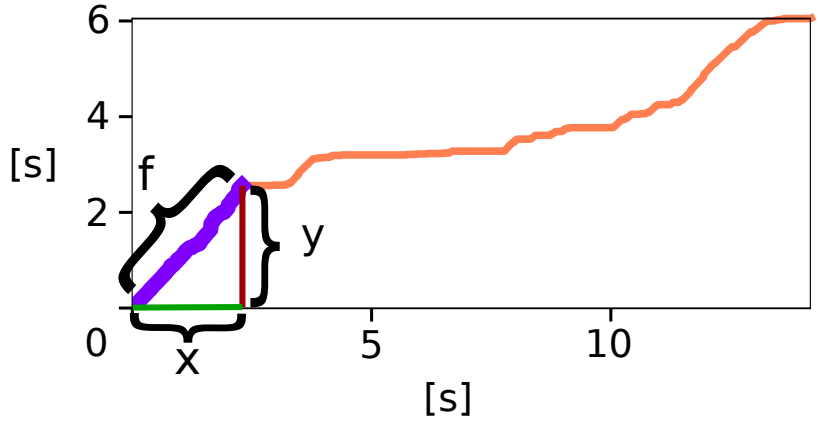


Figure 5.2: The candidate detected on a warping path. First, the number of points in the list – f, must be at least δ points, which represents $\delta/100$ seconds. To confirm f frames are a candidate, a ratio is calculated. $Ratio = y/x$ must be in interval τ, where $\tau \in \langle 1-a, 1+a \rangle$.

Algorithm 5.2: Formal description of DTW classification process.

```
Input: unprocessed files
Result: text file with evaluations
results ← textFile.empty()
for file in unprocessed_files do
    feature_vector ← load(file) if not in cache
    cache.add(feature_vector)
    shuffled_set ← shuffle(unprocessed_files)
    index ← 0
    have_hit ← False
    for nested_file in shuffled_set do
        index++
        nested_feature_vector ← load(nested_file) if not in cache

        cache.add(nested_feature_vector)
        alignment_path, DTW_distance ← DTW(feature_vector,
         nested_feature_vector)
        candidates_list ← getCandidates(alignment_path)
        if candidates_list isnotEmpty then
            DTW_distance ← findMinimumDtwDistance(feature_vector,
             nested_feature_vector, candidates_list)
        if DTW_distance < Φ then
            have_hit ← True
            results.append(file, DTW_distance, "have pre-recorded message")
            break
        if index > 1/8 count(shuffled_set) then break

    if have_hit is False then
        results.append(file, DTW_distance, "no pre-recorded message")
```

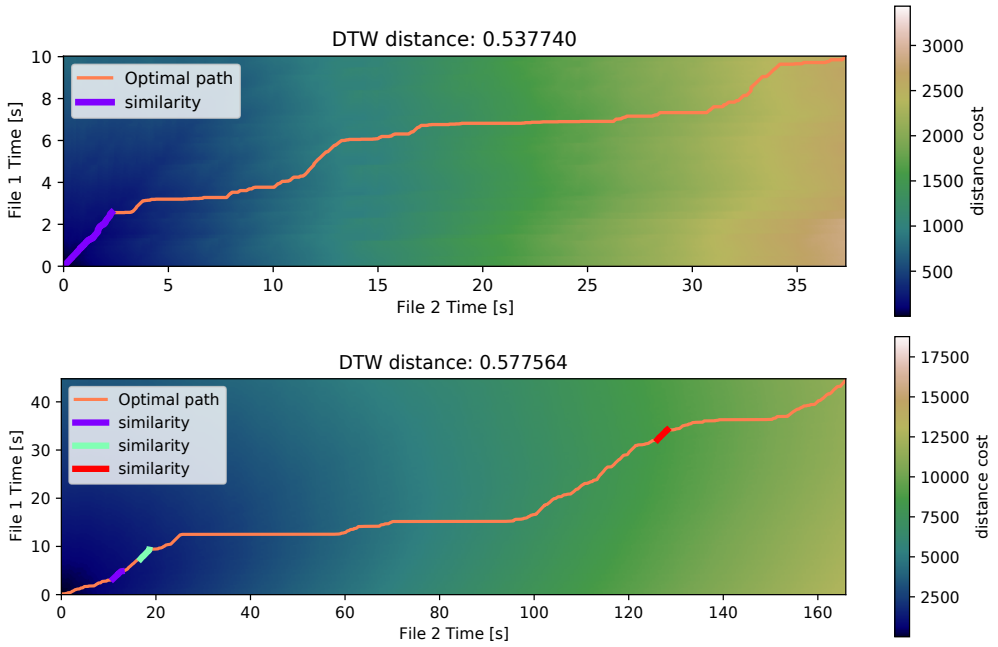


Figure 5.3: DTW distance of two pairs of time series. The top one with one candidate, which is later classified as a hit. The bottom alignment obtains several candidates, but none of them resulted in a hit.

Results

The baseline DTW system provides better results than just random guessing, but its error rate is too high to use in a practical system. The system produces only binary decision without exact timing. The processing time is also higher above the acceptable limit. Due to high processing time, the system is not tested on "Eval" dataset as it would take months on a standard machine. The advantage of this approach is simple parallelisation, and the system uses the unknown messages method. The results on different features as input is shown in table 5.1 and corresponding DET curve in Figure 5.4.

Features	**Average time of one file [seconds]**	**CPU-core time [hours]**	**EER**	**MinDCF**	**The best threshold**
MFCC	244.39	8.46	**41.17%**	0.61	43.28
Phoneme posteriors	284.30	9.79	**30.48%**	0.42	3.96
Bottleneck*	477.49	16.53	**29.41%**	0.50	1.14

Table 5.1: The results of the baseline DTW system on "Sub eval" dataset. The total run-time would be approximately three times longer – the evaluation was split to three CPU threads. *Run with bottleneck features uses chunks of 1/4 instead of 1/8. The best threshold corresponds to get the EER result. MinDCF parameter is set to 0.5, as "Sub eval" has an equal proportion of target and non-target files.

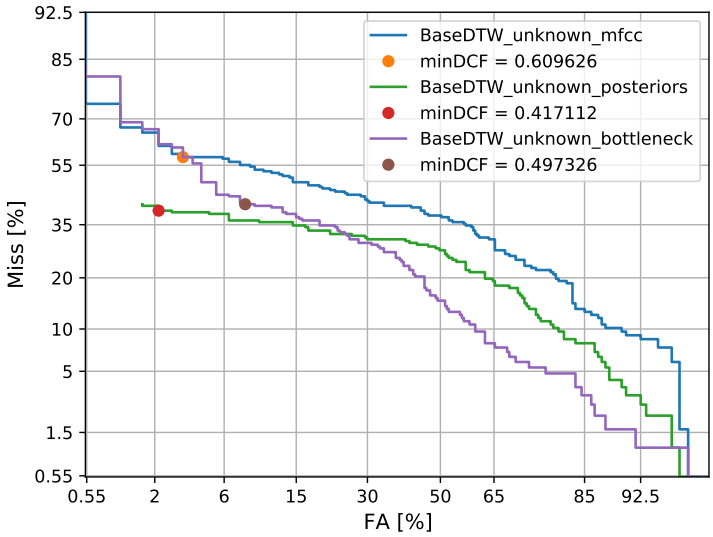


Figure 5.4: DET curve of the baseline system by using MFCC, phoneme posteriors and bottleneck features. "Sub eval" dataset is used.

5.2 RQA approach

Recurrence quantification analysis (RQA) analyses recurrence matrix for diagonal segments 2.2.2.

The RQA approach is based on Aren Jansen's work [10] described in Section 2.2. RQA analysis expects the similarity matrix at the input. The steps of creating such a matrix are presented in 2.2.1. The self-similarity diagonal is removed. The recurrence matrix creation from the Librosa library 4.2 is chosen as my implementation of the first three steps is twice as slow. A cosine distance is used for creating a recurrence matrix, as it is faster and used in various papers as in [5] or [31] as a suitable distance metric. Affinity mode[1] is chosen as it preserves the most information and the number of nearest-neighbours for each file is set to 1/10 of the total frames count.

Classifying the recordings

As the telephone operator messages are often repeated several times within the recording, the recurrence matrix is computed for one file at a time only. This approach allows working in linear time-space of $O(n)$ as each file is analysed only once. Samples shorter than 3 seconds are not included, as short speech conversation should not be present in the analysis for correct computation. Shorter recordings are evaluated with a high penalty score.

Next, the heuristic to filter out false alarms from RQA analysis is to accept the best alignment sequence longer than δ only. Such a heuristic, of $\delta = 2.5$ seconds, helps to reduce false alarms, as those alignment paths usually happen to be only a word or a short inactive part. An unsatisfactory result is evaluated with a large score.

The suitable alignment path from RQA analysis is scored by the sum of the points in the result, divided by the length of the path. Scores below the threshold ϕ are marked as

[1] Affinity mode measures how similar the frames are. Affinity is also known as a self-similarity matrix.

a hit. Threshold ϕ is varied based on features.

To speed up the calculation, frame averaging is performed in the input array as described in Section 6.2. The performance of varied frame averaging values can be seen in table 5.2 and in Figure 5.5.

The lost accuracy is partly solved by using "knight moves" – as in the chess piece. Such a modification allows more possibilities to get a longer alignment path than strict diagonal moves only.

Frame reduction	**1**	**5**	**10**	**20**	**30**	**40**
Average time [seconds]	18.28	1.34	0.56	0.32	0.23	**0.19**

Table 5.2: Computation run-time after 100 repetitions of the whole process of RQA analysis, with different frame averaging optimisation. The input array contains phoneme posteriors, which represents a 37-second recording, or 3733 frames in total.

Results

The optimal balance between processing time and accuracy is achieved by using five frames for frame averaging, as DET curves reveal in Figure 5.6 and table 5.3. The system produces only binary decision without exact timing and the system uses the unknown messages method.

The advantage of this approach is fast processing as it is $O(n)$ time-space. The process can be parallelised. However, in that case, it is not possible to serialise the output list with saved candidates and their frames and scores. The serialised persistent list is later used for further processing.

	Average time [seconds]			Total time [minutes]			EER [%]			MinDCF		
Features / Dataset	M	PP	BN	M	PP	BN	M	PP	BN	M	PP	BN
Sub eval FR=5	2.06	2.56	3.08	12.84	16.95	19.20	**1.60%**	15.50%	16.04%	**0.02**	0.24	0.27
Sub eval FR=10	1.61	2.36	2.11	10.03	14.72	13.17	12.30%	20.86%	22.99%	0.13	0.36	0.38
Sub eval FR=20	**1.05**	1.68	1.55	**6.52**	10.45	9.65	32.62%	28.88%	27.81%	0.45	0.49	0.49
Eval FR=5	**1.92**	2.17	2.91	**365.73**	412.84	544.65	**1.34%**	6.68%	10.38%	**0.16**	0.73	0.93

Table 5.3: Processing time and evaluation results of RQA systems with different settings. Tested on "Sub eval" and "Eval" database. FR is acronym for frame reduction, M means MFCC features, PP means phoneme posteriors, BN stands for bottleneck features.

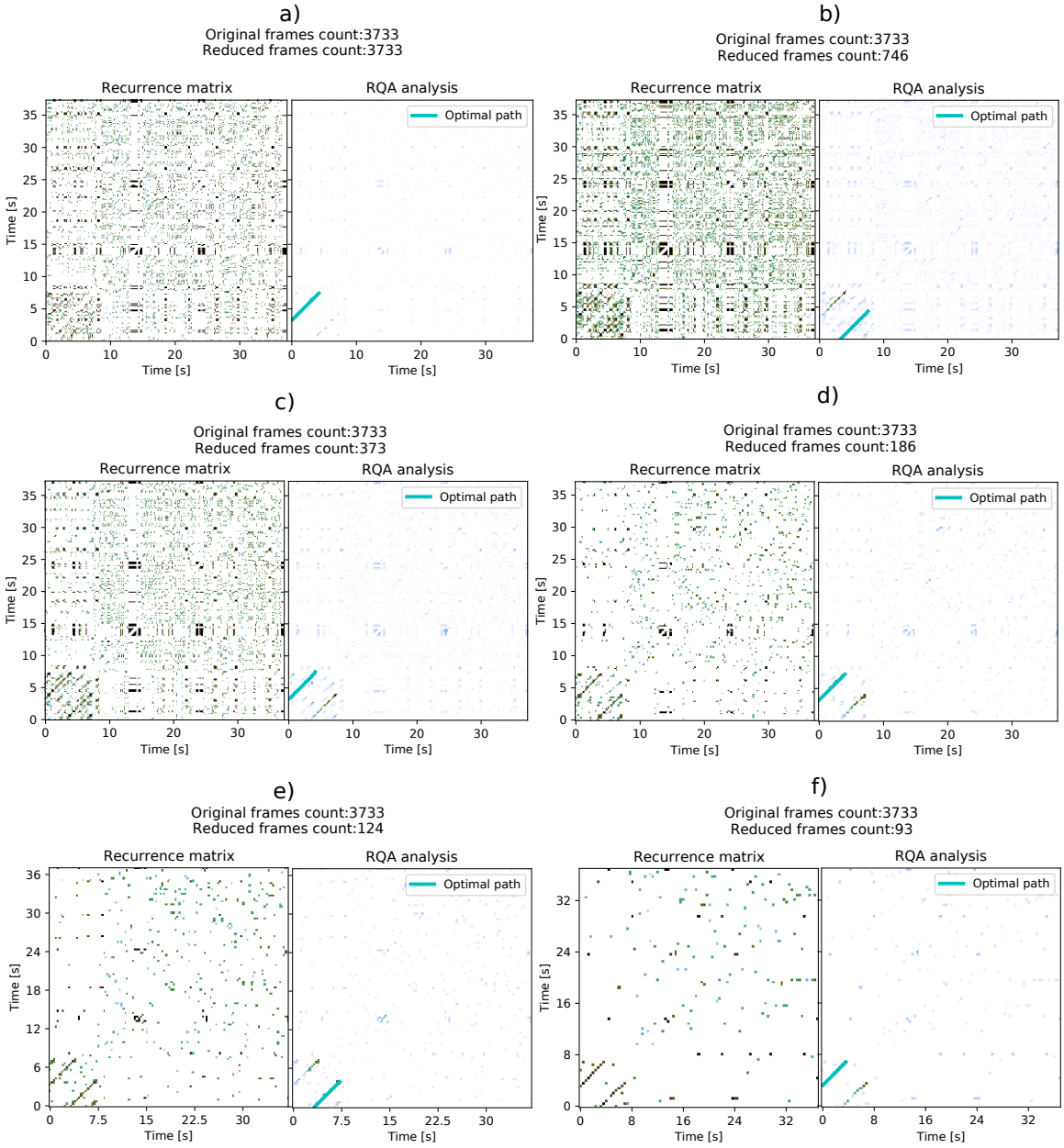


Figure 5.5: Recurrence matrix and RQA analysis visualisation for a 37-second long phoneme posteriors array. The pre-recorded message is in the first 7.36 seconds of the telephone conversation (file: `sw03720-B_5_30__A02_ST(0.00)L(7.36)G(-0.61)R(2.26)S(1.04)`). For each run, different value of frame averaging is used. No reduction for a), frame reduction five for b), c)=10, d)=20, e)=30, f)=40. At first glance, even 40 frames averaging looks accurate and usable, however in a real system, an acceptable frame reduction value is around five frames.

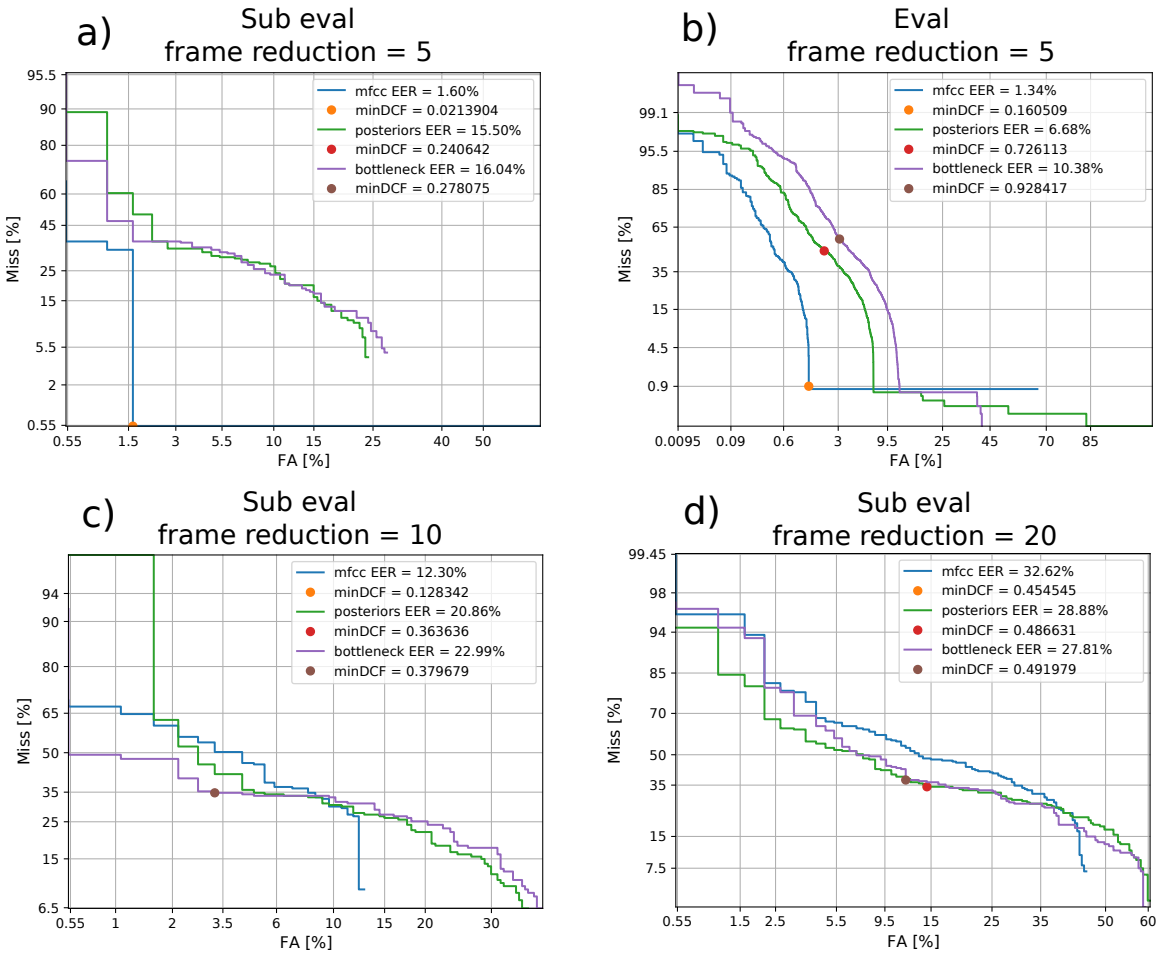


Figure 5.6: DET curves of RQA system with different settings. Both the a) (Sub eval dataset) and b) (Eval dataset) used frame reduction by five. In c) frame reduction by ten and d) frame reduction by twenty. MinDCF weights for *Sub eval* is 0.5, for *Eval* 0.08.

5.3 Fuzzy phone string matching approach

Fuzzy string matching (FSM), also known as approximate string matching is the technique of finding strings that do not match exactly, but only approximately. The approach is based on *Levenshtein distance* described in Section 2.4.

Classifying the recordings

First, the text file is imported and parsed accordingly:

- First element in the list represents the list of intervals, where each element in the list of intervals is the timestamp of the corresponding phoneme on that index. The interval corresponds to the original framed waveform (10 ms overlapping windows). For example, let assume a half second long phoneme "a" at the beginning of the speech recording. In this case, the first element in the list of intervals contains interval 0 – 50.

- Second element in the list depicts the list of phoneme durations (in hundredths of seconds) of the corresponding phoneme on the index.
- Third element represents the list of phonemes. Thus, each index in the parsed list represents one phoneme.
- The last element is the string to compare, where all "pau" phonemes are converted to spaces.

Such a parsing allows to convert between features and to return just needed parts of a string.

The comparison is processed by *FuzzyWuzzy* package, described in Section 4.2. The main advantage is the partial ratio function, which finds the best score in substrings. Apart from the raw ratio score, this ability to search substrings is more suitable for the given task but also makes it slower.

For example, let assume a pair of strings: i) "Thank you for calling, now leave a message." and ii) "Thank you for calling, please leave a message. Thank you for calling, please leave a message.". The ratio function returns score 60%, while partial ratio returns 84%. In such an ideal scenario, it works flawlessly as both the strings contain a message, and the results are reasonable.

The problem appears when a short string is compared to longer. For example, assume a pair of strings: i) "Thank you." and ii) "Thank you for calling, please leave a message.". The ratio function returns a score of 36%, while the partial ratio returns 90%. Even more extreme case is when the short string is just "you". In this scenario, the partial ratio returns a score of 100%. This can be fixed by applying brute force – ignore 100% scores, as it is almost certain that it is showed case. It is important to realise that even the same telephone operator messages will not probably return a score of 100%. Each message is repeated various times, changed in speed and volume, which causes the phoneme recogniser to act a bit different to each scenario and return a bit different best phoneme string.

The algorithm of this approach is based on comparing each string to another. Similar to the baseline DTW approach, it is in an $O(n^2)$ time-space. Accordingly, it is optimised by reducing the amount of data to compare to a single file, but in this case, it is a 1/4 bunch that is shuffled. The 1/4 chunks are provided only when the amount of data exceeds 2000 files. For example, let assume a set of 2000 files. With every evaluated file, a new chunk of 500 files is randomly chosen from the whole set.

Results

The system provides binary classification without the exact timing of the found message, and the system does not know the messages. The processing can be parallelised. The main drawback, as shown in DET curves 5.7, is the high false alarm rate – the main reason is the phenomenon mentioned in subsection 5.3. Several cases of false alarms occurred with mainly shorter recordings. The average time of processing is shown in the table 5.4.

Dataset	Average time [seconds]	Total time [hours]	EER [%]	MinDCF	The best threshold
Sub eval	29.57	3.07	**18.18%**	0.24	60.0
Eval	37.14	102.53	**33.14%**	0.97	65.0

Table 5.4: Processing time and evaluation results of baseline fuzzy string matching system. Tested on "Sub eval" and "Eval" sets. The best threshold corresponds to get the EER result. The evaluation on "Eval" dataset was parallelised to 6 CPU threads, thus the CPU-core computation time is around 17 hours. It is relevant to mention that the Eval dataset is unweighted – only 8% of the files contain the pre-recorded message, which results in a significant change in minDCF (minDCF parameter for Sub eval is 0.5, for Eval 0.08).

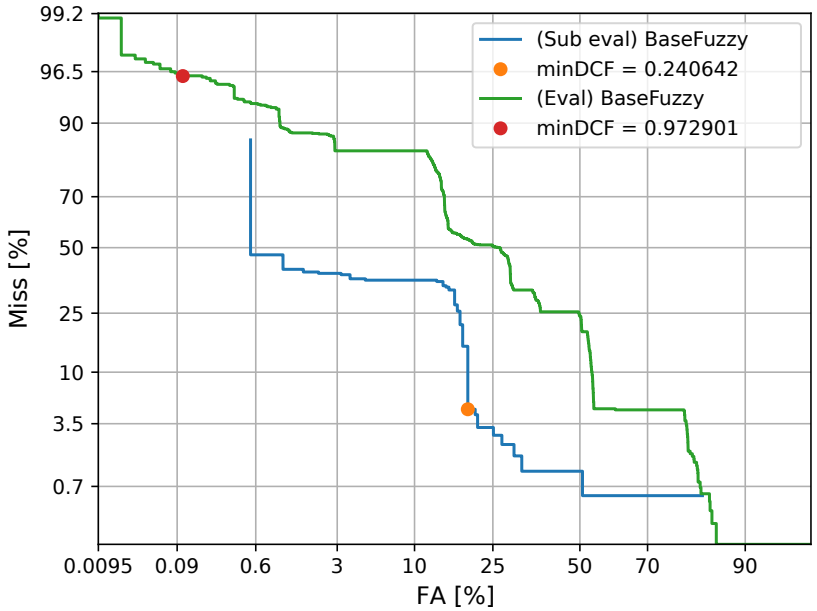


Figure 5.7: DET curves of baseline fuzzy phone string matching system. The accuracy drops almost twice on the Eval dataset.

Chapter 6

Optimisations

This chapter describes steps undertaken to optimise the implementation. The goal is to minimise files loading, performing redundant loops or use effective and necessary-only variables.

6.1 Caching

Caching is the process of storing copies of files in a temporary storage location so that they can be accessed more quickly and do not need to be loaded again. Each file is loaded to array and parsed in the beginning, and it is ready in cache for next use. A cache is a dictionary – the key is the file name, and the value is the list with all components needed for the present system for further processing. This simple improvement helps to reduce time by almost 80% in some cases, more in the table 6.1.

System (Sub eval)	Average time [seconds]	Average time cached [seconds]	Improvement [%]
SDTW clustering (FR=20, MFCC)	53.58	10.96	80%
SDTW clustering (FR=20, PPost)	131.09	38.02	71%
SDTW clustering (FR=20, BNF)	53.58	10.96	27%
RQA clusters + SDTW (MFCC)	21.07	9.39	55%
RQA clusters + SDTW (PPost)	46.71	15.25	67%
RQA clusters + SDTW (BNF)	43.21	22.67	52%
Base fuzzy string matching	39.00	29.57	43%

Table 6.1: Processing time with and without caching on various evaluation systems.

6.2 Frame averaging

Frame averaging, or reduction is a process of reducing the size of the feature vector resulting in faster computation.

Dimensions reduction

The dimension of the array depends on the current type of feature, MFCC features are a 13-dimensional array, phoneme posteriors a 138-dimensional array, and bottleneck features are an 80-dimensional array.

As each dimension preserves some relevant information, dimension reduction is not the right way to go – except for phoneme posteriors. The posteriors feature vector represents three states for each one of the 46 classes. To reduce computation time, posteriors of triplets of states are added to create posterior of one phoneme class. Thus, the array is reduced by three and this setting is used implicitly.

Reduction in a time axis

The array can be reduced in the time axis as well. Each second in an array is represented by a hundred of 10 ms frames. The idea is to get a mean value of n frames to minimise computation time while preserving as much information as possible.

The ratio differs from system to system, for example, the RQA analysis frame averaging by five is chosen to be the best, as table 5.3 shows. Frame reduction by five means that each second consists of 20 frames. For S-DTW clustering, the frame reduction is settled around $n = 20$ – five frames represents one second, for more details why the best value is $n = 20$ see table 8.1.

Chapter 7

Experiments with evaluating by a list of files

This chapter follows unknown-messages scenario outlined in Section 1.2. The experiments use evaluation by a random chunk of a set (experiment 7.2.1) or a list of candidates (experiments 7.1.1 and 7.1.2).

7.1 Experiments with DTW

This section describes experiments with DTW and its modification – S-DTW. The experiments are based on the second pass approach as outlined in Section 2.2.3 and it is similar to *relevance-feedback* mentioned in Section 2.3. The idea is to process files twice. The first pass RQA analysis detects candidates and the second pass compare all files again but just to the candidates. The goal is to improve the performance of the baseline systems – to bring higher accuracy and faster processing time. The advantage of this approach is that the RQA first-pass is processed only once, and the second-pass is independent of the used first-pass.

The experiments are divided into two subsections: i) DTW second-pass, ii) S-DTW second-pass.

7.1.1 DTW second pass

In this experiment, DTW is used as a technique for the second pass. The main difference to the baseline DTW system is that each file is not compared to all files but just to the candidates from the RQA analysis from the first pass. This method is described in Section 1.2.c).

In Aren Jansen's work [10] only S-DTW is used for the second pass. However, standard DTW is chosen for its faster computation time as it is searching for only one global optimal alignment. DTW is almost 90-times faster than S-DTW, as shown in table 7.1.

Preparation

To speed up the process, frame averaging is used. As shown in table 8.1, the best count of averaged frames is twenty. Twenty frames averaged into one frame represent 200 ms of information. Because of reducing the feature vector significantly, only recordings longer than 3 seconds are used in classification.

Features	MFCC		Phoneme posteriors		Bottleneck		Average
Samples lengths [seconds]	**30**	**30**	**30**	**30**	**30**	**30**	
DTW time [seconds]	0.008	0.180	0.011	0.204	0.014	0.313	
S-DTW time [seconds]	0.739	17.601	0.810	19.985	1.130	26.012	
Ratio of DTW and S-DTW speeds	96.15	97.90	73.03	97.80	79.98	83.105	**87.99**

Table 7.1: Comparison of Librosa DTW and S-DTW implementation by gray0302. The results are an average of a hundred runs for each comparison. The feature vectors are not frame averaged – original size is used. The conclusion is that SDTW is ≈ 88 times slower.

To get as relevant candidates as possible from the RQA first pass, the candidates are filtered to files longer than 4.5 seconds only. The threshold 4.5 seconds is chosen as this value eliminates most of the false alarms. It also helps to reduce the number of files to compare, and only the most relevant stay in the chunk.

Librosa DTW is used for computing as it is more suitable for shorter sequences than FastDTW as mentioned in Section 4.2. The settings for DTW are the same for Librosa DTW as for FastDTW – same distance metric, same multiplicative weights. The threshold varies depending on used features. For the MFCC features, the threshold is in general higher than for bottleneck features or phoneme posteriors.

Results

The goal was to reduce processing time and to achieve higher accuracy. Even though the processing time of a single file is reduced remarkably, it is still high enough to test the system on a larger evaluation dataset. The comparison to baseline DTW is presented in table 7.2, and the DET curve of the RQA + DTW second-pass system is shown in Figure 7.1. The system performs binary classification of files without the exact timestamp of a detected message and works entirely without known messages. The processing can be parallelised.

	Average time [seconds]			Total time [hours]		EER [%]			MinDCF		The best threshold
System	**Base DTW**	**DTW second pass**	**Change [%]**	**Base DTW**	**DTW second pass**	**Base DTW**	**DTW second pass**	**Change [%]**	**Base DTW**	**DTW second pass**	**DTW second pass**
MFCC features	**244.39**	**24.90**	89.81%	**25.38**	**2.59**	41.17%	42.78%	-3.76%	0.61	0.80	78.59
Phoneme posteriors	284.30	47.58	83.26%	29.37	4.94	30.48%	**22.99%**	**24.57%**	**0.42**	**0.33**	9.78
Bottleneck features	477.49	37.28	**92.19%**	49.59	3.87	**29.41%**	32.62%	-9.84%	0.50	0.65	2.97

Table 7.2: Comparison of baseline DTW system to RQA+DTW second-pass system. The second pass approach is 90% faster – the duration of the first pass of RQA analysis is excluded. Accuracy is about the same as baseline.

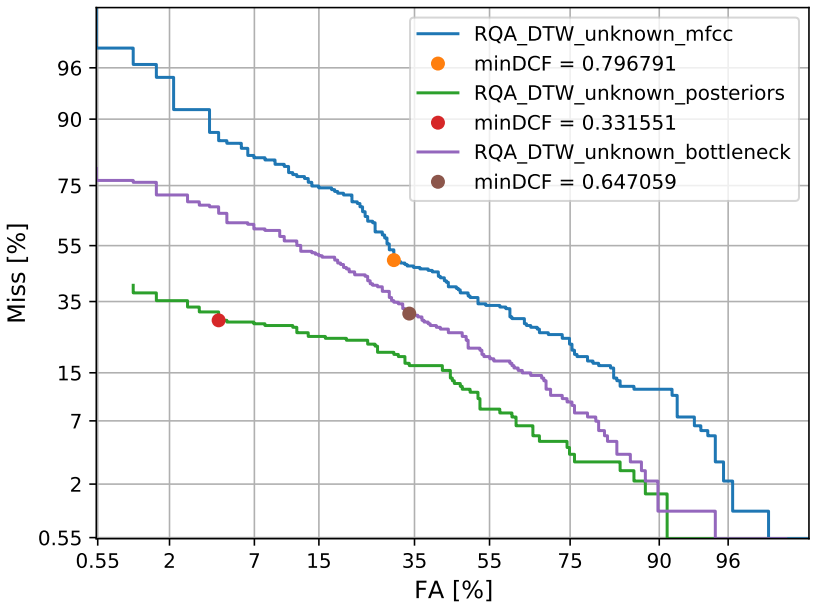


Figure 7.1: DET curve of RQA + DTW system. The accuracy is almost the same as the baseline DTW system. Evaluated on Sub eval dataset.

7.1.2 S-DTW second pass

Segmental DTW (S-DTW) is a modification of the DTW algorithm, which search for an optimal path everywhere, not on the main diagonal, more in subsection 2.1.3. This method is described in Section 1.2.c).

The approach should bring higher accuracy and reduced calculation time. Thanks to SDTW, the system specifies the exact position of the detected message in the audio.

Preparation

Preparation is the same as in DTW second pass approach. The same frame reduction of size 20 is used, and also we retain only files longer than 3 seconds for classification. The same filter selects recordings from RQA first-pass analysis – longer than 4.5 seconds only.

For computing, S-DTW by *gray0302* is used. The width R to restrict search space is set to five, as designed by the author. The length L is set to 300 hundred frames as the average duration of one repetition of operator message is 3 seconds. To be more precise, the constraint L is divided by the size frame reduction to preserve the length of 3 seconds. The cosine distance metric is used.

Results

A faster processing time is achieved compared to the baseline system, although, it is slower than DTW second-pass approach. Unexpectedly, the accuracy is not higher compared to both baseline and DTW second-pass systems. Thus, it is still not fast enough to test the system on a larger evaluation dataset. The comparison to baseline DTW is presented in

table 7.3, and the DET curve of the RQA + SDTW second-pass system is shown in Figure 7.2.

System	Average time [seconds]			Total time [hours]		EER [%]			MinDCF		The best threshold
	Base DTW	SDTW second pass	Change [%]	Base DTW	SDTW second pass	Base DTW	SDTW second pass	Change [%]	Base DTW	SDTW second pass	SDTW second pass
MFCC features	**244.39**	90.81	62.84%	**25.38**	9.30	41.17%	37.39%	9.18%	0.61	0.61	0.0039
Phoneme posteriors	284.30	**71.41**	**74.88%**	29.37	**7.32**	30.48%	35.77%	-14.79%	**0.42**	**0.47**	0.0449
Bottleneck features	477.49	193.82	59.41%	49.59	19.87	**29.41%**	**31.98%**	**9.84%**	0.50	0.56	0.4832

Table 7.3: The table presents a comparison of the baseline DTW system to the RQA+SDTW second-pass system. The second pass approach is 65% faster – the duration of the first pass of RQA analysis is excluded. Accuracy is about the same as baseline.

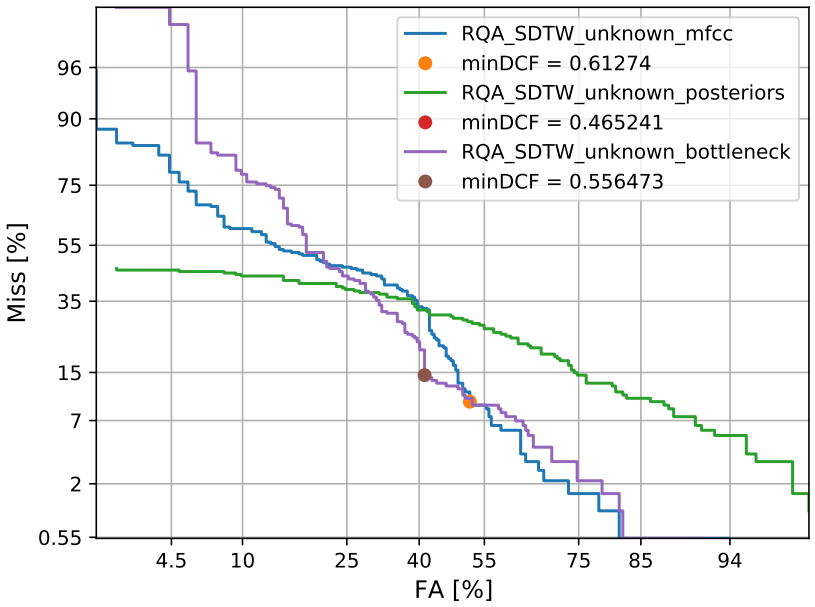


Figure 7.2: DET curve of RQA+SDTW system. The accuracy is almost the same as the baseline DTW system. The system was evaluated on Sub eval dataset. Added value compared to the base DTW system and RQA+DTW system is that RQA-SDTW indexes the position of the detected message.

The system performs binary classification with the exact timestamp of a detected message and works entirely without known messages. However, the timestamps are not indexing the entire position of the message but rather bring an idea of where the message is. Such issue happens as SDTW searches for paths longer than parameter L, however, SDTW does not maximise the search length entirely – the searching ends as L is satisfied with minimum final distance and does not search for longer sequences. Thus, to get an entire timestamp

of a message, further analysis is necessary (e.g. speaker diarisation). The processing can be parallelised.

7.2 Experiments with fuzzy phone string matching

The baseline fuzzy phone string matching system showed viability, though the system comes with drawbacks. The main goal of the following experiment is to minimise the errors and to try to increase the speed of the classification.

7.2.1 Pause analysis

The pause-analysis modification of the baseline fuzzy string matching system aims to solve the main issue with the baseline system by segmenting the recordings by longer pauses.

Preparation

After the phoneme strings are loaded, the files are processed by pause analysis. The idea is to exclude all parts which are not relevant for further classification.

All silent parts longer than 2 seconds are declared as dividing points. Two seconds is an optimal trade-off between segmenting every pause and being still sensitive to silent parts. The 2-second-long pause is longer than the typical pause between the words and the usual pause between the repetitions of the pre-recorded messages.

Then the length of the segmented lists is checked. If the segment is shorter than 50 elements of the list – it contains less than 50 phonemes, the segmented part is removed from the candidates. The reason is that the relevant length of the message is more than 50 phonemes – if fewer phonemes are present in the segment, it does not carry the pre-recorded message.

Three possible options of the analysis may occur: `i)` nothing happened – no significant silence detected, `ii)` occurrence of segmented parts but none removed, and `iii)` the segmented parts are present, but some may be removed.

The pause analysis process decreases the cause of errors.

Results

The simple pause analysis helps to improve accuracy and achieve faster processing. The results are better in the small *Sub eval* dataset than in the larger *Eval* dataset. The improvement is only in faster processing. The reason is probably due to the disproportion of the target and non-target files. The comparison to the baseline fuzzy system is in table 7.4 and figure 7.3.

The system provides information about the position of the detected message. However, it is only indicative and inaccurate information, as it depends solely on the pause analysis segments.

The process can be parallelised, and it uses an unknown messages approach.

(Sub eval dataset)	Average time [seconds]	Total time [hours]	EER [%]	MinDCF	The best threshold
baseline FSM	29.57	3.07	18.18%	0.24	60.0
pause analysis FSM	**19.10**	**1.98**	**12.83%**	**0.15**	68.0
(Eval dataset)	**Average time [seconds]**	**Total time [hours]**	**EER [%]**	**MinDCF**	**The best threshold**
baseline FSM	37.14	118.03	33.14%	**0.97**	65.0
pause analysis FSM	**24.44**	**77.54**	**31.56%**	0.98	72.0

Table 7.4: Performance of the baseline fuzzy phone string matching system and the pause analysis modification.

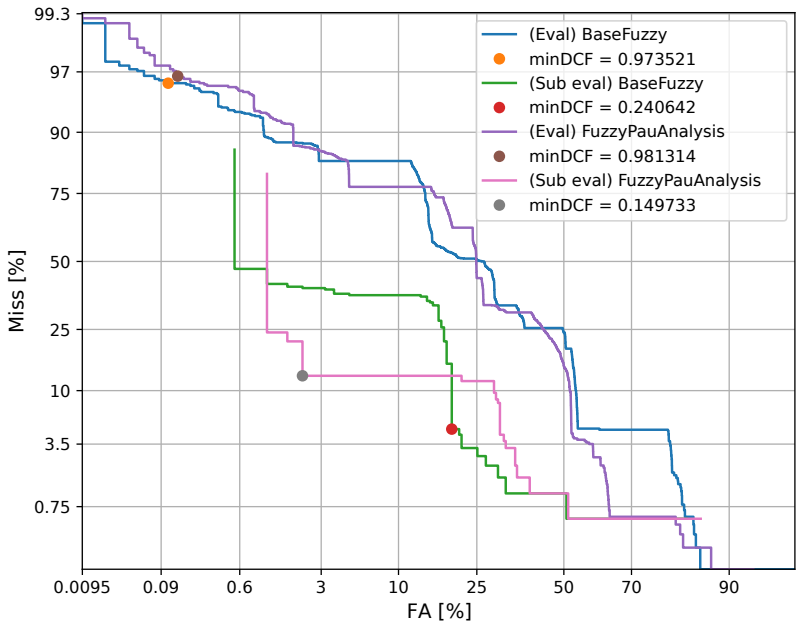


Figure 7.3: DET curves of the base fuzzy string matching system and the pause analysis modification. The accuracy is about the same for both systems, the pause analysis systems provide faster processing time.

Chapter 8

Experiments with clustering

In this chapter, we present the results of experiments performed with evaluation by clustering. Two types of clusters are used in evaluation: i) reference cluster – created by labelled pre-recorded messages (known-messages scenario, described in Section 1.1) and ii) predicted cluster – created by list of candidates from RQA analysis (unknown-messages scenario, described in Section 1.2.d))

8.1 RQA S-DTW clustering

To reduce the processing time of one file significantly more, it is necessary not to compare a file to every candidate. Clustering helps to divide candidates into groups – classes. Clustering adds another layer to the classification process, mainly aimed to decrease time complexity and to increase accuracy.

The clustering process consists of two steps: i) VAD detection and filtering and ii) dividing candidates into classes by using SDTW.

VAD detection

First, every relevant element from the RQA analysis is sent to VAD analysis. Relevant elements are those, having a non-empty list of frames from RQA analysis. These frames are sent to VAD, and the active parts are kept.

SDTW clustering

Second, filtered files from VAD are compared to each other by SDTW. To be more sensitive to any similarity, the length constraint L is set to register at least $200/frame\,reduction$-long frames. The comparison works similarly to any other mentioned systems. However, the list is gradually decreased as the files are divided into classes with every loop.

It follows that the first file is going through all of the filtered chunks, n files of the bunch are grouped by the smallest distance, then the next loop cycle, the chunk is smaller by the n files and so on. Thence, in every cycle, a new cluster class is created. Every cluster is sorted by the lengths of the files – the shortest recordings are at the top.

After the comparison, some files were not grouped anywhere. These "*singles*" are not automatically false alarms – they may not have "luck" to be compared to the most similar recordings to them. Thence, a second run is performed.

In the second run, the "singles" are compared to the first three recordings from each created cluster from the first run. If there are any "singles" left after the second run, then they are excluded.

The clusters are stored persistently by *Pickle* utility for further use. The processing cannot be parallelised. The advantage of the approach is that every file can be added to one of the cluster classes at any time. Formal description of clustering process is presented in algorithm 8.1.

Algorithm 8.1: Formal description of clustering process.

```
Input: list_of_candidates
Result: clusters
class_index ← 0
clusters ← list.empty()
filtered_candidates ← VAD(list_of_candidates)
for file in filtered_candidates do
    feature_vector ← loadPartOfInterests(file) if not in cache
    if feature_vector in clusters then
        continue // already in clusters - it is not processed again
    cache.add(feature_vector)
    clusters ← list.add(new_cluster) // new cluster in each run
    clusters[class_index].append(feature_vector)
    for nested_file in filtered_candidates do
        nested_feature_vector ← loadPartOfInterests(nested_file) if not in cache
        cache.add(nested_feature_vector)
        if nested_feature_vector in clusters then
            continue
        SDTW_distance ← SDTW(L, R, nested_feature_vector, feature_vector)
        if SDTW_distance < threshold then
            clusters[class_index].append(nested_feature_vector)
    clusters[cluster_index].sortByDuration()
    class_index++
singles_list ← findClusterOfSizeOne(clusters)
for file in singles_list do
    feature_vector ← cache.get(file)
    class_index ← 0
    for cluster in clusters do
        for the first 3 files in cluster do
            SDTW_distance ← SDTW(L, R, nested_feature_vector,
             feature_vector)
            if SDTW_distance < threshold then
                clusters[class_index].append(feature_vector)
        class_index++
removeClusterOfSizeOne(clusters)
```

Clustering evaluation

Clustering performance evaluation is necessary to ascertain the best settings and to select the best cluster among others. To evaluate created clusters, a reference clustering is created first.

The creation of the reference clusters is based on known messages and their labels. The filename provides information about the message ID, the start and end of the message. Every message ID represents one class – one cluster (of the total of 25 clusters). The clusters are sorted by the lengths of the files – the same way as in the SDTW clustering.

Clustering performance evaluation metrics are presented in Section 2.6. In table 8.1, clusters with different frame reduction size and various features are used. As seen in the table, the best speed-accuracy trade-off is achieved by using frame reduction of size 20.

Some errors in clustering are not relevant – several messages are the same in content, but the speakers differ in gender. Nevertheless, in reference clusters, this case has separate clusters for the messages. Thence, it is marked as an error in predicted clustering. However, some existing errors are not penalised. Predicted clustering by a list of candidates have a chance of getting false alarms in a cluster. This behaviour is not marked as an error, as the reference clusters do not contain any false alarms.

Metric	**Purity**			**Rand Index**			**NMI**			**Total Time [minutes]**		
Features	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**
Sub eval FR = 10	**0.877**	0.865	0.816	**0.986**	0.983	0.976	**0.932**	0.918	0.922	**18.30**	141.04	288.90
Sub eval FR = 20	0.657	0.676	0.805	0.931	0.960	0.978	0.817	0.801	0.916	4.93	28.51	58.20
Sub eval FR = 30	0.431	0.425	0.796	0.897	0.855	0.977	0.682	0.561	0.917	2.20	8.67	17.93
Sub eval FR = 40	0.309	0.328	0.769	0.829	0.748	0.970	0.542	0.448	0.897	1.76	6.52	11.15
Eval FR = 20	0.554	0.659	**0.772**	0.875	0.954	**0.969**	0.743	0.751	**0.870**	**183.57**	237.97	217.63

Table 8.1: The table shows clustering performance evaluation by several metrics as Purity, Rand Index and NMI. The evaluation runs on different features like MFCC features (M), phoneme posteriors (PP), and bottleneck features (BN) with various settings of frame reduction (FR). The best speed-accuracy trade-off is with the $FR = 20$.

8.2 Clustered S-DTW second pass

This experiment uses both reference clusters and clusters created by a list of candidates from RQA analysis (described in Section 8.1).

Preparation

The algorithm is the same as other DTW, SDTW methods. The only difference is that each file is not compared to every relevant candidate from first-pass, but only to two files from every cluster class. All parameters are the same – frame reduction = 20, distance metric = cosine, parameter R = 5, and L = 300. While processing, the data points classified as hits, can be added to the existing cluster. However, the option is turned off by default.

Results

The results depend on the quality of given cluster. The system works in two modes: i) usage of the reference clusters, thence the messages are known and ii) usage of the created clusters by clustering algorithm, thus the messages are unknown.

The system provides information about the position of the detected message, though the same drawback is still present – as mentioned in SDTW second pass before. The system is fast enough to be evaluated on bigger datasets, what makes it the main advantage of the used DTW/SDTW systems. The processing can be parallelised.

The results of the systems are presented in table 8.2 and in Figure 8.1. The comparison between the baseline DTW and the clustered version of SDTW is presented in table 8.3.

Reference clusters	**Average time [seconds]**			**Total time [hours]**			**EER [%]**			**MinDCF**		
Features	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**
Sub eval [known]	**8.25**	10.89	21.88	**0.85**	1.12	2.24	11.11%	19.78%	**4.88%**	0.21	0.39	**0.09**
Eval [known]	17.35	**12.06**	19.78	9.12	**6.30**	10.15	15.59%	28.84%	**3.05%**	0.39	0.90	**0.08**
SDTW clusters	**Average time [seconds]**			**Total time [hours]**			**EER[%]**			**MinDCF**		
Features	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**
Sub eval [unknown]	**9.39**	15.25	22.67	**0.96**	1.56	2.32	**12.74%**	19.78%	13.82%	0.25	0.34	**0.24**
Eval [unknown]	7.93	**5.03**	17.74	4.12	**2.64**	9.19	39.92%	42.13%	26.59%	0.99	1.00	0.80
Eval (Sub eval BNF cluster) [unknown]	-	-	22.07	-	-	11.33	-	-	**13.80%**	-	-	**0.41**

Table 8.2: Performance of the clustered SDTW second pass system. The used features represent both used for clustering and classifying. All runs on the Eval dataset were parallelised to 6 threads. The best results on the Eval dataset is achieved by Sub eval's bottleneck feature cluster. The cluster is chosen because of the best NMI score and the small count of false alarms.

	Average time [seconds]			**Total time [hours]**		**EER [%]**			**MinDCF**		**The best threshold**
System	**Base DTW**	**clustered SDTW second pass** Sub eval (unknown FR= 20)	**Change [%]**	**Base DTW**	**clustered SDTW second pass** Sub eval (unknown FR= 20)	**Base DTW**	**clustered SDTW second pass** Sub eval (unknown FR= 20)	**Change [%]**	**Base DTW**	**clustered SDTW second pass** Sub eval (unknown FR= 20)	**clustered SDTW second pass** Sub eval (unknown FR= 20)
MFCC features	**244.39**	**9.39**	**96.16%**	**25.38**	**0.96**	41.17%	**12.74%**	**69.06%**	0.61	0.25	0.0029
Phoneme posteriors	284.30	15.25	94.64%	29.37	1.56	30.48%	19.78%	35.10%	**0.42**	0.34	0.2845
Bottleneck features	477.49	22.67	95.25%	49.59	2.32	**29.41%**	13.82%	53.01%	0.50	**0.24**	0.2095

Table 8.3: Comparison of baseline DTW system to RQA + SDTW clustering + SDTW second-pass system. The second pass approach is about 95% faster – the duration of the RQA analysis and clustering is excluded. Evaluated on "Sub eval" dataset.

Notice that in the case of using the reference clusters, DET curves are showing the performance of the used features because of no bias of better or worse starting point depending

on the used cluster. Thence, bottleneck features is the best choice for the system, which the previous DTW/SDTW systems' results confirm as well. Many errors are caused by the infiltrated non-message recordings into the clusters, as they then classify other non-message recordings as hits.

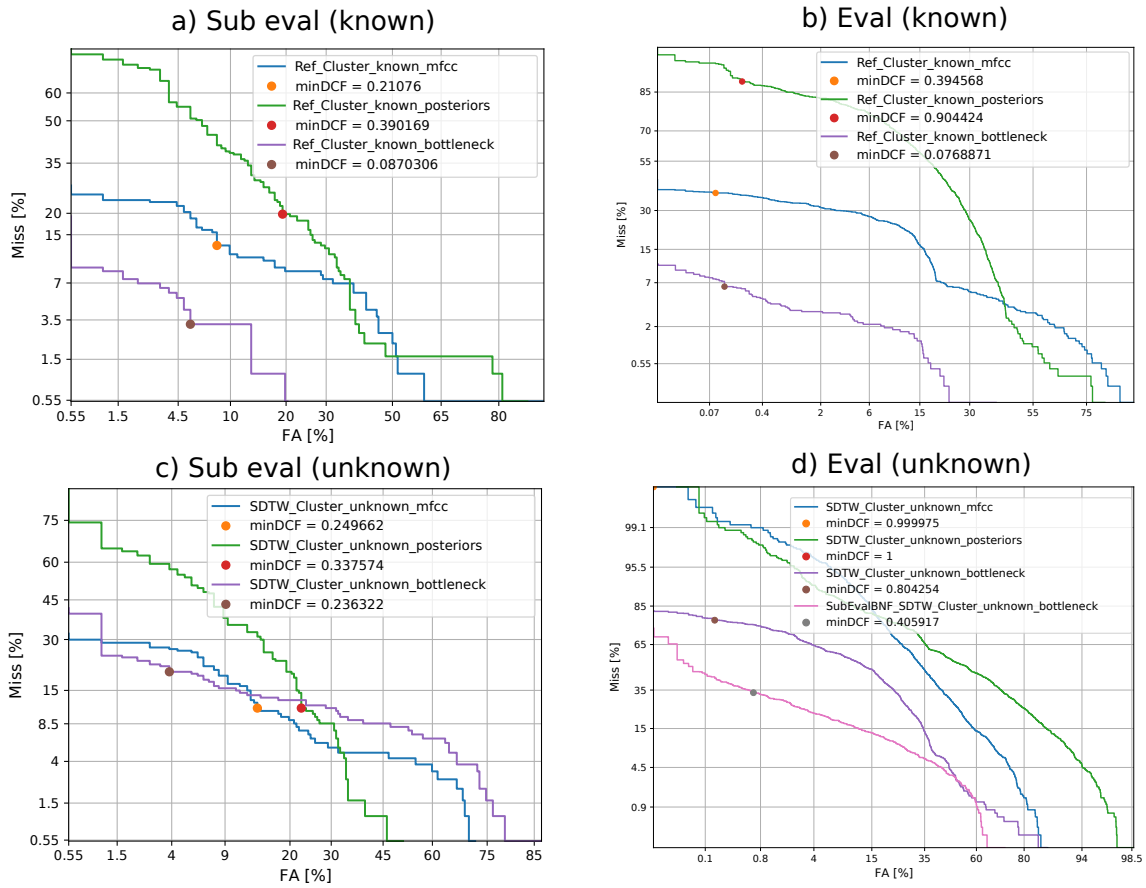


Figure 8.1: DET curves of the clustered SDTW second pass system.

8.3 Clusters with fuzzy phone string matching

The system combines all created systems to get the best performance. This experiment uses both reference clusters and clusters created by a list of candidates from RQA analysis (described in Section 8.1).

Preparation. This approach uses clusters explained in subsection 8.1, where frame reduction $FR = 20$ and clusters of all features are used, and reference clusters are used too. For the phoneme strings, pause analysis is applied. While classifying, each file is compared to first three elements from each cluster.

Results. From all presented systems, RQA first pass + SDTW clustering + fuzzy phone string matching obtained the best overall results and the fastest processing time. The system works both with known and unknown messages clusters and the system provides

info about the detected message – only indicative and inaccurate timestamp information. The comparison between baseline fuzzy phone string matching (FSM) system and SDTW cluster FSM is shown in table 8.5. The results are presented in table 8.4 and figure 8.2.

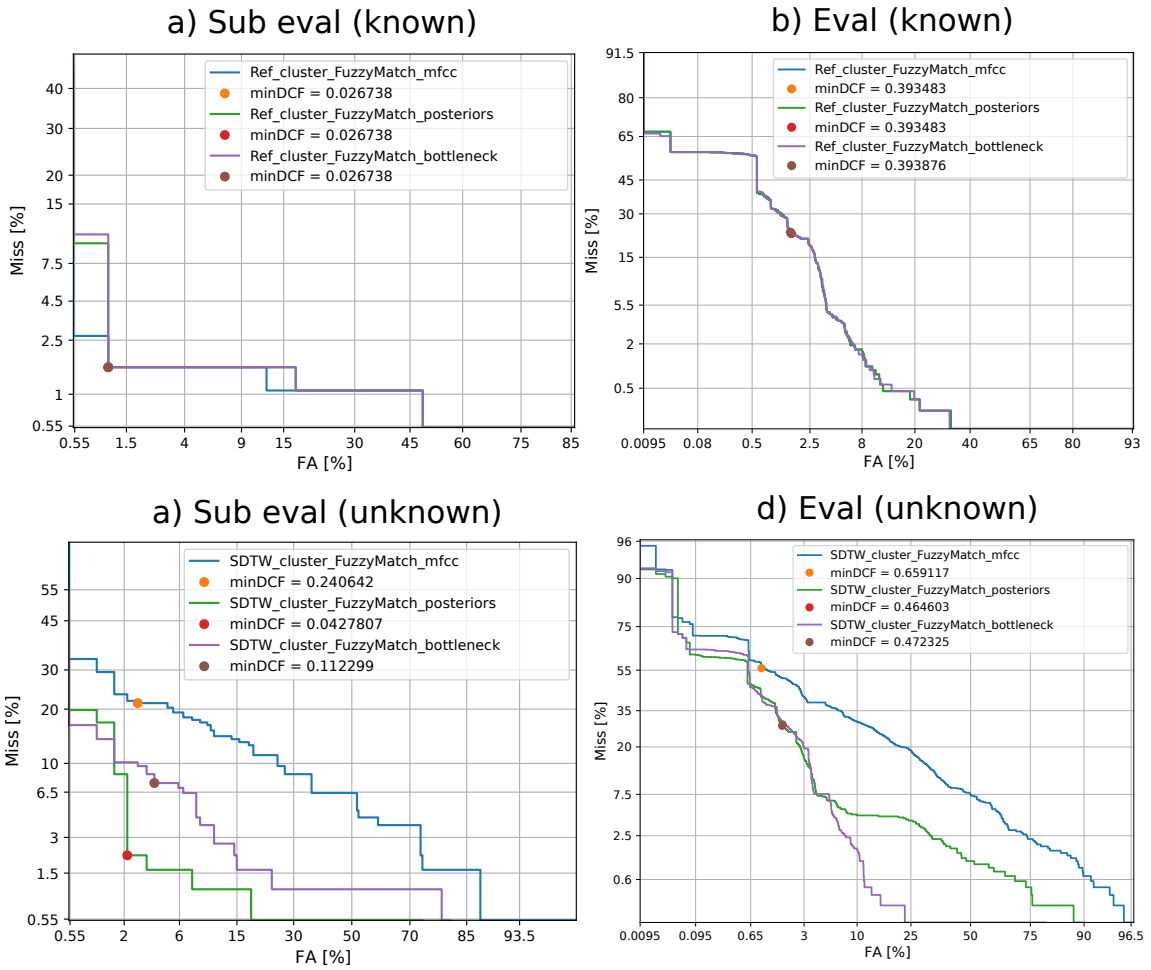


Figure 8.2: DET curves of the SDTW cluster + fuzzy string matching system. Used SDTW clusters are created from the Eval dataset from all available features. The system works with both known messages a), b) and unknown messages c), d).

Ref cluster	**Average time [seconds]**			**Total time [hours]**			**EER [%]**			**MinDCF**		
Features	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**
Sub eval	6.68	5.98	**5.96**	0.69	0.62	**0.62**	1.60%	1.60%	**1.60%**	0.03	0.03	**0.03**
Eval	17.35	**12.06**	19.78	9.12	**6.30**	10.15	4.28%	**4.28%**	4.30%	0.39	**0.39**	0.39
SDTW cluster	**Average time [seconds]**			**Total time [hours]**			**EER [%]**			**MinDCF**		
Features	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**	**M**	**PP**	**BN**
Sub eval	**0.79**	3.91	3.73	**0.08**	0.41	0.39	13.90%	**2.14%**	6.41%	0.24	**0.04**	0.11
Eval	1.57	1.57	**1.43**	4.99	6.39	**4.54**	20.48%	6.34%	**5.77%**	0.66	**0.46**	0.47

Table 8.4: Results of SDTW clustered FSM system. Evaluated on both the Sub eval and Eval datasets. The system works with both known messages (reference cluster) and unknown messages (SDTW cluster).

(Sub eval dataset)	Average time [seconds]			Total time [hours]		EER [%]			MinDCF		The best threshold
System	***Base FSM**	**clustered FSM**	**Change [%]**	***Base FSM**	**clustered FSM**	***Base FSM**	**clustered FSM**	**Change [%]**	***Base FSM**	**clustered FSM**	**clustered FSM**
MFCC features	29.57	**0.79**	**97.32%**	3.07	**0.08**	18.18%	13.90%	23.54%	0.24	0.24	51.0
Phoneme posteriors	29.57	3.91	86.78%	3.07	0.41	18.18%	**2.14%**	**88.23%**	0.24	**0.04**	51.0
Bottleneck features	29.57	3.73	87.39%	3.07	0.39	18.18%	6.41%	64.74%	0.24	0.11	51.0
(Eval dataset)	**Average time [seconds]**			**Total time [hours]**		**EER [%]**			**MinDCF**		**The best threshold**
System	***Base FSM**	**clustered FSM**	**Change [%]**	***Base FSM**	**clustered FSM**	***Base FSM**	**clustered FSM**	**Change [%]**	***Base FSM**	**clustered FSM**	**clustered FSM**
MFCC features	37.14	1.57	95.77%	118.03	4.99	33.14%	20.48%	38.20%	0.97	0.66	45.0
Phoneme posteriors	37.14	2.01	95.59%	118.03	6.39	33.14%	6.34%	80.87%	0.97	**0.46**	51.0
Bottleneck features	37.14	**1.43**	**96.15%**	118.03	**4.54**	33.14%	**5.77%**	**82.59%**	0.97	0.47	48.0

Table 8.5: Comparison between baseline fuzzy phone string matching and SDTW clustered FSM. Evaluated on both the Sub eval and Eval datasets. *Base FSM does use only phoneme strings for the classification process none of the MFCC, posteriors, bottleneck features are used – that is the reason behind the same values in the corresponding columns.

8.4 Overall results

The performance varies depending on many aspects. The most evident parameters are the used features, where bottleneck features provide overall the best performance, followed by phoneme posteriors. Even though the MFCC features are not the best in accuracy, they provide the fastest processing time, and they are completely language independent.

The accuracy increased with every new experiment compared to baselines, while the processing time gradually decreased with every new approach.

However, there is still space to increase accuracy and effectiveness. Many shorter files cause errors, especially those containing shorter phrases resulting in good scores. I glimpsed several cases when an error was caused by imperfection of the dataset – e.g. in case of strong cross-talk between speaker A and speaker B channels, thence, when comparing these two sequences, the parts were correctly marked as similar. Such a difference in accuracy between "Sub eval" and "Eval" datasets may be caused by the percentage of recordings with messages in all files, which, in the Sub eval dataset, is 50% (half of the files is the goal data points), and in the Eval dataset, the data points with messages are around 8% of all files.

The best approach in any means is to use known-messages (for reference clustering). If such an option is not possible, then perform the RQA analysis first, followed by the SDTW bottleneck feature clustering, and finally use the fuzzy string matching approach.

Chapter 9

Conclusion

9.1 Summary of the work performed

In this thesis, we have presented the methods for detecting re-occurring sequences across audio data. More precisely, we focused on searching for pre-recorded telephone operator messages in speech conversations.

We took a deeper look at three techniques: Dynamic time warping, recurrent quantification analysis and fuzzy string matching. The main idea was to choose the most accurate and fastest approach. Particularly, we aimed to find a suitable combination of the techniques while the system runs without the knowledge of the messages.

To determine which system is the best, it was essential to simulate a proper dataset. We accomplished that by mixing the operator messages into the Switchboard corpus. The methods were evaluated on the derivatives of this simulated dataset. The results show the necessity of a combination of various procedures. We found out that RQA provides the best performance among the three baseline methods. We showed that the bottleneck features bring the highest accuracy and MFCC features the fastest processing time. To speed up the detection, caching and frame averaging significantly reduced the computation time.

In the end, we observed that the best solution is to use all techniques combined. First, we provide RQA analysis to get a list of candidates. Second, we perform SDTW clustering from the RQA list, and finally, we classify the speech conversation recordings by fuzzy string matching. The results are twice as good when using reference clusters.

9.2 Future directions

We did not cover some other aspects, which could increase accuracy and reliability. In general, the enhanced performance of the systems can be achieved by tuning the parameters in VAD or tweaking SDTW.

RQA second pass

The obtained results could be improved by performing RQA second pass instead of SDTW or FSM classification. The performance of RQA is proven to be fast and accurate. The possibility to compare all representatives of cluster classes to a file at once may improve the speed significantly while preserving high accuracy.

Speaker diarisation techniques

The problem of providing reliable timestamps of detected messages can be solved by speaker diarisation (SD) systems. SD systems mark the position of the message accurately, as the whole message is recorded by the same person. Another usage of SD systems could be to save the voiceprint of a speaker of a detected message. The saved voiceprint may help in the detection process.

Bibliography

[1] B, J. and Babu.N, R. Speech recognition using MFCC and DTW. In: *2014 International Conference on Advances in Electrical Engineering (ICAEE)*. January 2014, p. 4. DOI: 10.1109/ICAEE.2014.6838564.

[2] Caswell, T. A., Droettboom, M., Lee, A., Hunter, J., Andrade, E. S. de et al. *Matplotlib/matplotlib: REL: v3.3.2*. Zenodo, september 2020. Available at: https://doi.org/10.5281/zenodo.4030140.

[3] Chen, Y., Chen, K., Wang, H. and Chen, B. Effective pseudo-relevance feedback for spoken document retrieval. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, p. 8535–8539. DOI: 10.1109/ICASSP.2013.6639331.

[4] Fan, J., Upadhye, S. and Worster, A. Understanding receiver operating characteristic (ROC) curves. *Canadian Journal of Emergency Medicine*. Cambridge University Press. 2006, vol. 8, no. 1, p. 19–20. DOI: 10.1017/S1481803500013336.

[5] Fapšo, M. *Query-by-Example Spoken Term Detection*. Brno, CZ, 2014. Ph.D. thesis. Brno University of Technology, Faculty of Information Technology. Available at: https://www.fit.vut.cz/study/phd-thesis/282/.

[6] Fér, R., Matějka, P., Grézl, F., Plchot, O., Veselý, K. et al. Multilingually Trained Bottleneck Features in Spoken Language Recognition. *Computer Speech & Language*. november 2017, vol. 46, p. 252–267. DOI: 10.1016/j.csl.2017.06.008.

[7] Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P. et al. Array programming with NumPy. *Nature*. Springer Science and Business Media LLC. september 2020, vol. 585, no. 7825, p. 357–362. DOI: 10.1038/s41586-020-2649-2. Available at: https://doi.org/10.1038/s41586-020-2649-2.

[8] Hazen, T. J., Shen, W. and White, C. Query-by-example spoken term detection using phonetic posteriorgram templates. In: *2009 IEEE Workshop on Automatic Speech Recognition Understanding*. 2009, p. 421–426. DOI: 10.1109/ASRU.2009.5372889.

[9] Ho, T., Oh, S.-R. and Kim, H. A parallel approximate string matching under Levenshtein distance on graphics processing units using warp-shuffle operations. *PLOS ONE*. Public Library of Science. october 2017, vol. 12, p. 1–15. DOI: 10.1371/journal.pone.0186251. Available at: https://doi.org/10.1371/journal.pone.0186251.

[10] Jansen, A., Church, K. and Hermansky, H. Towards spoken term discovery at scale with zero resources. In:. January 2010, p. 1676–1679.

[11] Kashyap, R. and Oommen, B. An effective algorithm for string correction using generalized edit distances—I. Description of the algorithm and its optimality. *Information Sciences*. 1981, vol. 23, no. 2, p. 123–142. DOI: https://doi.org/10.1016/0020-0255(81)90052-9. ISSN 0020-0255. Available at: https://www.sciencedirect.com/science/article/pii/0020025581900529.

[12] Likens, A. D., McCarthy, K. S., Allen, L. K. and McNamara, D. S. Recurrence Quantification Analysis as a Method for Studying Text Comprehension Dynamics. In: *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*. New York, NY, USA: Association for Computing Machinery, 2018, p. 111–120. LAK '18. DOI: 10.1145/3170358.3170407. ISBN 9781450364003. Available at: https://doi.org/10.1145/3170358.3170407.

[13] Manning, C. D. *Introduction to Information Retrieval*. Cambridge University Press, jul 2008. 356-359 p. ISBN 0521865719. Available at: https://www.xarg.org/ref/a/0521865719/.

[14] Martin, A., Doddington, G., Kamm, T., Ordowski, M. and Przybocki, M. A. The DET curve in assessment of detection task performance. In: *EUROSPEECH*. 1997.

[15] McFee, B., Lostanlen, V., Metsai, A., McVicar, M., Balke, S. et al. *Librosa/librosa: 0.8.0*. Zenodo, july 2020. Available at: https://doi.org/10.5281/zenodo.3955228.

[16] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J. and Khudanpur, S. Recurrent neural network based language model. In: Kobayashi, T., Hirose, K. and Nakamura, S., ed. *INTERSPEECH*. ISCA, 2010, p. 1045–1048. Available at: http://dblp.uni-trier.de/db/conf/interspeech/interspeech2010.html#MikolovKBCK10.

[17] Muda, L., Begam, M. and Elamvazuthi, I. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *J Comput*. march 2010, vol. 2, p. 138–143.

[18] Nosratighods, M., Ambikairajah, E., Epps, J. and Carey, M. Score weighting in speaker verification systems. In: *2007 6th International Conference on Information, Communications Signal Processing*. 2007, p. 1–4. DOI: 10.1109/ICICS.2007.4449714.

[19] Parekh, R. and Das, B. 2012 : Recognition of Isolated Words using Features based on LPC, MFCC, ZCR and STE, with Neural Network Classifiers. *International Journal of Modern Engineering Research*. january 2012, vol. 2, p. 854–858.

[20] Park, A. and Glass, J. Unsupervised Pattern Discovery in Speech. *Audio, Speech, and Language Processing, IEEE Transactions on*. february 2008, vol. 16, p. 186 – 197. DOI: 10.1109/TASL.2007.909282.

[21] Park, A. S. *Unsupervised pattern discovery in speech: Applications to word acquisition and speaker segmentation*. 2006. 46-53 p. Dissertation. Massachusetts Institute of Technology.

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, vol. 12, p. 2825–2830.

[23] Pfitzner, D., Leibbrandt, R. and Powers, D. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowl. Inf. Syst.* june 2009, vol. 19, p. 361–394. DOI: 10.1007/s10115-008-0150-6.

[24] Picheny, M., Nahamoo, D., Goel, V., Kingsbury, B., Ramabhadran, B. et al. Trends and advances in speech recognition. *IBM Journal of Research and Development*. 2011, vol. 55, no. 5, p. 2:1–2:18. DOI: 10.1147/JRD.2011.2163277.

[25] Portilla, R., Heintz, B. and Lee, D. *Understanding Dynamic Time Warping - The Databricks Blog* [https://databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html]. April 2019. (Accessed on 01/26/2021).

[26] Rao, K. and e, M. k. *Speech Recognition Using Articulatory and Excitation Source Features*. January 2017. ISBN 978-3-319-49219-3.

[27] Saenz Lechon, N., llorente, J. godino, Osma Ruiz, V. and Gomez, P. Methodological issues in the development of automatic systems for voice pathology detection. *Biomedical Signal Processing and Control*. april 2006, vol. 1, p. 120–128. DOI: 10.1016/j.bspc.2006.06.003.

[28] Salvador, S. and Chan, P. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intelligent Data Analysis*. january 2004, vol. 11, p. 70–80.

[29] Schwarz, P. *Phoneme recognition based on long temporal context*. Brno, CZ, 2009. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/phd-thesis/109/.

[30] Serrà, J., Serra, X. and Andrzejak, R. Cross recurrence quantification for cover song identification. *New Journal of Physics*. september 2009, vol. 11, p. 093017. DOI: 10.1088/1367-2630/11/9/093017.

[31] Tejedor, J., Fapšo, M., Szöke, I., Černocký, J. u. and Grézl, F. Comparison of Methods for Language-Dependent and Language-Independent Query-by-Example Spoken Term Detection. New York, NY, USA: Association for Computing Machinery. 2012, vol. 30, no. 3. DOI: 10.1145/2328967.2328971. ISSN 1046-8188. Available at: https://doi.org/10.1145/2328967.2328971.

[32] Van Leeuwen, D. and Brummer, N. An Introduction to Application-Independent Evaluation of Speaker Recognition Systems. In:. January 2007, vol. 4343, p. 330–353. DOI: 10.1007/978-3-540-74200-5_19. ISBN 978-3-540-74186-2.

[33] Velivelli, A., ChengXiang Zhai and Huang, T. S. Audio segment retrieval using a short duration example query. 2004, vol. 3, p. 1603–1606 Vol.3. DOI: 10.1109/ICME.2004.1394556.

[34] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, vol. 17, p. 261–272. DOI: 10.1038/s41592-019-0686-2.

[35] Wu, J., Martin, A., Greenberg, C. S. and Kacker, R. The Impact of Data Dependence on Speaker Recognition Evaluation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2017, vol. 25, p. 5–18.

[36] Yujian, L. and Bo, L. A Normalized Levenshtein Distance Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2007, vol. 29, no. 6, p. 1091–1095. DOI: 10.1109/TPAMI.2007.1078.

[37] Zbilut, J. P. and Webber Jr., C. L. Recurrence Quantification Analysis. In: *Wiley Encyclopedia of Biomedical Engineering*. American Cancer Society, 2006. DOI: https://doi.org/10.1002/9780471740360.ebs1355. ISBN 9780471740360. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780471740360.ebs1355.

[38] Černocký, J. *Zpracování řečových signálů — studijní opora* [http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf]. Brno University of Technology, december 2006. 100-108 p. (Accessed on 01/27/2021).

Appendix A

Contents of the included storage media

This appendix lists contents of the attached CD:

- `sub_eval.zip` – Zip file with the "sub_eval" dataset used in the experiments. Contains *eval_clear* – `wav` audio data without pre-recorded messages and *eval_goal* – `wav` audio data with mixed pre-recorded messages. The subfolders include processed phoneme posteriors, bottleneck features in `htk` format, and phoneme strings in `txt` format.
- `messages/` – Original unmixed messages with transcription.
- `scripts/` – Created and used source codes.
  The folder `scripts/` contains these subdirectiories:
  - `scripts/evaluations/` – The scores, processing time and created objects (persistent Python objects (RQA list, cluster analysis) in `pkl` format) from the experiments.
  - `scripts/third_party_scripts/` – The folder contains third party source codes.
- `text/` – LaTeXsource codes of the thesis. The PDF can be created with the command `make pdf`.
- `xbobos00.pdf` – This thesis in the PDF format.

Appendix B

Manual

This appendix describes installation process and user manual. The setup is tested, and it is working on Ubuntu 20.04 machines.

B.1 Installation manual

To have properly working scripts, it is necessary to install Python dependencies. Installation can be done using the following commands:

```
cd scripts/
pip install -r requirements.txt
```

Scripts for editing audio files need `ffmpeg` utility. This dependency can be installed by `sudo apt-get install ffmpeg`.

Additionally, in the case of creating phoneme posteriors, it is necessary to install the phoneme recogniser, from https://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context. After successful installation, change path to installed phoneme recogniser in script `/scripts/set_phnrec`.

B.2 User manual

Creation of a simulated dataset

The creation of a simulated dataset is accomplished by two scripts `split.py` and `mix.py`.

Script split.py

Python script *split.py* is responsible for cutting phone calls into smaller parts.

There are many arguments to specify wanted result:

- `--sec` – Sets the length of the cut part (in seconds).
- `--src` – Path to source directory containing the uncut phone calls.
- `--dst` – Path to destination directory where the cut phone calls will be saved.
- `--lt, --st` – Will cut longer/shorter recordings than specified (in seconds).

- `--rm` – Remove original file after splitting.
- `--stats` – Shows statistics about the files from given source.
- `--move, --copy, --count` – Will move/copy "count" recording.

Example command for splitting files to 180-second-long segments:

```
python3 split.py --src=/source_folder/ --dst=/destination_folder/ --sec=180
--lt=100 --st=500 --rm
```

Script mix.py

Python script *mix.py* is responsible for mixing pre-recorded telephone operator messages into phone calls.

First, the given message is edited in repetitions, speed, volume, and afterwards, it is inserted into the phone calls. Arguments to specify wanted result include:

- `-m, --mode` – Specifies, which type of pre-recorded message you want to use (possible values are: A, B, C).
- `--mpath` – Path to the directory with the pre-recorded messages.
- `--spath` – Path to the directory with the speech .wav files.
- `--export` – Path to the directory to export the mixed audio files.
- `--lt, --st` – Will use longer/shorter recordings than specified (in seconds).
- `-g, -s, -r` – Change volume/speed/repetitions of the pre-recorded messages.
- `--random` – severity of randomness. Possible values are:
  - *0* – no random values, use the original values of the pre-recorded message
  - *1* – low differences (gain between -3dB+3dB, speed 0.95-1.05, repeat 1-10.0)
  - *2* – optimal differences (gain between -6dB+6dB, speed 0.9-1.1, repeat 0.8-30.0)
  - *3* – high differences (gain between -10dB+6dB, speed 0.85-1.15, repeat 0.7-40.0)
- `--onlymodify` – Will modify the operator message only, without mixing into phone calls.
- `-stats` – shows statistics about the files from given source

Example command for mixing pre-recorded messages of type A to audio recordings:

```
python3 mix.py --mpath=/messages/ --spath=/telephone_conversations/
--export=/mixed_conversations/ -m=A --lt=100 --st=150 --random=2 -g -s -r
```

Experiments

This section provides detailed instructions on how to obtain the results presented in Chapters 5, 7 and 8. The process can be described as follows:

To start the classification process, it is necessary to set a source directory containing goal (with mixed messages) and clear (pure phone calls) data. This is accomplished by `--src` argument.

It is necessary to specify the system to classification by the `--system` argument. The options are as follows:

- Baseline DTW system (in 5.1) – `--system=basedtw`
- Baseline RQA system (in 5.2) – `--system=rqa_unknown`
- Baseline FSM system (in 5.3) – `--system=fuzzy_match_base`
- RQA+DTW system (in 7.1.1) – `--system=rqa_dtw_unknown`
- RQA+SDTW system (in 7.1.2) – `--system=rqa_sdtw_unknown`
- FSM pause analysis system (in 7.2.1) – `--system=fuzzy_match_pau_analysis`
- RQA+SDTW clustering+SDTW system (in 8.2)
  - unknown messages scenario: `--system=rqacluster_sdtw_unknown`
  - known messages scenario: `--system=cluster_sdtw_known`
- RQA+SDTW clustering+FSM system (in 8.3)
  - unknown messages scenario: `--system=rqacluster_fuzzy_match_unknown`
  - known messages scenario: `--system=rqacluster_fuzzy_match_known`

To be clear, all systems using the RQA list of candidates search for the file *evaluations/objects/rqa_list_FEAT.pkl*, where "FEAT" is the one specified in *--feature* argument. If the file is not present, the system will create one. The same applies to other systems using SDTW clusters, where the system looks for the file *evaluations/objects/cluster_rqa_list_CLUSTFEAT.pkl*, where "CLUSTFEAT" is the feature specified by *--cluster-feature*. It is required to specify used type of feature arrays:

- RQA, DTW/SDTW system – `--feature=mfcc/posteriors/bottleneck`
- FSM system – `--feature=string`
- Systems using SDTW clusters need additional parameter – `--cluster-feature=mfcc/posteriors/bottleneck`

The systems using frame averaging are specified with `--frame-reduction=NUM`.
Parallelisation is accomplished by `--parallelize-from` and `--parallelize-to` parameters. The parameters specify which files are processed by the system.
The example command can be as follows:

```
python3 main.py --src=../sub_eval/ --system=rqacluster_fuzzy_match_unknown
--feature=string --cluster-feature=bottleneck --parallelize-to=200
```