Unsupervised Motif Discovery from Acoustic Time Series Data

Chantriolnt - Andreas Kapourani

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2013
Abstract

The current state-of-the-art Automatic Speech Recognition (ASR) systems are now capable of performing large-vocabulary continuous speech recognition based on the assumption that have sufficient training resources. But in many languages these annotated training data resources are absent. The aim of this project is to discover and extract linguistic phrases or words from input speech data in an unsupervised manner. Thus, we form the problem of discovering words as a motif discovery problem, where we want to identify repeated patterns in longer time series data, which in our case are MFCCs. The method that we re-implemented discretizes the time series data using Symbolic Aggregate approXimation (SAX) method [Chiu et al., 2003] and then it applies the Random Projection algorithm [Buhler and Tompa, 2002] to efficiently discover motif seeds. Finally, it scans all the time series data to discover additional motif occurrences.
Acknowledgements

Initially I would like to express my sincere thanks to my supervisor, Dr. Sharon Goldwater, for all the advice and encouragement she has given me throughout the duration of this project. The interest she showed and the ideas she gave me in dealing with the problems that occurred during these three stressful months were crucial for the understanding and implementation of the project.

Undoubtedly I would like to thank my family for the incomparable patience, understanding and goodwill that they showed throughout the preparation of my dissertation.

This dissertation is dedicated to the memory of the late former dean and professor George Karabatzos.
Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(Chantriolnt - Andreas Kapourani)
# Table of Contents

1 Introduction .................................................. 1  
  1.1 Motivation .............................................. 1  
  1.2 Outline ................................................. 2  

2 Background .................................................. 3  
  2.1 Definitions and Notation ............................. 3  
  2.2 Dynamic Time Warping ............................... 5  
  2.3 Related Work ........................................ 8  

3 Experimental Background ................................ 12  
  3.1 Speech Corpora ....................................... 12  
  3.2 Feature Extraction .................................. 13  
  3.3 Evaluation Method ................................... 14  
  3.4 Evaluation Metrics .................................. 16  

4 Motif Discovery Algorithm ............................... 18  
  4.1 Motif Discovery via SAX and Random Projection . . . . 18  
    4.1.1 Dimensionality Reduction and Subsequence Discretization . 19  
    4.1.2 Random Projection ................................ 21  
    4.1.3 Discovering Motif Occurrences .................. 23  
  4.2 Performance and Analysis ............................ 26  
    4.2.1 Results .......................................... 27  
    4.2.2 DTW vs Euclidean Distance .................... 34  

5 Two Phase Neighbourhood Range ......................... 36  
  5.1 Performance and Analysis ........................... 38  
    5.1.1 Results .......................................... 38
List of Figures

2.1 A visual intuition of a time series $T$ (light line), a subsequence $C$ (bold line) and a subsequence $M$ that is a match to $C$ (bold gray line indicates the $C$ subsequence) [Chiu et al., 2003]. .......................................................... 4

2.2 For almost any subsequence $C$ in continuous time series data, the best matches are the trivial subsequences immediately to the left and right of $C$ [Lin et al., 2002]. ................................................................. 4

2.3 A) Two sequences $Q$ and $C$ which are similar, but out of phase. B) To align the sequences we construct a matrix, and search for the optimal warping path, shown with solid squares. C) The resulting alignment given by applying DTW [Keogh, 2002] .......................................................... 6

2.4 Imposing a global constraint on the warping path using the Sakoe-Chiba band. The admissible warping paths are those that are within the grey area [Keogh and Ratanamahatana, 2004] ................................................. 7

2.5 Subsequence density shown in 1-dimention. The dots represent the different subsequences. The two motifs are discovered in the high density regions [Minnen et al., 2007a] ................................................................. 11

3.1 Block diagram of the steps needed to transform the raw speech data to MFCC vectors [Renals, 2013]. ................................................................. 13

4.1 PAA representation approximates the normalized continuous time series with a linear combination of box basis functions. The time series of length 128 is reduced to 16. ................................................................. 19

4.2 A continuous time series converted into a SAX word ‘bababddde’ using eight PAA segments and four SAX symbols as alphabet size. ............... 20
4.3 A single iteration of the Random Projection algorithm. (a) A subset of string symbols are selected (positions one and three). (b,c) The symbols are hashed and (d) collisions are recorded in the collision matrix by incrementing the count in the appropriate cell [Minnen et al., 2007b].

4.4 A visual explanation of why each motif should be const \( R \) apart. If the motifs are only required to be \( R \) distance apart as in A, then the two motifs may share the majority of their elements. In contrast, B illustrates that requiring the centres to be const \( R \) apart insures that the motifs are unique [Lin et al., 2002].

4.5 This graph shows the frequency of different entries in the collision matrix, where the frequencies are in log scale. For example, we saw about 22,250,000 times number 0 (zero occurrences) and only 7000 times the number 2 (two occurrences) in the collision matrix. We observe that the distribution of occurrences in the collision matrix follows the power law distribution.

4.6 Two utterances from the TIDIGITS corpus by showing the 13 MFCC vectors. The one on the top is utterance '75443z7' and the other is '8045647'. The highlighted regions are the occurrences of the digit '4' and show its length variation. In this example the smallest duration is 22ms and the highest 51ms.

4.7 Above, MFCCs for the utterance ”three seven one three four six six”. Below two levels of bars, the top has the ground truth labels, whereas the bottom has the discovered motif occurrences. The discovery algorithm failed to detect the digits ”one” and ”four”, and it discovered one extra occurrence that was not expected. We should notice the difference in the length of the occurrences, in our algorithm we have fixed-length motif occurrences, whereas the ground truth labels are variable-length.

4.8 Performance of the different subsequence lengths (20, 24, 25, 28, 30). We should note that the graph starts at 40% performance.

4.9 Performance of the discovery algorithm, accuracy and F-measure, for different values of neighbourhood range \( R \). We clearly observe the instability of the algorithm for small changes in the value of \( R \).
4.10 Performance of the discovery algorithm for DTW and Euclidean distance, red and blue colour respectively. As we observe accuracy is higher when we use the DTW distance to discover motif occurrences. We should note that the graph stops at 60% accuracy.

5.1 On the left a single neighbourhood range may not capture ideally the different motifs. When the range is allowed to vary, as shown on the right, then each motif can adapt and ideally include all the valid occurrences from the time series data [Minnen, 2008].

5.2 Accuracy of the extended method versus the baseline. We observe that the two-phase neighbourhood range method has a better performance. We should note that the graph stops at 70% accuracy.

5.3 Performance of the two phase algorithm for different subsequence lengths. We should note that the graph start at 40% performance.

6.1 Comparing two subsequences $Q$ and $C$ of different length. A direct comparison (left) yields a larger Euclidean distance. But, as shown on the right, stretching subsequence $Q$ by a scaling factor $s$, produces $Q'$ which resembles very closely the prefix of subsequence $C$ [Yankov et al., 2007].
## List of Tables

4.1 *User specified parameters for the discovery algorithm* ........................................... 26  
4.2 *Choice of subsequence length and PAA segments.* ............................................. 28  
4.3 *Best performance with the optimal combination of parameter values.* ................. 29  
4.4 *Optimal combination of parameter values.* ......................................................... 29  
4.5 *Results and parameter values for different subsequence lengths.* ....................... 31  
4.6 *Fixed parameter values.* ....................................................................................... 32  
4.7 *Different values for projection size* \( k \). We notice that as we increase \( k \), the number of discovered motifs decreases. ......................................................... 32  
4.8 *Different values for threshold parameter* \( t \). We observe that as we increase \( t \), the number of discovered motifs decreases. ................................................. 33  
4.9 *Different values for the* \( \text{const} \) *parameter. We observe that a small increase in the value has a great effect on the number of discovered motifs and consequently on the performance of the algorithm.* ................. 34  

5.1 *Best performance with the optimal combination of parameter values for the two-phase neighbourhood range.* ......................................................... 38  
5.2 *Optimal combination of parameter values for the two-phase neighbourhood range.* ......................................................... 38  
5.3 *Different values for the* \( p \) *parameter. In this example the optimal choice for the parameter* \( p \) *is 1.25.* ................................................................. 39  
5.4 *Combination of parameter values for the two-phase neighbourhood range used for the results shown in Table 5.3.* ........................................... 39  

6.1 *Parameters that could be learned off-line in a completely unsupervised way.* ......... 42
Chapter 1

Introduction

1.1 Motivation

The current state-of-the-art Automatic Speech Recognition (ASR) systems are now capable of performing large-vocabulary continuous speech recognition based on the assumption that sufficient training resources (e.g. hours of hand-transcribed speech data and large pronunciation dictionaries) exist in order to construct and train acoustic and language models. Even though they have been proven to be very accurate they have some notable drawbacks. Their performance solely depends on the quality and the quantity of the training data. Also, these data are annotated by human experts who have spent a huge amount of time in doing this process, which means that it is very expensive and time consuming. Finally, it has to be stated that for most of the spoken languages, preprocessed and annotated data resources are absent.

Our primary aim in conducting this project is to discover and extract terms (i.e. linguistic phrases or words) from input speech data in a completely unsupervised manner, zero-resource scenario [Jansen et al., 2010], which will give us an indication of how much can be learned from input speech data alone. This project is inspired by the work of [Saffran et al., 1996], on how 8-month-old infants start to identify and segment individual words. They found that infants are able to detect statistical patterns in syllables, indicating that the identification of recurring patterns may be important in the word acquisition process [Park and Glass, 2008].

This problem is known as motif discovery, where the task is to find approximately repeated subsequences in a longer time series data [Chiu et al., 2003]. The problem of motif discovery, comes from the computational biology area where they are interested in finding biologically significant recurring patterns, such as nucleotide sequences,
which they call *motifs*. In our case the time series data will be the continuous multivariate features (such as Mel Frequency Cepstral Coefficients (MFCCs) ) extracted from speech data. By discovering motifs, we potentially discover frequently occurring terms, because patterns in speech are more likely to be within phrases or words boundaries than across [Park and Glass, 2008].

The main goal of this project is to review and compare different methods that are used in discovering motifs in time series. Due to time constraint one method was implemented and evaluated. The method that we chose will extend the *SAX motif discovery* algorithm of [Chiu et al., 2003] in order to work with multivariate data (MFCC feature vectors), by applying SAX to each dimension independently and then concatenating the resulting string symbols of each dimension together. We improve this algorithm by introducing a two-phase neighbourhood range estimation method. When discovering motif seeds we apply a more strict neighbourhood range to require that the extracted subsequences are very similar to each other, and in the second phase we widen the neighbourhood to include motif occurrences that have a higher distance from the motif seeds.

The algorithm is very precise in discovering the motif occurrences, but it is not effective in detecting all possible motif occurrences (i.e. recall). This drawback is mainly due to the fixed subsequence length assumption, whereas the duration of spoken words, in our case digits, varies greatly even for same words, so when we apply the algorithm it is inevitable that many similar variable-length patterns will not be discovered.

1.2 Outline

The rest of this document is organized as follows: In Chapter 2 we provide the background information for this project, where we formally define the problem of motif discovery in time series data, then we briefly describe the Dynamic Time Warping (DTW) algorithm, and finally we briefly provide the related work for the motif discovery problem in many different research areas. Chapter 3 provides the experimental background needed for our work. In Chapter 4 we describe the discovery method that we implemented, which uses the SAX method combined with Random Projection algorithm and then we present the empirical results. In Chapter 5 we propose an extension to the current algorithm by applying a two phase neighbourhood range estimation. In Chapter 6 we suggest some directions for future work and finally, Chapter 7 summarizes the main contributions of this project.
Chapter 2

Background

2.1 Definitions and Notation

The following section is dedicated in understanding the motif discovery problem in continuous time series data. Thus, it will be dense in terminology and notations, which are necessary for having a more concrete definition of the problem.

In feature extraction phase, which will be briefly explained in Section 3.2, we convert the spoken utterances to times series of spectral vectors, which in our case are Mel Frequency Cepstral Coefficients (MFCCs). The length of the time series $T$ can be very long, e.g. recording a class lecture or concatenating utterances from different people. From these long time series our goal is to discover and extract recurring terms (i.e. linguistic phrases or words).

A subsequence $C$ of length $n$, is a subsection of contiguous positions from a time series $T$ of length $m$, that is, $C = t_{p},...,t_{p+n-1}$ for $1 \leq p \leq m - n + 1$ [Lin et al., 2002]. In order to discover all potential motif occurrences we have to extract all possible subsequences of a user defined length. We can achieve this by using a sliding window, where we create a matrix $S$ and subsequence $C_{p}$ will be placed in the $p^{th}$ row of this matrix. The matrix size will be $(m - n + 1) \times n$.

In order to determine the similarity of all pairs of subsequences we have to define a distance measure $D(C,M)$. The most common distance measures for continuous time series data are Dynamic Time Warping (DTW) (see Section 2.2) and the Euclidean distance (Equation 2.1).

$$D(C,M) = \sqrt{\sum_{i=1}^{n}(c_{i} - m_{i})^{2}}$$ (2.1)
We say that we have a \textit{match} between two subsequences $C$ and $M$, if $D(C,M) \leq R$, where $R$ is the neighbourhood range. If we set $R = 0$ then we search for exact matching, but this is unusual since it is very difficult to find exactly the same patterns in continuous time series data.

Figure 2.1: A visual intuition of a time series $T$ (light line), a subsequence $C$ (bold line) and a subsequence $M$ that is a match to $C$ (bold gray line indicates the $C$ subsequence) [Chiu et al., 2003].

If we have time series that change smoothly then we face the problem of a \textit{trivial match}, which means that the points exactly to the left or the right of the subsequence are the most similar, thus we should not include them as potential motifs. Figure 2.2 gives a visual intuition of a trivial match. A restrictive definition for excluding trivial matches is that potential motif occurrences should not temporally overlap each other [Minnen, 2008].

Figure 2.2: For almost any subsequence $C$ in continuous time series data, the best matches are the trivial subsequences immediately to the left and right of $C$ [Lin et al., 2002].

Finally, we should mention that the time series data should be Z-normalized (i.e.
have zero mean and unit variance), since there is no meaning to compare time series with different offsets and amplitudes, since even 'small changes in the offset rapidly dwarf the information about the shape of the two time series in question' [Keogh and Kasetty, 2003].

In this section we briefly mentioned the necessary background information for having a better understanding of the problem. A more detailed explanation of these definitions can be found in [Lin et al., 2002] and [Chiu et al., 2003].

### 2.2 Dynamic Time Warping

An well known issue in the area of speech recognition is the speaking rate variation between the speaker and utterances which causes timing differences between the speech patterns [Sakoe and Chiba, 1978]. For example, the utterance six, can be uttered by the same speaker in different speaking rates, e.g. s-ih-k-s or s-ih-k-s-s-s or s-ih-ih-k-s, and this fluctuation can be observed in TIDIGITS corpus [Leonard and Doddington, 1993] that we work, where the duration of the digit six, varies from 0.31 sec. to 0.64 sec..

Dynamic Time Warping (DTW) is an algorithm that tries to eliminate the timing differences between two spoken utterances, by warping the time axis in a non-linear fashion, in order to minimize the residual distance between the utterances. The minimization process is efficiently calculated using Dynamic Programming technique, where we efficiently solve the sub-problems and finally combine the solutions.

Assume that we have two spoken utterances (time series) $Q$ and $C$, where:

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n$$

(2.2)

$$C = c_1, c_2, \ldots, c_j, \ldots, c_m$$

(2.3)

We use DTW to align these time series by constructing an $n \times m$ matrix, where the $(i^{th}, j^{th})$ entry of the matrix contains the distance between the two points, e.g.

$$d(i, j) = || q_i - c_j ||$$

(2.4)

Our goal is to find a sequence of contiguous matrix elements that gives the best mapping from time axis of pattern $Q$ onto that of pattern $C$ [Sakoe and Chiba, 1978], which is called a warping path $W$ (Figure 2.3). The $k^{th}$ element of the warping path $W$ is defined as $w_k = (i, j)_k$. We define $W$ as:

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K, \quad \max(m, n) \leq K \leq m + n - 1$$

(2.5)
Since our goal is to find speech patterns, we have to apply some constraints, in order to preserve the linguistically structure of the utterances. These are:

1. **Boundary Conditions**: We require the warping path to start and finish in diagonally opposite corner entries of the matrix [Keogh and Ratanamahatana, 2004].

2. **Continuity**: The allowable steps in the warping path is restricted to adjacent cells.

3. **Monotonicity**: Is imposed to force the time ordering of the aligned points.
4. **Adjustment window condition**: We impose a global constraint to the admissible warping paths, by using the *Sakoe-Chiba* band which runs along the main diagonal (Figure 2.4). We use this constraint due to the fact that time fluctuations usually never cause an excessive timing difference [Sakoe and Chiba, 1978].

The window condition is:

\[ j - r \leq i \leq j + r, \quad r \geq 0 \text{ is the window length} \tag{2.6} \]

![Figure 2.4: Imposing a global constraint on the warping path using the Sakoe-Chiba band. The admissible warping paths are those that are within the grey area [Keogh and Ratanamahatana, 2004].](image)

To find the optimal warping path we use Dynamic Programming, which gives us an \( O(nm) \) time and space complexity algorithm for DTW. We compute each entry of the matrix using the following equation from [Sakoe and Chiba, 1978], where \( g(i, j) \) is the cumulative distance and \( d(i, j) \) is the distance found in the current cell.

\[
g(i, j) = \min \begin{cases} 
  g(i, j - 1) + d(i, j) \\
  g(i - 1, j - 1) + 2d(i, j) \\
  g(i - 1, j) + d(i, j)
\end{cases}
\tag{2.7}
\]
So, the optimal warping path, is the one that minimizes the following warping cost.

\[
DTW(Q, C) = \min \sqrt{\sum_{k=1}^{K} w_k}
\]

(2.8)

To deal with the multivariate time series data, such as our cepstral coefficients we use the Euclidean distance for all the dimensions \(S\), between each pair of vectors and store them to the distance matrix:

\[
d(i, j) = \sqrt{\sum_{s=1}^{S} (q_{is} - c_{js})^2}
\]

(2.9)

This review of DTW is very brief, a detailed one can be found in [Müller, 2007] Chapter 4 and [Sakoe and Chiba, 1978].

### 2.3 Related Work

In this section we will briefly survey the related work for finding repeated patterns in univariate and multivariate data. Initially, we will review the work that is done in the bioinformatics domain and then we will see how it relates to the general problem of pattern discovery in time series data which also includes speech data.

Researchers from many different areas such as artificial intelligence, data mining and bioinformatics have developed methods for the pattern discovery problem. In the bioinformatics community, the aim is to automatically discover biologically significant patterns, such as nucleotide sequences, which they call motifs. Due to the nature of the problem they have focused mainly on discrete univariate data.

Some motif discovery algorithms are MEME [Bailey and Elkan, 1994], which uses expectation maximization (EM) to fit a mixture model to the set of DNA or protein sequences, GEMODA [Jensen et al., 2006], which is a generic motif discovery algorithm that computes pair-wise distances between the subsequences and finally PROJECTION [Buhler and Tompa, 2002], which is a form of Locality Sensitive Hashing that performs a probabilistic dimensionality reduction. This algorithm solves efficiently the planted \((w-d)\)-motif problem, which was first proposed by [Pevzner and Sze, 2000].

The planted \((w-d)\)-motif problem is stated as follows: Suppose we are given \(t\) strings, each of length \(n\), where the symbols in the strings are i.i.d. (independent and identically distributed) generated. Also, we have a fixed but unknown motif \(y\) of length \(w\). If \(y\) occurs once in each string \(t\), but it is corrupted with exactly \(d\) substitutions, find the unknown motif \(y\).
We will use the *PROJECTION* algorithm in the method that we will re-implement, in order to efficiently discover motif seeds (*Section 4.1*).

[Chiu et al., 2003] introduced the concept of motif discovery on continuous univariate time series data. The brute force approach for finding recurring subsequences is quadratic in the length of the time series data, so is prohibitive for most of the real applications due to their size. Thus, [Chiu et al., 2003] proposed a discretization procedure of the time series combined with the *PROJECTION* algorithm, in order to reduce the comparison space. In the discretization step, they used Symbolic Aggregate approXimation (SAX) method [Lin et al., 2003], which is a quantization method that replaces each Piecewise Aggregation Approximation (PAA) segment [Keogh et al., 2001] with an equi-probable symbol. PAA approximates the normalized continuous time series data with a linear combination of box basis functions, by dividing the continuous data in equal-length segments and keeping the mean value of the data that fall on that segment.

The method of [Chiu et al., 2003] is very efficient since it discretizes the time series and significantly reduces the number of comparisons by applying the *PROJECTION* algorithm. This is very important when we have to deal with large amount of time series data. On the other hand, by modelling a large region of space with a single symbol we lose important information that may hurt the effectiveness of the discovery method. Thus we observe that there is a trade-off between efficiency and effectiveness. Also, as it will become apparent in *Chapter 4*, this method has lots of parameters that have to be tuned in order to find the optimal performance.

Later, extending the notion of motif discovery in multivariate times series data, [Tanaka et al., 2005] applied a univariate method very similar to SAX [Lin et al., 2003] by using Principal Component Analysis (PCA) in order to project the multivariate time series to a lower dimension and keeping only the first principal component which gives the highest variance and explains better the data. After converting the single dimensional continuous data to a sequence of symbols using SAX, they discover the motifs by using the Minimum Description Length (MDL) principle [Rissanen, 1989]. Although PCA is commonly used in many high dimensional applications, by keeping only the first principal component we lose sufficient information from the multivariate data.

[Minnen et al., 2007b] also extended the SAX motif discovery algorithm to work on multivariate time series data. They applied the SAX method independently on each dimension of the multivariate time series data and finally they concatenated the sym-
bols of each dimension. This approach is straightforward, but an issue with space requirements arises. In addition to working with multivariate data [Minnen et al., 2007b] also provided a method that automatically estimates the neighbourhood of each motif, resulting in improved performance of the algorithm compared to the standard SAX implementation of [Chiu et al., 2003]. Using a similar approach [Vahdatpour et al., 2009] presented a two phase algorithm for locating multivariate motifs in time series data, but their work mainly concentrated on extracting non-synchronous multivariate time series data (e.g. timing between moving right and left feet in walking activity slightly differs [Vahdatpour et al., 2009]). These methods worked well for data that came from different sensors and are promising to work well also for speech data where we have one multivariate source [Kapourani, 2013].

In the speech domain, [Park and Glass, 2008] attempted to find motifs directly on the acoustic data without using an intermediate symbolic representation. The technique that they used is a variant of Dynamic Time Warping (DTW) which they called Segmental DTW (S-DTW), between each pair of segments and they discover similar lexical entities by searching for low distortion regions in the audio data. Their technique has two parts, in the first part they find low distortion subsequence alignments using the S-DTW algorithm and in the second phase they cluster together these patterns by representing the audio stream as an abstract adjacency graph [Park and Glass, 2008].

The presented S-DTW algorithm, which is an $O(n^2)$ problem, is not scalable to many real applications for time series data. [Jansen et al., 2010] used the same technique as [Park and Glass, 2008], to efficiently solve the zero resource problem in the speech domain (without using any training data and dictionaries), by taking advantage of the representational and dotplot sparsity [Church and Helfman, 1993]. This approach resulted in speeding-up, in orders of magnitude, the process of the term discovery problem.

Finally, [Minnen et al., 2007a] proposed a method that operates directly on the continuous time series data to discover similar subsequences and avoid to discretize them as [Chiu et al., 2003] did. They formulated motif discovery as a problem of locating regions of high density in the space of all time series subsequences, where a high density region means that we have many motif occurrences in a small space. This idea is shown in Figure 2.5.

This method uses the approximate $k$-nearest neighbours to efficiently estimate the density around each subsequence and then selects the subsequences that are near the local density maxima. To detect all the occurrences of each motif and the right number
of motifs they adopted a *Greedy Mixture Learning* framework [Blekas et al., 2003]. What they actually did is to learn a Hidden Markov Model (HMM) from the selected subsequences that are near the local maxima, and then the motif models would compete to explain better the time series data.

For the purpose of this project, the method that we decided to re-implement is the algorithm of [Chiu et al., 2003] combined with the work of [Minnen et al., 2007b]. This choice was made mainly because of time constraint of this MSc project and we found that this method is feasible for the time required. Another reason, was the efficiency and the scalability of this method to big data sets compared to the other methods. Also, we tried to re-implement the density based method of [Minnen et al., 2007a], but due to limited time we could not manage to finish it and obtain some experimental results.
Chapter 3

Experimental Background

3.1 Speech Corpora

In this project, we will use the TIDIGITS corpus [Leonard and Doddington, 1993] to test and evaluate the performance of our motif discovery algorithms for speech data. TIDIGITS corpus consists of spoken digit sequences by over 300 men, women and children, each of them composing seventy seven (77) digit sequences. The data contain 11 classes which are from 'zero' to 'nine' plus 'oh' and each speaker provided in total 253 digits.

We evaluate our motif discovery algorithms in a small subset of the data containing only one speaker. This choice was made due to time requirements (the overall length of the time series data would be prohibitive for our algorithms) and because from only one speaker we can obtain all the information we need, since we will have repeated occurrences of the digits thus we will have motifs that are dense.

A problem that we face with the chosen corpus is that it is recorded in a controlled environment and we have a very small vocabulary. Even though, this may help us in having a better understanding on the experiments, it does not reflect realistic speech data where we have many different words and also some noise. We believe that in a different corpus that would contain more realistic speaker utterances, like SWITCHBOARD corpus [Godfrey et al., 1992], the pattern discovery algorithm would face some problems since the motifs would not be so dense (small number of occurrences in each different motif) and the number of the discovered motifs would be very big (a different motif for every different word).

We need to pre-process the dataset in order to treat the speech waveform as time series of acoustic feature vectors. The representation that we use is *Mel Frequency
**Chapter 3. Experimental Background**

Cepstral Coefficients (MFCCs). We extract the first 12 MFCC coefficients and one energy coefficient, using a 25 ms window length and 10 ms frame period.

### 3.2 Feature Extraction

In this section we will briefly explain the steps needed to transform the raw speech data to spectral vectors. A detailed explanation can be found in many speech and signal processing books, e.g. [Taylor, 2009]. Figure 3.1 shows the block diagram of the feature extraction process.

![Block diagram of the steps needed to transform the raw speech data to MFCC vectors](image)

**Figure 3.1:** Block diagram of the steps needed to transform the raw speech data to MFCC vectors [Renals, 2013].

1. Use a pre-emphasis filter to boost higher frequencies \((a = 0.97)\)

   \[
   x'[n] = x[n] - 0.97x[n-1]. \tag{3.1}
   \]

2. Multiply the full waveform by a tapered Hamming window \(w[n]\), in order to attenuate the discontinuities at the window edges. We have used a window size of 25\(ms\) and a frame shift of 10\(ms\), which are standard choices in speech recognition community.

3. Compute Discrete Fourier Transform (DFT), where we extract spectral information from the windowed signal.

4. Apply a Mel-scale filter bank to DFT power spectrum, then compute the log magnitude squared of each Mel filter bank output.
5. Obtain the Mel-frequency cepstral coefficients (MFCCs) by applying the inverse DFT (IDFT) to the log magnitude spectrum, which is equivalent to Discrete Cosine Transform.

The final feature vector has the first 12 MFCCs plus the energy coefficient, which leads to a 13-dimensional feature vector. For space concerns we did not compute the delta features, which would lead us to 39 dimensional feature vectors. For the feature extraction phase we used the HTK tool [Young et al., 2002].

### 3.3 Evaluation Method

In this project, we concentrate on evaluating the accuracy and not the run time characteristics of the pattern discovery algorithms, which is also a big issue for very long time series data. In the unsupervised setting that we work, we do not use any labeled data for training the algorithm, but we try to find interesting patterns directly in the raw input speech data.

Thus, we will adopt the approach that [Minnen, 2008] used in his thesis work. We measure the performance of the system by comparing the agreement between the discovered motifs and the expected motifs. The expected motifs are ground truth labels that identify which motifs exist and where they occur in the time series data [Minnen et al., 2007b]. These ground truth labels can be created by domain experts or automatically by doing time alignment and labeling the digit start point and its duration in seconds. In our work we used the second approach.

Even though, this approach is very similar to the supervised evaluation, it has some notable differences. First, the discovery algorithms try to find any patterns in the time series data, without knowing if they are important for the task that they are used for. In other words, the system may find patterns which are not expected, such as the silence between the spoken digits and utterances. So, the number of the discovered motifs may be different from the expected, but this does not mean that they are not valid.

Another issue that we have in the unsupervised setting is the assignment problem of the pattern labels. When we hear digits we recognise the digit 'one', 'two', etc., but the unsupervised system returns some motif labels with the occurrences in each motif. Thus, we have to map each motif label with the expected digit label. Of course, not all discovered motif labels have to map with the ground truth labels, since as we mentioned earlier the system may discover more motifs than those that we might have
expected.

This problem, which is an assignment problem, can be solved efficiently in $O(n^3)$ time using the Kuhn-Munkres algorithm, which is also known as the Hungarian algorithm [Kuhn, 1955] [Munkres, 1957]. The assignment problem can be stated as follows: Assume that we have $N$ workers and $M$ activities, our goal is to assign each worker to only one activity with the minimum cost, which for example could be his happiness in performing the assigned activity. In our problem, the expected labels are the workers and the discovered labels are the activities and we want to assign them to the most similar expected labels.

The final step is to define a similarity measure between the expected and the discovered motifs, which will act as the assignment cost. We adopt a straightforward and simple approach, where in each of the discovered motifs we count the number of matches with the expected patterns. The higher the count of matches the most similar is the discovered motif with the expected pattern. Thus, we only need to find the best assignment of the maximum match counts.

We consider a match if there is an overlap between the discovered and the expected motif occurrences [Minnen, 2008]. Adopting the notation from [Minnen, 2008], we define: $s = \text{time series index (name of the file)}, \alpha = \text{first frame of the motif occurrence}$, $\beta = \text{last frame}$ and $N = \text{length of the frame}$. We have a match between two occurrences $i$ and $j$, when the following criterion holds:

$$\left(s_i = s_j\right) \land \left(\alpha_i \leq \beta_j\right) \land \left(\alpha_j \leq \beta_i\right) \land \left(|\min(\beta_i, \beta_j) - \max(\alpha_i, \alpha_j)| \geq N / 2\right) \quad (3.2)$$

The first three parts ensure that there is an overlap between the two occurrences while the last part ensures that the amount of overlap is more than the half size of the occurrence length. The amount of overlap may vary depending on how strict or loose is the metric that we want to use.

We perform this procedure for all the discovered motifs, which finally gives us a matrix, with rows the expected motifs and columns the discovered motifs, and each entry contains the count of matches as described above. This matrix is given as entry to the Hungarian algorithm, which returns the best assignment.
3.4 Evaluation Metrics

In our experiments we will use *accuracy* and *F-measure* to evaluate the performance of the discovery algorithms. Accuracy measures how well the detected labels from our algorithms match the ground truth label occurrences. Adopting the notation from [Minnen, 2008], equation for accuracy is given by:

\[
\text{accuracy} = \frac{C - I}{N} \tag{3.3}
\]

In Equation 3.3, we have the following statistics: \(C\) is the number of *true positives*, which is the number of times that our algorithm detected an occurrence and this detection was the expected one. \(I\) is the number of *false positives*, which is the number of times that our algorithm detected an occurrence and we did not expected to have one in that position. Finally, \(N\) is the total number of expected motif occurrences, which can be expressed as \(N = C + D + S\), where \(D\) is the number of deletion errors, the number of times that there was a motif occurrence but our algorithm did not detect it, and \(S\) is the number of substitution errors, which is the number of times our algorithm detected a motif occurrence, but it assigned it to a different motif class.

Another statistic is the \(E\) statistic, which we define it as the number of extra detections from extra motif classes. As we described in the previous section, our algorithm may detect additional pattern labels that do not correspond to the expected motif labels. The occurrences in these additional patterns increase \(E\). The \(E\) statistic could be considered as *false positives*, but as we mentioned earlier these additional patterns may be valid (e.g. one possible pattern may represent the silence between two different digits or utterances), thus we decided to ignore these patterns and not use them when evaluating the performance of algorithms.

F-measure, is based on the *precision* and *recall* measures, and summarizes the performance of the system in a single number from zero (worst performance) to one (best performance). Recall is the fraction of the true occurrences that were detected, in other words how many true occurrences did we miss.

\[
\text{Recall} = \frac{C}{N} \tag{3.4}
\]

Precision is the proportion of the detected occurrences that were in fact correct, which means how certain are we that the detected motif occurrences are indeed correct.

\[
\text{Precision} = \frac{C}{M} \tag{3.5}
\]
where \( M = ( \text{number of all the detected occurrences} ) - ( \text{number of extra detections from the additional motifs (E)}) \).

So F-measure is defined as the \textit{weighted harmonic mean} of precision \((P)\) and recall \((R)\), which is given by \textit{Equation 3.6}:

\[
F - \text{measure} = \frac{(\beta^2 + 1) * P * R}{\beta^2 * P + R} \tag{3.6}
\]

In our experiments we will use \( \beta = 1 \), which means that we give equal importance to precision and recall, this is called the \( F_1 \)-\textit{measure}. Values of \( \beta > 1 \) emphasize recall. So, the \( F_1 \)-\textit{measure} is:

\[
F_1 - \text{measure} = \frac{2 * P * R}{P + R} \tag{3.7}
\]
Chapter 4

Motif Discovery Algorithm

The main goal of this project is to review algorithms that are used in the motif discovery problem. Thus we will try to re-implement one method that was described in the Related Work section and adjust it to multivariate speech data.

4.1 Motif Discovery via SAX and Random Projection

This approach is based on [Chiu et al., 2003] probabilistic discovery method of time series motifs. It is a local discretization method that converts the real-valued subsequences into a single string. By using Random Projection we reduce computational complexity for the motif discovery process. We will extend this method to work for multivariate time series data in a similar way as [Minnen et al., 2007b] did. The steps of the algorithm are shown below:

1. Extract all possible subsequences of length $n$ using a sliding window.
2. Using SAX convert each subsequence to a word of length $w$ and alphabet size $a$.
3. Apply $m$ times the Random Projection algorithm and build a collision matrix.
4. Select the largest entry in the collision matrix which corresponds to a pair of subsequences and treat them as motif seeds.
5. Iterate through all subsequences and add them as motif occurrences if they are less than a fixed distance from the motif seeds.
4.1.1 Dimensionality Reduction and Subsequence Discretization

The extraction of all possible subsequences using a sliding window is straightforward, thus in this section we will explain the second step of the algorithm. To reduce the dimensionality of the subsequences from \( n \) to \( w \) dimensions we apply Piecewise Aggregate Approximation (PAA) [Keogh et al., 2001]. PAA approximates the normalized continuous time series with a linear combination of box basis functions as shown in Figure 4.1. First we divide the \( n \)-dimensional data to \( w \) equal length segments and then we keep the mean value of data falling in each segment to create a \( w \)-dimensional vector \( \bar{C} \). Vector \( \bar{C} \) is the dimensionality reduced representation where each element is calculated using Equation 4.1.

\[
\bar{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j
\]  

Figure 4.1: PAA representation approximates the normalized continuous time series with a linear combination of box basis functions. The time series of length 128 is reduced to 16.

Having reduced the dimensionality of the time series we can apply another transformation to obtain a discrete representation instead of real-valued numbers that we have in PAA representation. We can achieve this by applying SAX [Lin et al., 2003], which is a quantization method that replaces each PAA segment with an equi-probable symbol [Lin et al., 2002].
To produce these equi-probable symbols we use the fact that normalized time series data are highly Gaussian. Given the alphabet size $a$ we have to determine the breakpoints $\beta_i$ that produce equi-probable regions under the standard normal distribution $N(0, 1)$, that is, the area from $\beta_i$ to $\beta_{i+1} = 1/a$. We can determine the breakpoints by looking them up in a statistical table.

Finally we have to map each PAA segment with the corresponding symbol. We can achieve this by mapping all values that fall below the smallest breakpoint to symbol ‘a’, all values that are greater than the smallest breakpoint and smaller than the second breakpoint to symbol ‘b’ and so on. So for each subsequence we will have a string, also called a word, that is the concatenation of these symbols, whose length will be $w$. This procedure is shown in Figure 4.2.

![Figure 4.2: A continuous time series converted into a SAX word 'bababdddc' using eight PAA segments and four SAX symbols as alphabet size.](image)

Each of these SAX words is stored as a row in the matrix $S$ whose size is $(m - n + 1) \times w$, where $m$ is the total length of the time series, $n$ is the length of the subsequence and $w$ is the length of the SAX word.

We observe that SAX is only applicable for univariate time series data, whereas the speech data are multivariate (13-dimensional feature vector). To solve this issue we apply the SAX method independently to each dimension of the MFCC vectors and finally we concatenate the words of each dimension [Minnen et al., 2007b]. So the matrix $S$ will have dimensions $(m - n + 1) \times w \times 13$. Even though this technique is straightfor-
ward, an issue with space requirements arises, since we have a linear increase in the word length at each row of the matrix. This makes the discovery process slower since we have a bigger amount of data that need to be compared, but it is inevitable since we work with multivariate data and we need to keep as much information we can.

4.1.2 Random Projection

After the transformation of all subsequences to SAX words we can use the Random Projection algorithm [Buhler and Tompa, 2002] to efficiently estimate the similarity between each pair of words. The intuition behind Random Projection is simple, but yet is very powerful for many problems that have time scalability issues. While the brute force approach for direct comparisons of the strings is infeasible since we have $O(N^2)$ pairs of strings, using $I$ iterations of the Random Projection we can identify similar subsequences in $O(I * N)$ time.

We begin by randomly choosing a subset of the columns of $S$ to act as a mask. Then we build a hash table based on the mask of each word. If the masks of two words are the same, then they fall on a common bucket. For all combinations of the words in each bucket we increase the count of the appropriate cell in the collision matrix, which corresponds to their position in matrix $S$. The collision between the masks implies similarity between the words, but they may also be false alarms. To overcome this problem we repeat the process $I$ times, each time using a new randomly chosen mask. This procedure hopefully will give us a collision matrix that has high values for the most similar subsequences. Figure 4.3 shows this process.

The size of the mask (or the random projection size) is important for the algorithm. According to [Buhler and Tompa, 2002], the size of the mask $k$ should be chosen so as to ’minimize contamination of the planted bucket by random background sequences’. Following the analysis of [Buhler and Tompa, 2002], for a particular mask length $k$, the lower bounding probability that two motif occurrences hash in the same bucket is:

$$\hat{p}(w, d, k) = \frac{(w-d)}{k} \binom{w}{k} \quad (4.2)$$

In Equation 4.2, $w$ is the length of the concatenated SAX words and $d$ is the number of the possible errors (or substitutions). We can observe that the probability $\hat{p}(w, d, k)$ increases as we decrease the projection size $k$, which means that is more likely two random motifs to hash to the same bucket. As we increase $k$, becomes unlikely that
Figure 4.3: A single iteration of the Random Projection algorithm. (a) A subset of string symbols are selected (positions one and three). (b,c) The symbols are hashed and (d) collisions are recorded in the collision matrix by incrementing the count in the appropriate cell [Minnen et al., 2007b].

random motif occurrences hash to the same bucket if they are not really similar. Also, it should be mentioned that a big number of iterations \( I \) and a small projection size \( k \), increases the number of false positive pairs, which is the number of dissimilar subsequences that hash to the same bucket. Even though, this is a problem for the efficiency of Random Projection, since it will return more potential motif occurrences to compare in later step, does not hurt so much the effectiveness of the algorithm, since in the motif selection phase we will compute the actual distance, and if it is less than a threshold we will finally consider them as motif seeds.

According to [Buhler and Tompa, 2002] the projection size parameter should be chosen such that:

\[
\begin{align*}
k &< w - d \\
\end{align*}
\]  \hspace{2cm} (4.3)

\[
\begin{align*}
k &\geq \log \left( \frac{t(n-w+1)}{a} \right) \\
\end{align*}
\]  \hspace{2cm} (4.4)

In Equations 4.3 and 4.4, \( w \) and \( d \) are the same as mentioned above, \( n \) is the total length of the time series data, \( t \) is the number of different time series (in our case 1), and \( a \) is the SAX alphabet size.
Equation 4.4 purpose is to filter out the noise, and return only the most similar subsequences for actual comparison. In our experiments, we set the projection size parameter in a small range around this threshold to observe its performance, in the cost of being less efficient when we set it smaller, because of the higher number subsequences that had to be compared. Values much higher than this threshold did not return at all any potential motifs, thus we decided to not run experiments on those values.

It should also be mentioned that the Random Projection algorithm requires that we have equiprobable symbols in the alphabet $a$. This is guaranteed by first applying the SAX method which produces equiprobable regions and then assigning to each region a symbol. Finally, these equiprobable symbols are used in the Random Projection to find the most similar subsequences.

In addition to being an efficient method, Random Projection can deal with the problem of a small amount of noise in the data. For example, imagine that two similar subsequences are only different in a very small fraction of their positions. In some of the $I$ iterations of the algorithm the mask of the random projections will not include these positions, so the two subsequences will appear to be similar when we will examine the collision matrix. Finally, we should also mention that the majority of the entries in the collision matrix will be zero. Thus, in order to save space complexity (a full matrix requires $O(N^2)$ space), we make use of a sparse matrix implementation.

### 4.1.3 Discovering Motif Occurrences

After creating the collision matrix we are ready for the motif discovery phase. A large value in the collision matrix implies a similarity between the pair of subsequences which eventually may not correspond to a motif, but it is a strong indicator [Chiu et al., 2003].

We begin by retrieving the pair of subsequences that have the highest value in the collision matrix. This pair of subsequences, if it satisfies some criteria which are listed below, can be selected as motif seeds. Having this pair of seeds we can discover all possible subsequences that are within distance $R$ from the pair of motif seeds by scanning all the time series data (i.e. $\min(dist(w_i, c_1), dist(w_i, c_2)) \leq R$, where $w_i$ represents all the possible subsequences and $R$ is a user defined neighbourhood radius).

We continue this procedure for the other maximal entries of the collision matrix, until the cell value in the collision matrix is below a user defined threshold that allows only the matrix entries whose score is greater than we would have expected by pure chance.
Chapter 4. Motif Discovery Algorithm

[Chiu et al., 2003].

The criteria that the subsequences must satisfy in order to be considered as motif seeds are three:

1. The distance between the pair of seeds should not be more than $R$. The distance can be measured using the Euclidean distance or DTW.

2. The seeds should not overlap each other (*trivial match*).

3. The seeds should not overlap previously identified occurrences in other motif seeds and they should satisfy $D(C_K, C_i) > const * R \forall 1 \leq i < K$, where $K$ is the number of motifs, and $1 < const < 2$, as shown in Figure 4.4 [Lin et al., 2002].

![Diagram of motif discovery criteria](image)

Figure 4.4: A visual explanation of why each motif should be $const * R$ apart. If the motifs are only required to be $R$ distance apart as in A, then the two motifs may share the majority of their elements. In contrast, B illustrates that requiring the centres to be $const * R$ apart insures that the motifs are unique [Lin et al., 2002].

Equation 4.5 [Chiu et al., 2003] below, gives us the average hit in each entry of the collision matrix in each iteration, where $s$ is the number of the SAX words of length $w$, $\alpha$ is the size of the alphabet, $d$ is the number of errors and $k$ is the projection size.

$$E(s, \alpha, w, d, k) = \left( \frac{s}{2} \right) \sum_{i=0}^{d} \left( 1 - \frac{i}{w} \right)^k \binom{w}{i} \left( \frac{\alpha - 1}{\alpha} \right)^i \left( \frac{1}{\alpha} \right)^{w-i} \quad (4.5)$$

Equation 4.5 consists of two parts, where the first part (Equation 4.6) gives the probability that two random SAX words match with up to $d$ errors, and the second
part (*Equation 4.7*) gives the probability that two SAX words project to the same value (Locality Sensitive Hashing).

$$p_1(a, w, d) = \sum_{i=0}^{d} \binom{w}{i} \left( \frac{\alpha - 1}{\alpha} \right)^{i} \left( \frac{1}{\alpha} \right)^{w-i}$$  \hspace{1cm} (4.6)

$$p_2(w, d, k) = \sum_{i=0}^{d} \left( 1 - \frac{i}{w} \right)^{k}$$  \hspace{1cm} (4.7)

What we expect from the collision matrix is to be very sparse (i.e. containing many zeros), and only contain some high values in pairs of subsequences that are very similar to each other. This expectation means that the value of *Equation 4.5* is going to be quite small, since it will be dominated by the zeros of the collision matrix. Indeed, this intuition is true since we ran the Random Projection algorithm with 250 iterations and we kept the frequencies of occurrences in the collision matrix.

![Figure 4.5](image)

**Figure 4.5:** This graph shows the frequency of different entries in the collision matrix, where the frequencies are in log scale. For example, we saw about 22,250,000 times number 0 (zero occurrences) and only 7000 times the number 2 (two occurrences) in the collision matrix. We observe that the distribution of occurrences in the collision matrix follows the power law distribution.
Figure 4.5 shows the results for the frequency (in log scale) of each different entry in the collision matrix. We observe that after number 60 the occurrences vanish, but we have a spike for the number 70, this is mainly due to the occurrences of trivial matches between the subsequences.

For this reason, in the experiments that we conducted we used a higher threshold that would keep only the most similar subsequences, but would not discard some potential motifs (false negatives). Thus, our goal was first to set the threshold parameter in order to decrease the rate of false negatives, and then to concentrate on the efficiency of algorithm by keeping, and hence comparing, only the most similar subsequences. In the next section we describe the combinations of parameters that we used to run our experiments.

4.2 Performance and Analysis

The data that we are going to evaluate our algorithm is the TIDIGITS corpus, which consists of spoken digits from 'zero' to 'nine' plus 'oh'. For quantitative evaluation of the discovered motifs we will use the accuracy and F-measure metrics as they were described in Section 3.4.

A main issue with our motif discovery algorithm is the big number of user specified parameters that have to be tweaked in order to find the optimal performance. Some of the parameters do not make a big difference on the performance of the algorithm, while others have a great impact even by slightly changing their value (e.g. the parameter for the neighbourhood range). Table 4.1 shows all the user specified parameters.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>subsequence (window) length</td>
</tr>
<tr>
<td>w</td>
<td>number of PAA segments (SAX word length)</td>
</tr>
<tr>
<td>a</td>
<td>number of discrete symbols (SAX alphabet size)</td>
</tr>
<tr>
<td>I</td>
<td>number of iterations of the Random Projection algorithm</td>
</tr>
<tr>
<td>k</td>
<td>projection size for the RP algorithm</td>
</tr>
<tr>
<td>t</td>
<td>threshold of the collision matrix</td>
</tr>
<tr>
<td>R</td>
<td>range of the neighbourhood</td>
</tr>
<tr>
<td>const</td>
<td>to ensure no overlap between the discovered motifs</td>
</tr>
</tbody>
</table>
Another issue that we have to deal with is the requirement of a fixed subsequence length. This is a well understood problem in pattern discovery area and becomes apparent in the speech domain that our project is focused. Spoken words, digits in our case, differ greatly in their duration (length) and even same words have huge variation in length, as shown in Figure 4.6. As it is easily understood this huge length variation puts a barrier in the effectiveness of the algorithm, which works only for fixed-length subsequences.

Figure 4.6: Two utterances from the TIDIGITS corpus by showing the 13 MFCC vectors. The one on the top is utterance ‘75443z7’ and the other is ‘8o45647’. The highlighted regions are the occurrences of the digit ‘4’ and show its length variation. In this example the smallest duration is 22ms and the highest 51ms.

4.2.1 Results

In the experiments that we conducted, we tried to make as many as possible combinations of the parameters that we described in the previous section. Due to time constraint, we used representative parameter values and we did not investigate them thoroughly, thus the results that we are going to show may not be optimal.

In all the experimental results we used the DTW distance measure to compute the actual distance between the pairs of subsequences. More specifically we used the constrained DTW with Sakoe-Chiba band of 10%, which is a standard practice in data mining and speech recognition community [Ratanamahatana and Keogh, 2005].

When applying PAA we require that the subsequence is segmented in \( w \) equal length segments. For this reason we chose the pair of parameters so as the subsequence...
length to be exact multiple of $w$ as is shown in Table 4.2. We tried to choose a high value for $w$, in order to lose as less information as we could when we discretize the time series data.

<table>
<thead>
<tr>
<th>Subsequence length</th>
<th>20</th>
<th>24</th>
<th>25</th>
<th>28</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAA segments</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

In Figure 4.7 we give an example output of the motif discovery algorithm. The utterance in the example is "three seven one three four six six". The ground truth labels (i.e. starting and ending positions of the digits from the automatic alignment) are shown in the top bars below the plot of the MFCCs. The bottom bars show the discovered motif occurrences. We observe that the algorithm discovered five out of seven digits and also it found an extra motif occurrence at the end of the second "six" digit. We also notice that the discovered motif occurrences are quite accurate and they are shifted only a small fraction on the left or on the right of the ground truth labels. Finally, the variable-length of the ground truth labels is very clear especially in the last "six" digit, whereas our motif occurrences have a fixed-length.

Figure 4.7: Above, MFCCs for the utterance "three seven one three four six six". Below two levels of bars, the top has the ground truth labels, whereas the bottom has the discovered motif occurrences. The discovery algorithm failed to detect the digits "one" and "four", and it discovered one extra occurrence that was not expected. We should notice the difference in the length of the occurrences, in our algorithm we have fixed-length motif occurrences, whereas the ground truth labels are variable-length.
4.2.1.1 Optimal Parameter Setting

The optimal parameter setting that we found after experimenting with different parameter values, had an overall accuracy of 54.72% and $F_1$-measure 70.73%. Table 4.3 shows the best performance using the optimal combination of parameter values which is given in Table 4.4. The number of discovered motifs is 31 which is really high compared to 11 true motifs that we expected. The total number of correct detections (i.e. true positives) is 58 out of 106 (total number of spoken digits in our dataset). Also, there were neither substitution nor insertion errors (i.e. false positives).

Table 4.3: Best performance with the optimal combination of parameter values.

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>54.72%</td>
</tr>
<tr>
<td>$F_1$-measure</td>
<td>70.73%</td>
</tr>
<tr>
<td>Recall</td>
<td>54.72%</td>
</tr>
<tr>
<td>Precision</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 4.4: Optimal combination of parameter values.

<table>
<thead>
<tr>
<th>N</th>
<th>w</th>
<th>a</th>
<th>I</th>
<th>k</th>
<th>t</th>
<th>R</th>
<th>const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>20</td>
<td>5</td>
<td>5</td>
<td>250</td>
<td>7</td>
<td>5</td>
<td>10.5</td>
</tr>
</tbody>
</table>

From the different evaluation metrics we can observe that the problem of the discovery algorithm is that it cannot capture all the occurrences of the different motifs (i.e. recall, which is very similar to the accuracy metric), but it is very precise in the motif occurrences that it discovers. The same behaviour is observed also for other parameter combinations, which shows that indeed the algorithm can capture the interesting patterns of the data, but the length variation of speech patterns puts a barrier in its performance.

To our knowledge, only [Minnen, 2008] in his PhD thesis has also run experimented with the TIDIGITS corpus, testing the performance of the same discovery algorithm that we re-implemented. The best performance that he found had 71.5% accuracy, but he did not provide any results about the F-measure metric. We believe that we cannot directly compare our results with the ones that were found in his thesis,
since he mentions that he used a small part of the corpus but not exactly which. Also, he does not mention the optimal parameter setting for obtaining this performance.

In our results, the high number of discovered motifs, can be explained in many different ways. First of all, the small subsequence length, 20 ms, results in discovery of patterns that were not expected. For example, some digits have duration which varies from 30 ms to 60 ms, which means that a subsequence of length 20 ms can separate the digit in 2 or 3 segments, or discover patterns between different digits.

Another reason is the procedure for discovering motif seeds. When we apply SAX we discretize the subsequences, but this way we lose important information, since the continuous subsequences are replaced with a small number of symbols. Thus, when we use Random Projection, some differences in the raw subsequences may not be observed when we have the SAX words, and the RP algorithm falsely considers them as similar. Also, as was described in the previous section, by applying the RP algorithm many times we increase the number of false positives and thus if the threshold is not too high, we consider the pair in the entry of the collision matrix as potential motifs.

Finally, in the last phase of the discovery algorithm, we compare the actual distance between the potential motif seeds. In this phase, parameters $R$ and $\text{const}$ are crucial and strongly connected to each other. The neighbourhood radius parameter should be big enough to include motif occurrences but it should be small enough to include only the most similar subsequences. The $\text{const}$ parameter ideally should be big enough to ensure that there is no overlap between the discovered motifs. But this is really difficult, since the speech patterns, represented by the 13 MFCCs, are not so different (i.e. the difference depicted by the DTW distance measure). This means that if we set a high $\text{const}$ parameter, then the number of the discovered motifs would reduce drastically, even to one motif. The connection between these two parameters is that when we set the $R$ parameter small enough, then the $\text{const}$ parameter can be set higher, and the opposite when we set a bigger $R$ parameter.

### 4.2.1.2 Different Subsequence Length

As we already mentioned each different digit, and even same digits, have different duration times. After observing the length of each digit occurrence in the dataset we concluded in using five different subsequence lengths and evaluate in each of them the performance of the discovery algorithm. Table 4.2 shows the subsequence lengths that we decided to choose. In Figure 4.8 we see the performance of the different subsequence lengths, where for each different subsequence length we plot accuracy,
F-measure, recall and precision. From the graph we notice that as we increase the subsequence length the precision decreases and consequently F-measure.

![Performance graph](image)

Figure 4.8: Performance of the different subsequence lengths (20, 24, 25, 28, 30). We should note that the graph starts at 40% performance.

<table>
<thead>
<tr>
<th>Subsequence Length</th>
<th>20</th>
<th>24</th>
<th>25</th>
<th>28</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td># Motifs</td>
<td>31</td>
<td>21</td>
<td>23</td>
<td>17</td>
<td>22</td>
</tr>
<tr>
<td># Correct</td>
<td>58</td>
<td>56</td>
<td>56</td>
<td>59</td>
<td>50</td>
</tr>
<tr>
<td># Insertions</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td># Substitutions</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>k</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>t</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>R</td>
<td>10.5</td>
<td>11.5</td>
<td>11.75</td>
<td>13</td>
<td>13.25</td>
</tr>
<tr>
<td>const</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Table 4.5: Results and parameter values for different subsequence lengths.

Table 4.5 gives us more details about the performance and the parameter settings. We should mention that throughout all the experiments, alphabet size $a$ was set to 5, and the number of iterations $I$ for Random Projection was set to 250.

Something worth mentioning from Table 4.5, is the insertion errors for subsequences of length 28. Even though the correct occurrences that were discovered is
the highest, 59 occurrences, accuracy is not the best because of insertion errors. On
the other hand the recall metric is the highest that we found 55.66%, since it does not
depend on insertion errors.

4.2.1.3 Projection Size and Threshold

In this section we will initially study what is the impact of the projection size $k$ in the
overall performance and the discovery of the motifs and then we will study what effect
has the threshold parameter $t$ in our algorithm.

Table 4.7 shows the performance of the algorithm for different values of the pro-
jection size $k$, by keeping all the other parameters fixed. The values of the other pa-
rameters are set as shown in Table 4.6.

Table 4.6: Fixed parameter values.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>w</th>
<th>t</th>
<th>R</th>
<th>const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>24</td>
<td>6</td>
<td>7</td>
<td>12.25</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Table 4.7: Different values for projection size $k$. We notice that as we increase $k$, the
number of discovered motifs decreases.

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>52.83</td>
<td>46.23</td>
<td>42.45</td>
<td>43.4</td>
</tr>
<tr>
<td>F-measure (%)</td>
<td>61.88</td>
<td>55.14</td>
<td>51.65</td>
<td>53.11</td>
</tr>
<tr>
<td># Motifs</td>
<td>21</td>
<td>13</td>
<td>12</td>
<td>10</td>
</tr>
</tbody>
</table>

As we expected, increasing the value of $k$ decreases the number of the discovered
motifs, since there will be less SAX words falling on the same bucket, thus the values
in the collision matrix will be smaller. Also, we observe that we have the best perfor-
ance for $k = 5$, but this result is mostly due to the choice of the combination of the
parameters and not so much because of the projection size parameter.

The next parameter that we will observe is the threshold parameter $t$. The other
parameters were kept fixed and we used the same values as in Table 4.6 and we set
$k = 5$. As we observe in Table 4.8, the threshold parameter does not have such a big
impact as the projection size, but it has the same effect. As we increase the threshold $t$
the number of discovered motifs decreases.
Table 4.8: *Different values for threshold parameter t. We observe that as we increase t, the number of discovered motifs decreases.*

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>52.83</td>
<td>52.83</td>
<td>52.83</td>
<td>47.17</td>
</tr>
<tr>
<td>F-measure (%)</td>
<td>61.88</td>
<td>61.88</td>
<td>61.54</td>
<td>55.56</td>
</tr>
<tr>
<td># Motifs</td>
<td>21</td>
<td>21</td>
<td>20</td>
<td>18</td>
</tr>
</tbody>
</table>

4.2.1.4 Different Neighbourhood Range

In this section we will investigate the neighbourhood range parameter $R$ and the $\text{const}$ parameter which ensures that there is no overlap between the different motifs.

*Figure 4.9* shows the performance of the algorithm over different values of the neighbourhood range parameter. In the horizontal axis we have the two different evaluation metrics, *accuracy* and *F-measure* for all the different values of $R$ ranging from 10.25 to 12.5. We observe that the $R$ parameter has a great effect on the performance of the algorithm since for a small change of the value from 12.5 to 12.25, F-measure increases from 41.41% to 61.88%. This high instability of the algorithm is because of the single neighbourhood size for all the different motifs, where some of them may actually have a bigger or a smaller neighbourhood range.

*Figure 4.9:* *Performance of the discovery algorithm, accuracy and F-measure, for different values of neighbourhood range $R$. We clearly observe the instability of the algorithm for small changes in the value of $R.*
Finally, we examine the behaviour of the algorithm for different values of the const parameter. The purpose of the const parameter is to allow only very different pair of subsequences to be considered as motif seeds. So, if two subsequences are very similar to each other, but also they are similar to a previously identified pair of subsequences then we shall not consider them as different motif seeds. Table 4.9 shows the performance of the algorithm for different values of this parameter and the one that we are interested in is how the number of discovered motifs changes as we increase the value of const. The fixed parameters are the same as shown in Table 4.6. For const = 1.3 we discover only one motif which means that there were no motif seeds that had a distance higher that $1.3 \times R$, where $R = 12.25$, from the first discovered motif seeds.

Table 4.9: Different values for the const parameter. We observe that a small increase in the value has a great effect on the number of discovered motifs and consequently on the performance of the algorithm.

<table>
<thead>
<tr>
<th></th>
<th>1.1</th>
<th>1.15</th>
<th>1.2</th>
<th>1.25</th>
<th>1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>52.83</td>
<td>24.53</td>
<td>16.04</td>
<td>5.66</td>
<td>5.66</td>
</tr>
<tr>
<td>F-measure (%)</td>
<td>61.88</td>
<td>33.77</td>
<td>26.56</td>
<td>10.17</td>
<td>10.53</td>
</tr>
<tr>
<td># Motifs</td>
<td>21</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

### 4.2.2 DTW vs Euclidean Distance

In the Background chapter, we mentioned that the DTW distance is mainly used in the speech recognition community due to its ability to warp the time axis in a non-linear fashion and thus eliminate the timing differences between two spoken utterances. But, does this time warping help our algorithm to return better results and improve its performance against the widely used Euclidean distance? The answer is yes, and it is shown in Figure 4.10, which compares the two distance metrics for different subsequence lengths.

We observe that accuracy for DTW is about 5% higher in general than the Euclidean, except from subsequences of length 20 that we have almost the same performance. We should note that the choice of the parameter values is optimal for a fixed projection size $k = 5$, since there was not enough time to run experiments with all possible values of $k$ also for the Euclidean distance.

When running experiments using the Euclidean distance, accuracy decreases as we
Figure 4.10: Performance of the discovery algorithm for DTW and Euclidean distance, red and blue colour respectively. As we observe accuracy is higher when we use the DTW distance to discover motif occurrences. We should note that the graph stops at 60% accuracy.

increase the length of the subsequences. This probably happens because Euclidean distance does pointwise comparisons between the two subsequences and thus if one of them is slightly shifted the distance measure will increase falsely, and the longer is the subsequence length the bigger will be the accumulated distance.

We should notice that the way we use the DTW in our experiments is not the optimal since we compare two equal length subsequences. But, when two same digits have time variations when they uttered they will also have different length. Unfortunately, the method that we have implemented limits us to compare only fixed and equal length subsequences.

From the experimental results we observe that the algorithm works quiet well for the discovery process, since it is very precise in the motifs that it discovers, but it is weak in discovering all the possible motif occurrences mainly due to variable-length of the speech patterns. Also, it is very unstable to the combination of the parameter values and especially for the neighbourhood range $R$. In the next chapter we propose an extension to the current algorithm to consider a two phase neighbourhood range.
Chapter 5

Two Phase Neighbourhood Range

In this chapter we will extend the motif discovery algorithm that we described earlier, by applying a two phase neighbourhood range estimation. As was observed in the empirical results the neighbourhood range parameter $R$ is vital for the performance of the algorithm and the discovery of the correct motifs. We should mention that this parameter is highly dependent on the time series data characteristics and the distance metric used for measuring the similarity between the pair of subsequences.

The idea behind this method comes from a very simple observation. In steps 4 and 5 of the discovery algorithm as presented in Section 4.1, we first discover the motif seeds and then depending on those motif seeds we scan all the time series to discover additional motif occurrences. A natural hypothesis is that we want to find motif seeds that are the nearest (i.e. most similar) among the other occurrences in the discovered motif, and then using these almost identical seeds we can perform step 5 of the algorithm, where we should not be so strict about the similarity of the other occurrences that are going to be included in each motif.

Thus, in the motif seed selection we can apply a smaller neighbourhood range $R$, and then for the discovery of the other motif occurrences we can include subsequences that belong to a wider neighbourhood. We should be careful though not to make a very small initial neighbourhood range $R$, since we may not discover any motif seeds.

Except from higher performance, we expect that this method can improve the discovery algorithm in different ways. First of all, by applying a stricter neighbourhood range we expect that the number of discovered motifs will be smaller, since even though many subsequence pairs will have their entry in the collision matrix higher than the predefined threshold, in the calculation of their actual distance, using DTW or Euclidean, they may be discarded due to the strict neighbourhood range. Also, as
a consequence of the decrease in the number of discovered motifs, the algorithm will be more efficient, since when discovering the other motif occurrences we will have to make less iterations and scan the whole time series data.

Finally we should mention that the range of the neighbourhood is not used during the discretization step and random projection, thus this part of the algorithm will remain the same.

[Minnen et al., 2007b] also noticed the importance of the neighbourhood range in the discovery process. In their work, they adopt a more complicated technique than ours, where they tried to automatically estimate the neighbourhood of each different motif. Thus each different motif can have a different range \( R \), as shown in Figure 5.1.

![Figure 5.1](image)

**Figure 5.1**: *On the left a single neighbourhood range may not capture ideally the different motifs. When the range is allowed to vary, as shown on the right, then each motif can adapt and ideally include all the valid occurrences from the time series data [Minnen, 2008].*

Even though their method works well in many domains, in the speech domain, and particularly with the TIDIGITS corpus, the performance of the extended algorithm was slightly worse than the standard algorithm of [Chiu et al., 2003]. They explain that this poor performance on the speech data is due to the difficulty of estimating the neighbourhood range, since the neighbourhood estimation procedure assumes that there are fairly well-isolated motifs [Minnen, 2008], which means that there is a big difference in the distance of occurrences that are included in a motif and those that are outside. As was explained in the previous section, in the speech data that we work, we do not have so big differences between different motif occurrences.
5.1 Performance and Analysis

In this section we will show the experimental results for the two phase neighbourhood range method and we will compare the results with the standard algorithm of [Chiu et al., 2003] that we implemented in the previous chapter.

5.1.1 Results

The optimal parameter setting, for the different combination of parameter values that we used, had an overall accuracy of 58.49% and $F_1$-measure 70.06%. This is the highest accuracy that we found in any of our experiments, showing that the two-phase neighbourhood range has an effect on the performance of the algorithm. In Table 5.1 we show all the evaluation metrics for the optimal combination of parameter values, which are given in Table 5.2.

Table 5.1: Best performance with the optimal combination of parameter values for the two-phase neighbourhood range.

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>58.49%</td>
</tr>
<tr>
<td>$F_1$-measure</td>
<td>70.06%</td>
</tr>
<tr>
<td>Recall</td>
<td>58.49%</td>
</tr>
<tr>
<td>Precision</td>
<td>87.32%</td>
</tr>
</tbody>
</table>

Table 5.2: Optimal combination of parameter values for the two-phase neighbourhood range.

<table>
<thead>
<tr>
<th>N</th>
<th>w</th>
<th>a</th>
<th>I</th>
<th>k</th>
<th>t</th>
<th>$R_1$</th>
<th>const</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>20</td>
<td>5</td>
<td>5</td>
<td>250</td>
<td>6</td>
<td>5</td>
<td>10</td>
<td>1.2</td>
</tr>
</tbody>
</table>

As we expected the number of discovered motifs was lower, we found 23 whereas in the previous method we had discovered 31, but again not as small as the expected number of different motifs. The total number of correct occurrences is 62 out of 106. We had 9 substitution errors and no insertion errors. The substitution errors probably occur because of the wider neighbourhood range in the second phase which is $R_1 + p = 10.75$, compared to 10.5 when we used the initial method.
From Table 5.2 we can also observe that the \textit{const} parameter is higher than 1.1, as it was in the most cases in the previous section. This is natural if we think that now we have a stricter (i.e. smaller) initial neighbourhood range so the multiplication of the \textit{const} parameter with R, will not give distances that will exclude all the other motif seeds.

The parameter \( p \) in Table 5.2 shows how much we will widen the neighbourhood range in the second phase (i.e. the \( p \) value will be added to the initial value of the neighbourhood range \( R_1 \)). Experiments for this parameter were ran over a small range around the optimal values of the neighbourhood range parameter \( R \) of the \textit{initial algorithm} mainly due to time pressure. An example of how the \( p \) parameter affects the performance of the algorithm can be shown in Table 5.3 for the combination of parameter values given in Table 5.4.

\textbf{Table 5.3:} Different values for the \( p \) parameter. In this example the optimal choice for the parameter \( p \) is 1.25.

<table>
<thead>
<tr>
<th>( p )</th>
<th>1</th>
<th>1.25</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>54.72</td>
<td>55.66</td>
<td>52.83</td>
</tr>
<tr>
<td>F-measure (%)</td>
<td>69.46</td>
<td>69.41</td>
<td>64.77</td>
</tr>
</tbody>
</table>

\textbf{Table 5.4:} Combination of parameter values for the two-phase neighbourhood range used for the results shown in Table 5.3.

<table>
<thead>
<tr>
<th>N</th>
<th>w</th>
<th>a</th>
<th>I</th>
<th>k</th>
<th>t</th>
<th>( R_1 )</th>
<th>\textit{const}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>28</td>
<td>7</td>
<td>5</td>
<td>250</td>
<td>5</td>
<td>8</td>
<td>11.5</td>
</tr>
</tbody>
</table>

In Figure 5.2 we compare the results between our extended method and the baseline method of [Chiu et al., 2003] that we implemented in the previous chapter. We observe that the accuracy of the extended method is higher than the baseline and only in the subsequences of length 30 the baseline algorithm has a better performance. We should note that due to limited time, for the extended method we did not experiment on a thorough combination of parameter values as we did for the baseline. So the results for the extended method may be even higher.

Finally, Figure 5.3 shows us the performance of the extended algorithm, where for each different subsequence length we plot accuracy, \( F_1 \)-measure, recall and precision.
Figure 5.2: Accuracy of the extended method versus the baseline. We observe that the two-phase neighbourhood range method has a better performance. We should note that the graph stops at 70% accuracy.

We observe that the behaviour of the extended algorithm is quiet similar to the baseline, with a slightly better performance.

Figure 5.3: Performance of the two phase algorithm for different subsequence lengths. We should note that the graph start at 40% performance.
Chapter 6

Discussion

The initial performance of the discovery algorithm was quiet good and we managed to improve it by applying the two phase neighbourhood range method. But this improvement comes with an expense, since we have to introduce another parameter \( p \) to the algorithm, which controls how much we should widen the neighbourhood range in the second phase.

As we already mentioned the high number of tunable parameters, plus the one that we introduced, make the algorithm not so attractive for users that do not have domain knowledge. Also, the optimal combination of parameters is sometimes a black art, since the parameter values have a big effect on the performance of the algorithm and they are also strongly connected to each other. In next section we discuss some improvements that could be done so as to reduce the number of tunable parameters.

The constraint of time did not allow us to make many useful experiments by using a wider range of parameter combinations, especially in the extended method that we proposed. We believe that the final performance would get slightly better, but we would not have a great improvement since the assumptions that are made put a barrier in the algorithm’s performance.

One of the assumptions, which we believe that has the greatest effect in the performance of the algorithm, is the search for fixed-length motif occurrences. As we thoroughly explained in the previous chapters the duration of spoken words, which in our case are digits, varies greatly even for same words, so when we apply the algorithm it is inevitable that many similar patterns will not be discovered. Another issue that arises with the current implementation is the loss of important information when we discretize the time series data since one symbol becomes responsible for a big region of space.
6.1 Future Work

In this section we propose some directions for future projects that will be related to our current work.

6.1.1 Reduce Number of Tunable Parameters

As we mentioned the big number of parameters that have to be tuned so as to get optimal performance is unattractive for the algorithm. One idea for solving this problem is to estimate some of the parameters in a completely unsupervised way based on separate and unlabeled training data. This approach was adopted by [Yankov et al., 2007], who also used the discovery algorithm of [Chiu et al., 2003] and tried to learn some of the parameters automatically.

They claim that they could learn off-line the optimal combination of the parameters shown in Table 6.1, in terms of efficiency and effectiveness, by requiring the user only to give the minimum subsequence length N. A detailed explanation of how this estimation is done can be found at Section 6 in the work of [Yankov et al., 2007].

Table 6.1: Parameters that could be learned off-line in a completely unsupervised way.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>number of PAA segments (SAX word length)</td>
</tr>
<tr>
<td>a</td>
<td>number of discrete symbols (SAX alphabet size)</td>
</tr>
<tr>
<td>I</td>
<td>number of iterations of the Random Projection algorithm</td>
</tr>
<tr>
<td>k</td>
<td>projection size for the RP algorithm</td>
</tr>
</tbody>
</table>

6.1.2 Variable Length Motifs

In addition to reducing the number of tunable parameters, [Yankov et al., 2007] extended the discovery algorithm to handle variable length subsequences. They achieved this by discovering motif occurrences under uniform scaling. First, they used in the same way the random projection algorithm to discover candidate motif seeds, and then they searched over a small range of similar subsequence lengths to find the one that fits better to the motif seeds. The idea of searching under uniform scaling in time series data is shown in Figure 6.1.
We should mention that this method does not exactly find variable length motifs, since the motif seeds that are found using the collision matrix of random projection are required to be of the same length. An issue that arises using uniform scaling is that increases the time complexity of the algorithm, since by using uniform scaling we have to search for a range of similar subsequence lengths in each comparison that we make.

6.1.3 Motif Discovery as Density Estimation Problem

As we mentioned in the related work Section 2.3, we tried to re-implement the method of [Minnen et al., 2007a], where they formulated the motif discovery as a problem of locating regions of high density in the space of all time series subsequences.

Initially, we have to extract all possible subsequences using a sliding window in a similar way as we did for the first discovery method. Then we need to locate the k-nearest neighbours of each subsequence, but we should be careful to consider only non-overlapping subsequences (i.e. ignore trivial matches). Since the k-nearest neighbour search require $O(n^2)$ time, approximation methods can be used. After locating all the k-nearest neighbours, we compute the density of each subsequence by calculating their distance to the $k^{th}$ nearest neighbour. The next step is to discover the motif seeds by selecting only the subsequences that have the highest densities (i.e. are near the density mode).

After having discovered the motif seeds, we model each of them as a Hidden
Markov Model and the objective is to fit the set of HMMs to the time series data, in order to maximize the total data log-likelihood [Minnen et al., 2007a]. In order to find the best motif, we greedily compute the optimal mapping of each motif seed and then we choose the one that has the highest likelihood [Kapourani, 2013].

The part of the algorithm that we have already implemented is till the step of the density estimation of each different subsequence. From this part we cannot obtain as much information that we would want since the most important and interesting part is learning with the HMMs.

By using HMMs for the motif discovery process has many advantages, which tackle most of the limitations of our previous implemented method. Initially, we allow the motif models to compete and try to explain the data as better they can, so the are able to find the best choice of the motif neighbourhood range. Another advantage, and the most important for our problem, is that it allows the motif models to stretch or shrink which results in variable-length subsequences [Minnen et al., 2007a].

Experimenting with the TIDIGITS corpus, [Minnen et al., 2007a] showed the superiority of the density based method, which had an overall accuracy of 91.7%.
Chapter 7

Conclusion

Our primary aim in conducting this project was to discover and extract *linguistic phrases or words* from input speech data in a completely unsupervised manner. This would give us an indication of how much can be learned from speech data alone, which would be useful for languages with not sufficient training resources. We reviewed a wide range of general time-series motif discovery methods and we investigated how these methods could be applied effectively to speech data, bearing in mind that there was a limited time to implement and evaluate the chosen discovery method.

The discovery method that we chose, first discretizes the time series data using the SAX method and then it applies the Random Projection algorithm to efficiently discover the motifs. Despite its efficiency, we felt that the algorithm did not scale well to very large time series data, thus we did not use the whole TIDIGITS corpus, but only a small part of it.

In the experiments that we conducted, we observed that the method had notable drawbacks. The most important that we can mention is the assumption of fixed-length subsequences, which lead to very poor performance in accuracy and recall since it could not discover all the motif occurrences. Another issue, was the large number of user specified parameters which make the algorithm unattractive for users that do not have domain knowledge.

We initially implemented the discovery algorithm in the same way that was explained by [Chiu et al., 2003] and extended by [Minnen et al., 2007b], which lead to an overall accuracy of 54.72%. Understanding the importance of the neighbourhood range parameter for the discovery process we proposed a two phase neighbourhood range estimation method. By applying this extension to the algorithm the performance of the algorithm increased, leading to 58.49% accuracy.
Finally, we believe that the assumptions made by the method that we re-implemented in this work, put a barrier in its performance and other methods that can adjust to the particularities of the speech data, such as the speaking rate variation, should be investigated.
Bibliography


