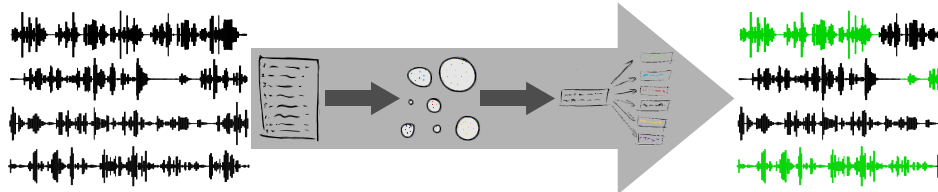


Detection of pre-recorded messages in speech

Dominik Boboš*



Abstract

Recognition of pre-recorded messages in speech such as "This number is not reachable" is useful for any follow-up speech data mining. To investigate the identification of redundant information in audio, it is necessary to have a large amount of data with the exact phrases repeated multiple times. Such a set is generated by mixing pre-recorded messages into phone calls with variations in speed, volume and repetitions. The created 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.

Keywords: Detection of re-occurring sequences in audio — Segmental dynamic time warping — Recurrence quantification analysis — Fuzzy string matching — Bottleneck features — Phoneme posteriors

Supplementary Material: [Downloadable Code](#)

*xbobos00@stud.fit.vutbr.cz, Faculty of Information Technology, Brno University of Technology

1. Introduction

The typical scenario for investigating audio recordings starts with a large dataset. Storing a large amount of speech data comes with a lot of disadvantages. On one hand, the low-level problems like running out of free space. On the other hand, listening through unnecessary, repetitive data or time-consuming automatic processing.

Since these recordings are often telephone calls, many share typical features. For instance, ringing, voicemail message, music on hold or any pre-recorded message. The more recordings of telephone conversations, the higher the probability of occurrence of mentioned shared segments. These typically include pre-recorded operator messages (for instance, "Thank you for calling, please leave a message.", "Sorry the number you are calling does not answer at the moment, please try again later").

Detection could improve productivity for many professions, such as law enforcement agencies (LEA) or call centres. Hence the main objective is to minimise wasting time by listening to redundant information in speech data and decrease processing time by further automatic processing. That means that the requirement for the system is to have a fast and accurate solution with a minimum hardware load. Another requirement is to have a robust system working independently on the language, as with low-resource languages, it is not possible to use such techniques as automatic speech recognition.

To find similar or identical parts effectively in a large dataset with nearly zero knowledge, it is essential to choose the methodology for tagging detected pre-recorded operator messages. Either mark an exact time in the phone call or provide a binary decision only – whether the given recording contains pre-recorded

19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

37 speech.

38 This paper presents two main scenarios how to
39 achieve this task:

- 40 1. **Known messages scenario** – This scenario is
41 used to create *reference clusters*. The cluster
42 analysis is accomplished by provided labelled
43 pre-recorded messages. Recordings are com-
44 pared to reference clusters.
- 45 2. **Unknown messages scenario** – Here the system
46 does not know the pre-recorded messages and
47 has to infer them as repeated parts of calls. This
48 approach can be divided into the following tasks:

- 49
- 50 (a) Detection based on *a recording itself* –
51 The process does not require any addi-
52 tional information, and the pre-recorded
53 message is detected in itself.
- 54 (b) Detection based on *all files* – Recordings
55 are compared to all recordings in the set or
56 by a chunk of the set. The new chunk is
57 randomly chosen with every recording.
- 58 (c) Detection based on *clusters* – Recordings
59 are compared to the created clusters. The
60 cluster analysis requires list of candidates.

61 This paper combines several techniques for search-
62 ing repeating sequences in speech. From the basic ones
63 like Dynamic Time Warping (DTW) – to search the
64 distance between two recordings, to more advanced
65 ones like Recurrence Quantification Analysis (RQA) –
66 which analyses diagonal line segments in recurrence
67 matrix. Approximate string matching (also known as
68 Fuzzy string matching) is used to analyse the similar-
69 ity between two phoneme strings. The best accuracy
70 while preserving low processing time is achieved by
71 the combination of all mentioned techniques.

72 2. Used techniques

73 2.1 Dynamic Time Warping

74 Even though the same person says the same sentence
75 twice, it will never sound exactly the same. It may
76 vary in length, speed, intonation, volume, pitch etc.
77 Linear frame-by-frame matching of two sequences
78 will fail, even though it is the same sentence. *Dynamic*
79 *time warping (DTW)* is an algorithm used to find the
80 shortest distance and compare two time series when
81 the time indices are not synchronised, the standard
82 procedure is explained in [1]. The comparison between
83 linear matching and DTW distance matching is shown
84 in Figure 1.

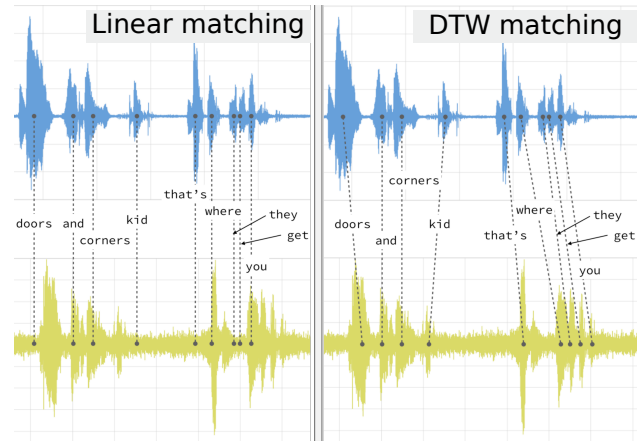


Figure 1. Comparison between linear matching and DTW matching on two speech recordings. [2].

2.2 Segmental DTW

DTW is finding one global optimal alignment path between the whole two sequences. This may be an issue for detecting similarities in sub-sequences. In Aren Jansen's work, *Segmental Dynamic Time Warping (S-DTW)* is presented as a solution [3]. 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) *Global constraints*, which are restricting space a warping path can take while 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* [4]. A comparison between DTW and S-DTW for two utterances is shown in Figure 2.

2.3 Recurrence quantification analysis

Recurrence quantification analysis (RQA) is a method of nonlinear data analysis. RQA calculates the value of path alignments by dynamic programming [5]. RQA provides objective quantification of important aspects revealed by the plot – recurrence matrix (RM). Points in an RM that form diagonal line segments are considered to be deterministic (apart from the isolated points) [6]. The method is similar to DTW. However, instead of finding the shortest alignment path, in the RQA analysis, the longest alignment paths are selected. RQA quantifies the structure of RM by several metrics which are used as a weights [7]. The used recipe for RQA is described in [5].

2.4 Fuzzy string matching

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. *Fuzzy string matching (FSM)* (also known as Approxi-

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

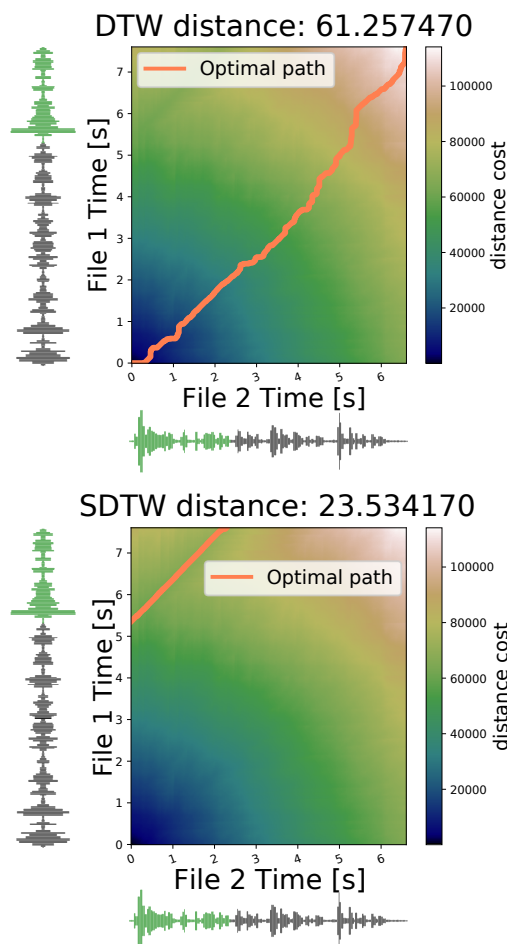


Figure 2. DTW and S-DTW alignment paths of the two utterances, where File 1 contains the phrase "Brno University of Technology" at the beginning, while File 2 at the end.

They were split into three categories: A) messages to occur at the beginning of the call, B) messages to occur anywhere through the call and C) messages to occur both at the beginning and the end of the call. All pre-recorded messages were pre-processed (removal of the initial silence and volume normalisation).

The next step is to mix the telephone operator messages with **517.13** hours of phone calls from the Switchboard corpus¹ in total of 4870 conversations. The conversations were trimmed to shorten calls to uniform (in terms of hours) eight categories from a 15-second-long to 180-second-long calls. For the mixing process, approximately 10% of the trimmed phone calls were chosen – 4260 calls. Also, 200 recordings of zero length calls were chosen – the total number of calls is then **4460**.

To get close to realistic data, each pre-recorded message is mixed into calls with varying speed (between 0.9 – 1.1), volume gain (between -6dB – +6dB) and varied repetition (between 0.8 to 30.0). The average length of a mixed pre-recorded message is around one minute. The total count of the mixed pre-recorded messages is **4460** files of a total length of **150.66 hours**. The metadata of the changes made to the messages are preserved in the filename.

The audio files are in 16-bit wave format with a standard sample rate of **8000 Hz**. The prepared recordings were split into training and evaluation sets and their subsets used for development – total numbers are presented in table 1. 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, "Train" and "Sub train" sets are created but not used in evaluations [9].

Table 1. Total lengths in hours and counts of audio files in individual sets.

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

mate string matching) is an algorithm for comparing two strings approximately. 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 [8].

3. Simulated dataset

No publicly known solution exists for the problem, that means no dataset as well.

To create a simulated dataset, 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 are in English, Czech and Slovak.

4. Baselines

This section provides details about the baseline systems used for detecting the pre-recorded messages. Their input can be i) either the recording itself (used in RQA approach) and ii) pre-recorded messages

¹LDC Switchboard-1 Release 2:
<https://catalog.ldc.upenn.edu/LDC97S62>

181 detection on all files or chunks of the set (used in DTW
182 and FSM approaches). The results provide binary de-
183 cision only – either the audio contains the message or
184 not.

185 For the work, I used standard MFCC features²,
186 phoneme posteriors³ and bottleneck features⁴.

187 *Detection error trade-off* (DET) curves [12] and
188 *Equal error rate* (EER) [13] are used for evaluation.
189 For measuring the quality of clustering, metrics like
190 purity, rand index, normalised mutual information are
191 used, more in book [14].

192 4.1 DTW approach

193 DTW gives optimal results when comparing similarly
194 long time series with isolated words. However, in our
195 case, the recordings are of various lengths. Also, the
196 similar parts are repeated unevenly, and the position
197 of a pre-recorded message could be anywhere in the
198 file. The first important step is to find candidates of
199 similar parts between two recordings on the DTW
200 alignment path and then find DTW distance between
201 the candidates.

202 Searching for candidates

203 To find the candidates for determining similarity, the
204 DTW warping path is computed first and then analysed.
205 The algorithm finds the similarities by looking back
206 to the previous steps the warping path has taken. In a
207 DTW warping path, 3 moves are possible: i) *diagonal*,
208 ii) *horizontal* and iii) *vertical*. Three types of moves
209 that a two-step pair can get:

- 210 • The “*good trend*” (GT) type happens when the
211 current step is in diagonal direction. The direc-
212 tion of the previous step does not matter in this
213 case.
- 214 • The “*false trend*” (FT) type happens when the
215 two-step pair consists of two horizontal moves or
216 two vertical moves.
- 217 • The “*neutral trend*” (NT) type happens when
218 the two-step pair consists of one vertical and
219 one horizontal moves or one is in the diagonal
220 direction.

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

²Python package `python_speech_features`: <https://python-speech-features.readthedocs.io/en/latest/>

³Phoneme recogniser based on long temporal context [10]: <https://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context>

⁴BUT/Phonexia Bottleneck feature extractor [11]: <https://speech.fit.vutbr.cz/software/but-phonexia-bottleneck-feature-extractor>

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

230 The detection ends when the constant false trend
231 is larger than the given threshold σ . Then the quality
232 of the detected part of a sequence is evaluated. First,
233 the length of the list needs to be longer than δ frames,
234 which means $\delta/100$ seconds. Sequences longer than
235 δ are scored by the ratio of the triangle created by the
236 warping path, y-axis and x-axis. If the ratio falls in
237 given interval τ , the detected line is a good candidate.
238 The whole process repeats until the end of the warping
239 path. In the presented baseline system, the variables
240 are set accordingly: $\sigma = 25$, $\delta = 200$, and $\tau \in \langle 0.9, 1.1 \rangle$.
241 The behaviour and the results of the algorithm is shown
242 in Figure 3.

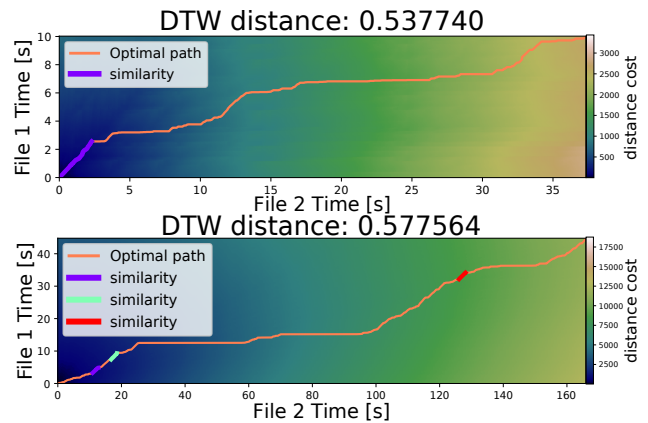


Figure 3. DTW path with applied candidates detection algorithm. The top one with one candidate is later classified as a hit. The bottom alignment obtains several candidates, but none of them resulted in a hit.

243 The proposed solution for the baseline DTW sys-
244 tem is to compare every feature vector and its found
245 candidates with all others. By using a brute-force ap-
246 proach, it is immediately an $O(n^2)$ problem. However,
247 a little trick is performed to get a better constant. Each
248 file is compared to a randomly chosen chunk of all
249 recordings. The chunk size is 1/8 of the set. As the
250 idea of detecting the pre-recorded messages is to find
251 parts repeated several times, at least one should appear
252 in the reduced bunch. Performance of DTW system
253 does not meet the required accuracy and fast process-
254 ing time as shown in table 2.

255 4.2 RQA approach

256 The RQA approach is based on Aren Jansen’s work [3].
257 RQA analysis expects the similarity matrix at the input
258 – a cosine distance is used for creating the matrix. The
259 self-similarity diagonal is removed. Default settings

Table 2. Results of the baseline DTW system on "Sub eval" dataset.

Features	Average time of one file [seconds]	CPU-core time [hours]	EER
MFCC	244.39	8.46	41.17%
Phoneme posteriors	284.30	9.79	30.48%
Bottleneck	477.49	16.53	29.41%

for RQA from Librosa package⁵ are used, together with *affinity mode* and *knight moves* on.

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 alignment path from RQA analysis is scored by the sum of the similarity points of the longest similarity path, divided by the length of the path. Scores below the threshold ϕ are marked as a hit. Lower processing time is achieved by frame reduction (FR) optimisation (more in subsection 5.1). MFCC features perform significantly the best over the phoneme posteriors and bottleneck features. The average processing time for one file and overall performance is presented in table 3.

Table 3. Processing time and evaluation result of RQA system with different settings. FR stands for frame reduction, M means MFCC features, PP – phoneme posteriors, BN – bottleneck features.

Features Dataset	Average time [seconds]			Total time [minutes]			EER [%]		
	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
Sub eval FR=10	1.61	2.36	2.11	10.03	14.72	13.17	12.30	20.86	22.99
Sub eval FR=20	1.05	1.68	1.55	6.52	10.45	9.65	32.62	28.88	27.81
Eval FR=5	1.92	2.17	2.91	365.73	412.84	544.65	1.34	6.68	10.38

⁵Librosa 0.8.0 on Zenodo:
<https://zenodo.org/badge/6309729.svg>

4.3 FSM approach

Fuzzy phoneme string matching approach is based on *Levenshtein distance*. First, the text file output from a phoneme recogniser by Petr Schwarz [10] is imported and parsed. The example of the phoneme recogniser output can be as following:

```
0 1300000 s -12.662029
1300000 2700000 e -13.111244
2700000 14900000 pau -118.012054
14900000 15300000 n -3.924468
```

The first two columns represent time interval of a phoneme, where 1 second is represented as 10000000. The third column shows an occurred phoneme. The last one provides the duration of the occurred phoneme. This output is parsed into list of lists with following structure:

`[[frames], [lengths], [index_map], string]`

, where `[frames]` represents the list of intervals (with trimmed 5 zeros from the end), `[lengths]` is the list of phoneme durations in hundredths of seconds, `[index_max]` is the list of occurred phonemes at the corresponding index, `string` is the phoneme string of the whole recording with replaced "pau" labels to spaces, which is used in comparison. The output from the example would be parsed into:

```
[
  [[0, 13], [13, 27], [27, 149], [149, 153]],
  [12.662, 13.111, 118.012, 3.924],
  ['s', 'e', 'pau', 'n'],
  "se n"
]
```

Such a parsing allows to convert between features and to return just needed parts of a string. A *partial ratio* function from *FuzzyWuzzy*⁶ package tackles the problem with uneven phoneme string lengths. Let assume a pair of strings of the same pre-recorded message. One is repeated three times, the other one is repeated once. Basic ratio function returns score 60%, while partial ratio function returns 84%. However, this approach causes that even one word "you" compared to a whole sentence with "you" somewhere returns a score of 100%. This can be fixed by applying brute force – ignore 100% scores, as it is almost certain that it is shown case. It is important to realise that even the same telephone operator messages will not probably return a score of 100%. The algorithm is based on

⁶FuzzyWuzzy 0.18.0 on PyPI:
<https://pypi.org/project/fuzzywuzzy/>

comparing each phoneme string to another – same as the DTW approach. Accordingly, it is optimised by chunking. The size of randomly chosen chunks from the set is 1/4 of all recordings.

Table 4. Performance of baseline fuzzy string matching system.

Dataset	Average time [seconds]	Total time [hours]	EER [%]
Sub eval	29.57	3.07	18.18%
Eval	37.14	102.53	33.14%

5. Experiments

This section describes experiments and results to proposed baselines, clustering techniques and optimisations.

5.1 Optimisations

Two main optimisations are used for decreasing processing time: i) caching and ii) frame averaging.

Caching is the process of storing copies of files in a temporary storage. 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 rapidly reduce processing time by almost 80% in some cases.

Frame averaging (or reduction) is a process of reducing the size of the feature vector. Two proposed methods for frame averaging is presented: i) dimensions reduction and ii) reduction in a time axis.

Dimensions reduction – The posteriors feature vector represents three states for each one phoneme, making it 138 elements long. To reduce computation time, posteriors of triplets of states are summed to create posterior of one phoneme class.

Reduction in a time axis – Each second in an array is represented by hundred frames. The idea is to get a mean value of n frames to minimise computation time while preserving as much information as possible. Used in RQA approach and S-DTW clustering.

5.2 Clustering

Two types of clusters are used in evaluation: i) reference cluster – created by labelled pre-recorded messages (known-messages scenario) and ii) predicted cluster – created by list of candidates from RQA analysis (unknown-messages scenario).

Clustering is performed to improve accuracy and decrease processing time. Clustering removes the necessity to compare to every candidate and compare to representatives of cluster classes only.

The clustering process consists of two steps. First step – Voice Activity Detection (VAD) and filtering⁷. RQA analysis is performed and only non-empty list of frames from the analysis are preserved. Next, VAD is applied to the output from the analysis.

Second step – dividing candidates into classes by using S-DTW⁸. S-DTW clustering is performed on the filtered RQA analysis from step one. The clustering is based on Agglomerative Hierarchical Clustering (AHC). Every cluster is sorted by the lengths of the element – the shortest recordings are at the top.

The process of creating reference (ground-truth) clusters for evaluation and experiments 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). Every cluster is sorted the same as the automatic one. The performance of the clustering is shown in tables 5, 6. MFCC provides the best processing time, however bottleneck features shows decent accuracy even with high frame reduction. The best speed-accuracy trade-off is with the $FR = 20$.

Table 5. Clustering performance evaluation by several metrics on different features like MFCC features (M), phoneme posteriors (PP), and bottleneck features (BN) with various settings of frame reduction (FR).

Metric	Purity			Rand Index			NMI		
	M	PP	BN	M	PP	BN	M	PP	BN
Sub eval FR = 10	0.88	0.87	0.82	0.99	0.98	0.98	0.93	0.92	0.92
Sub eval FR = 20	0.66	0.68	0.81	0.93	0.96	0.98	0.82	0.80	0.92
Sub eval FR = 30	0.43	0.43	0.80	0.90	0.86	0.98	0.68	0.56	0.92
Sub eval FR = 40	0.31	0.33	0.77	0.83	0.75	0.97	0.54	0.45	0.90
Eval FR = 20	0.55	0.66	0.77	0.88	0.95	0.97	0.74	0.75	0.87

5.3 Combination of all techniques

The final 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 and S-DTW AHC clustering. This experiment is performed by i) clustering and ii) FSM evaluation afterwards.

Clustering – the same process as described in subsection 5.2, where frame reduction of $FR = 20$ is used.

⁷Used VAD for filtering – py-webrtcvad on GitHub (used under MIT license): <https://github.com/wiseman/py-webrtcvad>

⁸S-DTW implementation based on [4] by gray0302 on GitHub. The implementation is modified to suit the needs: <https://github.com/gray0302/seg-dtw>

Table 6. Processing time of RQA system.

Features	Total Time [minutes]		
	M	PP	BN
Sub eval FR = 10	18.30	141.04	288.90
Sub eval FR = 20	4.93	28.51	58.20
Sub eval FR = 30	2.20	8.67	17.93
Sub eval FR = 40	1.76	6.52	11.15
Eval FR = 20	183.57	237.97	217.63

Used S-DTW clusters are created from the "Eval" dataset from all available features.

FSM evaluation – each testing file is compared to the first three elements of each cluster class. Used FSM in the experiment is modified to apply so called *pause analysis*. In pause analysis, all silent parts longer than 2 seconds are declared as dividing points. Then the lengths of the segmented lists are checked. If the segment is shorter than 50 elements of the list, the segmented part is removed from the candidates. 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. Results of the final system is shown in Figure 4 and table 7.

Table 7. Results of final system. The system works with both known messages (reference cluster) and unknown messages (S-DTW cluster).

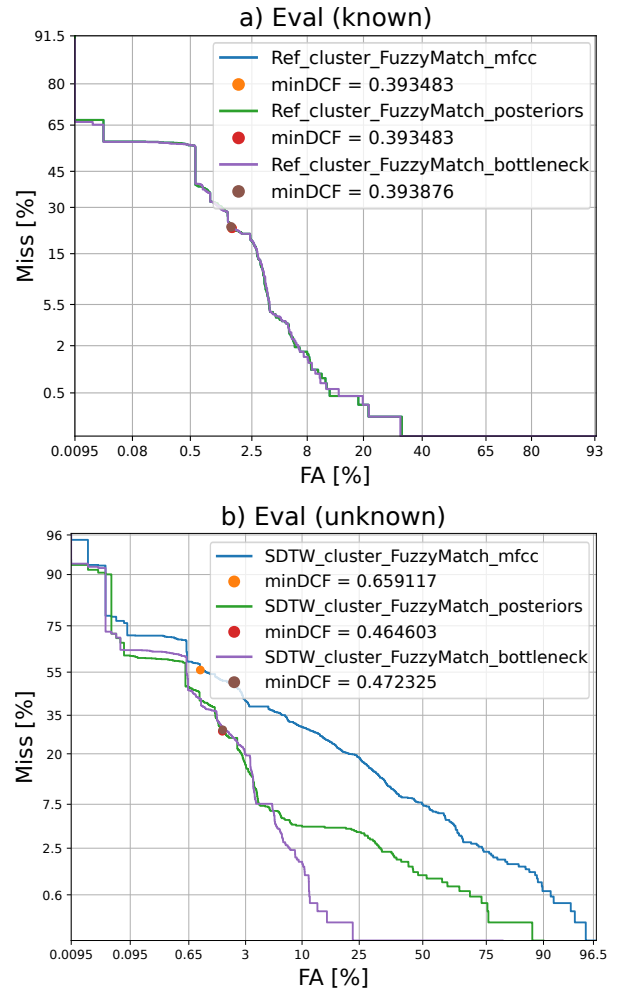
Ref cluster	Average time [seconds]			Total time [hours]			EER [%]		
	M	PP	BN	M	PP	BN	M	PP	BN
Features	M	PP	BN	M	PP	BN	M	PP	BN
Sub eval	6.68	5.98	5.96	0.69	0.62	0.62	1.60	1.60	1.60
Eval	17.35	12.06	19.78	9.12	6.30	10.15	4.28	4.28	4.30

S-DTW cluster	Average time [seconds]			Total time [hours]			EER [%]		
	M	PP	BN	M	PP	BN	M	PP	BN
Features	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
Eval	1.57	1.57	1.43	4.99	6.39	4.54	20.48	6.34	5.77

6. Conclusion and Future Work

In this paper, methods for detecting repetitive parts across audio recording sets were presented. The research aimed especially at searching for pre-recorded telephone operator messages in speech conversations.

This paper took a deeper look at three techniques: Dynamic time warping, recurrence quantification analysis and fuzzy phoneme string matching. The main idea was to focus on the techniques when the system runs without the knowledge of the messages. The main goal was to find the most accurate and fastest approach.

**Figure 4.** Performance of the S-DTW cluster + fuzzy string matching system on the "Eval" dataset. The system works with both known messages a) and unknown messages b).

To decide which system is the best, it was necessary to simulate a dataset. It was accomplished by mixing the operator messages into the Switchboard corpus while changing the speed, volume gain and a number of repetitions.

RQA performs the best among the three baseline methods. The bottleneck features bring the highest accuracy and MFCC features the fastest processing time. To decrease computation time, caching and frame averaging was applied. In the experiments, the best performance is achieved by all techniques combined.

For future research, voice biometrics techniques for creating voice-prints of a speaker is the point of interest and how to integrate the method into the presented workflow.

Acknowledgements

I would like to thank my supervisor prof. Dr. Ing. Jan Černocký for valuable suggestions and friendly approach.

- [1] B. J. Mohan and N. R. Babu. Speech recognition using MFCC and DTW. *2014 International Conference on Advances in Electrical Engineering (ICAEE)*, pages 1–4, 2014.
- [2] R. Portilla, B. Heintz, and D. Lee. Understanding Dynamic Time Warping - The Databricks Blog. <https://databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html>, 2019. (Accessed on 02/26/2022).
- [3] A. Jansen, K. Church, and H. Hermansky. Towards spoken term discovery at scale with zero resources. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, pages 1676–1679, 2010.
- [4] A. S. Park. *Unsupervised pattern discovery in speech: Applications to word acquisition and speaker segmentation*. PhD thesis, Massachusetts Institute of Technology, 2006.
- [5] J. Serrà, X. Serra, and R. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11:093017, 2009.
- [6] J. P. Zbilut and Charles L. Webber Jr. *Recurrence Quantification Analysis*. American Cancer Society, 2006.
- [7] A. D. Likens, K. S. McCarthy, L. K. Allen, and D. S. McNamara. Recurrence Quantification Analysis as a Method for Studying Text Comprehension Dynamics. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge, LAK '18*, page 111–120, New York, NY, USA, 2018. Association for Computing Machinery.
- [8] L. Yujian and L. Bo. A Normalized Levenshtein Distance Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095, 2007.
- [9] D. Boboš. *Detection of Pre-Recorded Messages in Speech*, 2021. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor J. Černocký.
- [10] P. Schwarz. *Phoneme recognition based on long temporal context*. PhD thesis, Brno University of Technology, Faculty of Information Technology, 2009.
- [11] R. Fér, P. Matějka, F. Grézl, O. Plchot, K. Veselý, and J. Černocký. Multilingually trained bottleneck features in spoken language recognition. *Computer Speech & Language*, 46:252–267, 2017.
- [12] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and Mark A. Przybocki. The DET curve in assessment of detection task performance. In *EUROSPEECH*, 1997.
- [13] N. Singh, Prof. R. Khan, and R. S. Pandey. Equal error rate and audio digitization and sampling rate for speaker recognition system. *Advanced Science Letters*, 20, 2014.
- [14] C. D. Manning. *Introduction to Information Retrieval*. Cambridge University Press, 2008.