# FI MU

**Faculty of Informatics**
**Masaryk University Brno**

# Employing Subsequence Matching in Audio Data Processing

**by**

**Petr Volný**
**David Novák**
**Pavel Zezula**

**FI MU Report Series** **FIMU-RS-2011-04**

**Copyright © 2011, FI MU** **August 2011**

**Copyright © 2011, Faculty of Informatics, Masaryk University.
All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

**Publications in the FI MU Report Series are in general accessible via WWW:**

`http://www.fi.muni.cz/reports/`

**Further information can be obtained by contacting:**

**Faculty of Informatics
Masaryk University
Botanick 68a
602 00 Brno
Czech Republic**

# Employing Subsequence Matching in Audio Data Processing

Petr Volný
xvolny1@fi.muni.cz

David Novák
xnovak8@fi.muni.cz

Pavel Zezula
Masaryk University, Brno, Czech Republic
zezula@fi.muni.cz

January 3, 2012

**Abstract**

We overview current problems of audio retrieval and time-series subsequence matching. We discuss the usage of subsequence matching approaches in audio data processing, especially in automatic speech recognition (ASR) area and we aim at improving performance of the retrieval process. To overcome the problems known from the time-series area like the occurrence of implementation bias and data bias we present a Subsequence Matching Framework as a tool for fast prototyping, building, and testing similarity search subsequence matching applications. The framework is build on top of MESSIF (Metric Similarity Search Implementation Framework) and thus the subsequence matching algorithms can exploit advanced similarity indexes in order to significantly increase their query processing performance. To prove our concept we provide a design of query-by-example spoken term detection type of application with the usage of phonetic posteriograms and subsequence matching approach.

# 1 Introduction

In a past couple of decades we have witnessed an enormous rise of the amount of digital data. First, it is caused by the digitization of the data from many branches of human's activity and the sharing of possibly any knowledge today is realized through a digital channels. Furthermore, and especially in recent years in the age of blogs, social networks and services like YouTube, people produce vast amount of digital content. Machines themselves are big content creators too. Let's just mention modern CT scanners or mesh nets of small devices producing constant flow of data about a measured phenomenon.

One of the data domains that people have accepted to handle in the digital form is audio. Every moment, we can track a vast amount of new audio data being created. Radio stations, television broadcasting, podcasts, lectures, voice chats – these all are the instruments of the creation of digitalized audio data and, more specifically, digitalized spoken utterances. The problem of an Automatic Speech Recognition (ASR) and analysis is more than three decades old. Many approaches were developed to cope with the speech analysis sub-problems like speaker verification, speech transcription, spoken term detection and many others. Solutions suitable for these problems were often related to signal/time-series processing. Modern techniques, like the large vocabulary continuous speech recognition (LVCSR) use large orthographically transcribed speech data to train their sophisticated acoustic and language statistical models and recognizers to achieve good results in automatic speech recognition. The problem can be seen as a classification of a speech where parts of the speech are segmented and classified into known classes, i.e. words from vocabulary. It is expected that subsequent data-mining tasks will be performed on the result of LVCSR.

It is not always feasible to construct such data sets due to the time and expense associated with the annotation of large quantities of audio. This is typically the case for low resource languages for which performing LVCSR is economically less profitable. For example, due to the lack of automatic transcription tools, only a few percent of the recordings of 4,000+ endangered languages, currently being made by linguists, can be analyzed [5]. It also fails when some out-of-vocabulary words are present in an utterance.

Possible drawbacks of LVCSR can be overcome by other approaches like unsupervised methods. Such approaches expects no (or only a minimum) knowledge about the examined utterance and its language specifics and they can be used for pattern discovery in speech, word discovery, or the whole key-phrase detection. It was argued in the literature [4, 35] that the unsupervised way of analyzing speech, compared with the LVCSR acoustic/language models approach, is much closer to the way of human speech perception and elaboration. The problem is that without a large preprocessing that is present in the LVCSR techniques, the unsupervised methods are very computational expensive because often the traditional similarity distance functions like DTW are extensively employed in the process. This is the problem that we would like to address in our work. We aim mainly on the performance improvements that could be achieved by joining our previous achievements in indexing, similarity searching and subsequence matching areas and one of the state of the art ASR methods. Especially when talking about unsupervised methods, which are much more similar to the classical time series data-mining tasks, we can find many places where our knowledge could enhance the performance of the possible applications. To demonstrate this, we have decided to implement demo application that would employ our

general Subsequence Matching Framework, which we are developing, together with one of the known ASR methods and show that many sub-problems of a general ASR can be solved with the subsequence matching approaches.

While time series and subsequence matching are natural part in music or general audio retrieval process, we believe that we can employ our framework there also.

The rest of this report is organized as follows. In Section 2 we overview an audio retrieval problems form the broader perspective, including music and general audio retrieval. The Section 3 describes the state of the art time series and signal similarity search approaches. In Section 4 we discuss the subsequence matching problem and its applicability for enhancing time series retrieval. In Section 5, our Subsequence Matching Framework and its benefits is introduced and finally the description of our demo query-by-example spoken term detection application based on the framework is presented. The whole report is finished with conclusion and future work directions in Section 6.

# 2 Audio Retrieval Overview

Audio retrieval is a very broad discipline ranging from general sound matching over music retrieval to sophisticated speech related methods like speaker identification and automatic speech recognition. In this chapter we make a brief overview of the approaches and techniques for audio related problems that are recognized by the majority of the research community and where, as we believe, it could be helpful to employ advanced techniques for indexing, time series similarity, and subsequence matching.

## 2.1 General Sound Similarity

In the case of general sound similarity, we have no prior knowledge about the sound like what is the origin of the sound. It must be presumed that it could be emitted by anything from singing birds or see waves to digital sound processor, so one cannot even tell whether the sound is of artificial or natural origin. Because of that, we can use only very low-level features for the sound description. MPEG-7 multimedia description framework [41] is one that is considered as a standard for describing multimedia content and it also includes the basic set of seventeen low level audio descriptors for audio features which can be divided into these six groups:

- basic descriptors – instantaneous waveform and power values,
- basic spectral descriptors – log-frequency power spectrum and spectral features (for e.g. spectral centroid, spectral spread, spectral flatness),
- basic signal parameters – fundamental frequency and harmonicity of signals,
- spectral timbral descriptors – log attack time and temporal centroid,
- temporal timbral descriptors – log attack time and temporal centroid,
- spectral basis representations – a number of features used in conjunction for sound recognition for projections into a low-dimensional space.

Depending on specific application objectives, these descriptors can be combined in various ways.

## 2.2 Music Information Retrieval

Music information retrieval, herein referred to as MIR, covers a broad range of topics. Although, the idea of using computers for the purpose of MIR is about four decades old, the true explosion of interest in this topic took place in 1990's. One of the factors that contributed to it was an establishment of MIDI format as basis for music information sharing. MIDI, as clearly-defined mathematical-based musical format, simplified music analysis and processing and brought new possibilities for music indexing. The next milestone was wide utilization of compression formats such as MP3 and Ogg Vorbis together with the increasing amount of freely available musical databases available for searching. Particular approaches for MIR are partly derived from the representation of the music that one works with; therefore we make a short overview of these.

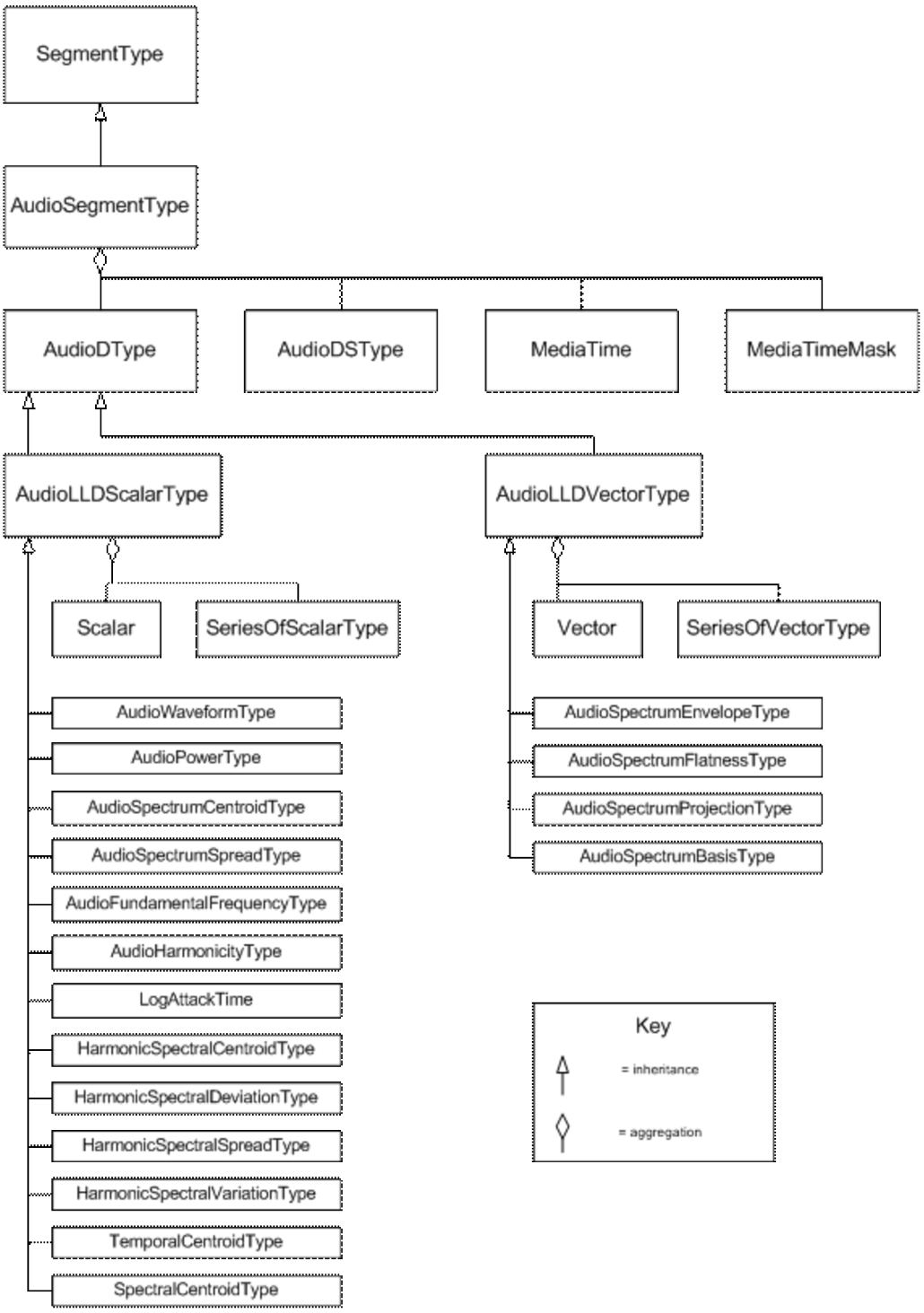


Figure 1: Hierarchy of MPEG-7 audio descrirptors. The structure depicts inner classes of an MPEG-7 framework.

**Sampled Audio** Very accurate when it comes to interpreting the music but very expensive from the MIR point of view. The reason for this is that it consists of thousands of samples of sound taken every second resulting in potentially very large files. Furthermore, sampled audio representation is not robust against noise or small variations of music so it can eventually decrease the performance of the query. Therefore, sampled audio representation is rarely used as an audio description format.

**Musical Instrument Digital Interface** MIDI is a comprehensive type of a representation. It means that it is suitable for quite accurate recreating of an original music data. In contrast to sampled audio, it is much less verbose format. Instead of producing thousands samples per second depicting frequency and amplitude of the sound, it works with discrete alphabet of tones and instruments and it tries to capture when the particular tone starts and where it ends. Although, the re-created sound cannot reach the quality of sampled audio, it is accurate enough for the listener to recognize the original tune. Moreover it is much more robust against background noise and it is much cheaper to process.

**Reduced Representations** The goal of members of this class of representations is not to transcribe complete musical information but only some of its features. It is not usually possible to reconstruct the original music but it gives us good amount of information about features on which basis we can quite easily compare two pieces of an audio data. Here, again, we would like to mention MPEG-7 standards and its audio descriptors. These can be expressible by sole number - scalar descriptors, or holding more complex information in a form of multidimensional vectors. MPEG-7 descriptors and theirs membership to one of the mentioned groups can be seen on Fig. 1.

### MIR Applications

There are several types of applications that employ MIR and they differ in a purpose of usage. One big class is a query by humming applications where an application expects human voice as an input which can be either a bunch of tones that should form a melody or even a part of the song hummed with lyrics. Other type of such an application could be genre detection or instrument detection; there are also applications that try to guess the mood of the tune and so on. Each needs a bit different set of features to be extracted or comparing them in a bit different way.

**Query by Humming** Query by humming usually referrers to a collection of input methods where the human voice is required to produce the query. The method was popularized by Ghias et. al in 1995 [17] and many works tried to enhance the query by humming approach [48] and sometimes it was also with the usage of MPEG-7 descriptors [1]. Nowadays, the technology in indexing and querying vast musical databases allows commercial projects to offer query by humming type of application to the widest audience. As an example lets mention SoundHound; it offers mobile application where user hums a small part of tune (maybe he wants to know the name of the song or want to buy it) and gets full name of the song and other metadata with the immediate possibility to buy it.

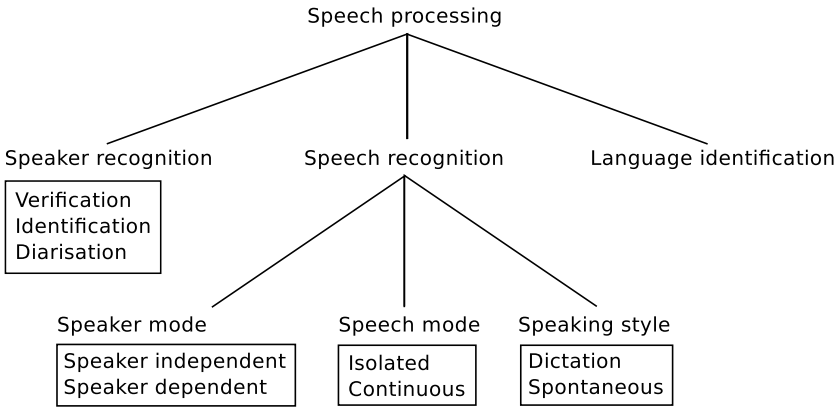


Figure 2: Overview of the speech processing areas.

## 2.3 Speech Related Areas

For humans, speech is the most natural form of communication, which leads to the fact that speech processing is one of the most exciting areas of the signal processing. In spite of the great effort of the research community, there is still significant communication gap between humans and machines. The spoken word would be the perfect interface for many real-life applications. Although we can observe rising number of applications that uses human voice as an input, solving speech processing problems is still very challenging.

In this section we make a brief overview of the most significant areas related to speech, which are depicted on Fig. 2. We name three basic application classes: speaker recognition, speech recognition, language identification. Speaker recognition can be further divided into speaker verification, identification and diarisation. Speaker verification, text dependent or text independent, is the case where we want to verify the identity of the person through specifics of his voice. That means that the system knows which person should be verified and the output is either yes or no or maybe some probability.

On the other hand, in the case of speaker identification, the system tries to guess who is speaking. In other words, there is a database of speakers and an application matches an input voice with indexed entries. Speaker diarisation is a method for distinguishing between speakers in the multi-person conversation and it tries to divide the conversation into utterances that belong to particular speaker.

There are also very specific areas like gender recognition for recognizing a gender of a speaker or language identification. The recognition method is usually based on statistical models of male/female speakers and models for specific languages.

Finally, there is a wide area of speech recognition applications, where one can find many variables for the particular application like whether it is speaker dependent or speaker independent, whether it needs to perform the recognition for continuous speech or preprocessed short utterances, or whether the speaker is aware that his words are to be recognized and she sort of dictates the text or if the system should recognize any spontaneous speaking style. All these variables influence the choice of the proper ASR approach for the specific application.

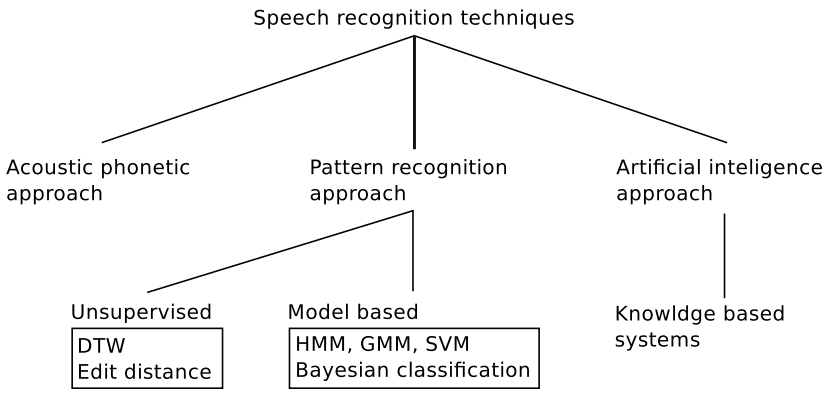


Figure 3: Taxonomy of techniques used in ASR.

**Automatic Speech Recognition**

One of the most popular approaches for the ASR is the large vocabulary continuous speech recognition (LVSCR). This technique usually uses specific models for each language and a decoder that extracts posterior probabilities of sub-word phonetic units. With the aid of the trained models (usually Hidden Markov Models) the input in the form of sub-word phonemes is classified as one of the words in vocabulary. These sub-words phonetic units are recognized by so-called phonetic recognizers. In this phase the acoustic model of the language is used to produce posterior probabilities of the sub-word units for the raw audio input. The output can be represented as a confusion network. In the case that we also have the knowledge about the speaker of the analyzed utterance, we can also employ a speaker specific acoustic model that was trained on the sample data produced by that speaker.

Basic components of a speech recognition process are front end and decoder. Decoder usually can not work with raw audio data. It works only with features extracted by front end. A good example of a wide spread representation is Mel-Frequncey Cepstral Coefficients (MFCC) [44]. MFCC is the most widely used among acoustic signal representations [34, 36] and not only for the sake of speech recognition (it is widely used for a speaker recognition tasks too).

The taxonomy of speech recognition related techniques can be seen on Fig. 3.

**Unsupervised Methods**

Unlike the model based approaches, the pattern recognition based on unsupervised methods have no, or only small, prior knowledge about the language being used or the speaker. In other words, it is possible to use this approach even for languages or dialects where no linguistic corpora are available. The unsupervised methods [21, 4, 35, 43] represent quite recent stream of research and it can be seen as a 'back to the roots' approach because it has much more common with the early ASR methods than with the current state of the art model based ways of ASR.

Unsupervised methods represent departure from traditional models of speech recognition, where the goal is to classify speech into categories defined by a given inventory of lexical units – phonemes or words. Instead, such an inventory is discovered in an unsupervised manner, where words, sub-words, word-like units or generally patterns in speech are extracted by exploiting the structure of repeating patterns within untranscribed audio stream.

As we observed, methods used for unsupervised extraction of such a patterns are often modifications of linear programming algorithms based on DTW approach. This might be one of the causes for performance issues of the unsupervised approaches and the reason for the fact that current state of unsupervised techniques do not allow to achieve the same performance as a model based approaches.

# 3 Time Series Processing: State of the Art

Time series data mining tasks got a lot of attention in last decade. It was caused by the data explosion in many branches and the need to handle vast amounts of data effectively and efficiently. In this section, we provide the reader with an introduction to the mentioned topics. We overview the most important similarity functions and data representations that are meaningful from our point of view. Furthermore, we discuss recent problems related to the outlined areas. Generally, there are four different tasks in time series data mining research [24]:

- Indexing (Query by content): Given a query $Q$, similarity function $D(Q, S)$ and indexed data in DB, find the nearest match in DB.
- Clustering: Find natural groupings of the time series in database DB under some similarity/dissimilarity measure $D(Q, S)$.
- Classification: Given an unlabeled time series $Q$, assign it to one of two or more predefined classes.
- Segmentation: Given a time series $Q$ containing $n$ data points, construct a model $\bar{Q}$, from $K$ piecewise segments $(K << n)$ such that $\bar{Q}$ closely approximates $Q$.

We aim mostly at the first case, but you will see that some approaches can be used to solve more than one of the mentioned tasks. First, we explain the lower bounding lemma and why it is so important.

## 3.1 Lower Bounding

The lower bounding lemma was introduced in [15] and it is very desirable property of representations and similarity models. It enables to prune big parts of the space being searched in a computationally cheap way and it guarantees no false dismissals in an answer (but it can contain false alarms). The key advantage of this approach is that we use the expensive true distance function only on the small subset of the indexed data which were selected as the answer candidates by the cheaper lower bounding mechanism.

**Lemma 1.** *To guarantee no false dismissals for range queries, the extraction function* $F$ *and distance functions* $\mathtt{Dist}$ *and* $\mathtt{LB_Dist}$ *should satisfy the following formula:*

$$\mathtt{LB_Dist}(F(S), F(Q)) \leq \mathtt{Dist}(S, Q)$$

For instance, it was proven [15] that DFT is lower bounding transformation for Euclidena Distance.

To compare representations according to their lower bounding properties the tightness of lowerbound measure [14, 23] is used. Its vale is between 0 and 1 inclusive and is computed as:

$$TLB = \mathtt{LowerBoundDist}(S, Q)/\mathtt{TrueEuclideanDist}(S, Q)$$

The higher the TLB value, the better the lower bound. TLB seems to be very meaningful measure and it seems to be the current consensus in the literature [6, 9, 10, 22]. The advantage of the measure is that it is implementation-free and independent on external variables, e.g. software or hardware testing infrastructure. With this measure one can easily predict indexing performance.

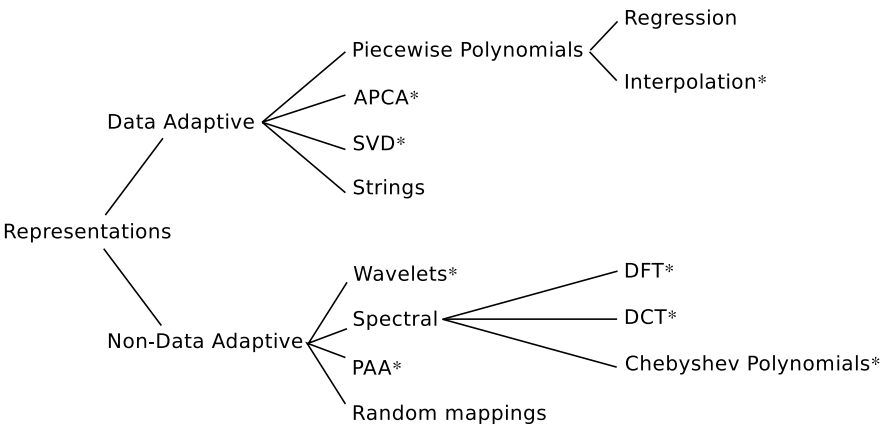


Figure 4: Basic hierarchy of representations. Those with asterisk after the name allows lower bounding.

## 3.2 Representations

Dimensionality reduction is one of the most important tasks in time series processing. The desirable features of the transformed representation are a good ratio between the level of reduction and the accuracy of an approximation of the original time series data and the ability to provide lower bounding options for a distance function. In this section we provide a brief overview of meaningful representations that were widely accepted by the research community (See Fig. 4).

Generally, we can look at representations from several points of view:

- data adaptive/non-data adaptive – the difference is whether the method reflects the features of the time series being transformed,
- continues/discrete domain representations – coefficients are from some continuous domain, e.g. real numbers, or from finite set of values (string representations) symbols; The latter is often called string representation,
- allows/disallows lower bounding – is it proven, that we can lower bound true distance on the original time series with the reduced representation?

The basic division of representations that we are going to overview can be seen in Fig. 4.

#### Non-data Adaptive Representations

**Discrete Fourier Transform** Discrete Fourier Transform was the first proposed method for time-series dimensionality reduction [15]. It transforms a time-series from the time/space domain into the frequency domain. A fast algorithm exists for such a transformation in $\Theta(n \log n)$ time [12]. The DFT represents the time-series data by using Fourier coefficients (a combination of sine and/or cosine waves, each represented as a complex number). Only the first few coefficients are required to approximate the original time-series since the slow changing/low frequency part of the sequence is the underlying shape of the sequence. The coefficients beyond the first few have such low amplitude that they only provide a very small gain in representative accuracy but produce a greater decrease in the compression. An advantage of DFT is that it maintains

the property of allowing Euclidean distance calculations. Due to the exclusion of some coefficients, and therefore some positive values, the Euclidean distance calculation of the two DFT's guarantees the lower bound of the Euclidean Distance [15].

**Discrete Cosine Transform** A discrete cosine transform (DCT) [28] expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. In particular, a DCT is similar to the discrete Fourier transform (DFT), but using only real numbers. DCT is operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample.

DCT also takes place in a process of extracting more sophisticated audio descriptors like Mel Frequency Cepstrum Coefficients (MFCC).

**Discrete Wavelet Transform** Discrete Wavelet Transform (DWT) [8] is similar to DFT in many aspects, except that rather than using sine and cosine functions, it uses wavelets as base function. Wavelets are basis functions used in representing data or other functions. Wavelet algorithms process data at different scales or resolutions in contrast with DFT where only frequency components are considered.

Another difference is the result of the transform. While the DFT transform the time-series into the frequency domain, DWT transforms the time-series into the time/frequency or space/frequency domain. Haar wavelets [8] are often used as s good example of DWT approach because of its good pruning power (i.e it is a good lower bound of Euclidean distance).

**Piecewise Aggregate Approximation** According to Piecewise Aggregate Approximation (PAA), to reduce the data from $n$ dimensions to N dimensions, the data is divided into N equi-sized "frames". The mean value of the data falling within a frame is calculated and a vector of these values becomes the data reduced representation [23].

#### Data Adaptive Representations

**Single Value Decomposition** Singular Value Decomposition (SVD) [28, 39] is an optimal dimensionality reduction technique on matrices under the Euclidean metric. The SVD reduction is suitable for use on any matrix, including applications to indexing images and other multimedia objects. Keogh et al. [23] first used the SVD on time-series data, noting its advantages and disadvantages. The difference with the other proposed techniques as opposed to this one is that the others are local transformations, but SVD is a global transformation.

**Adaptive Piecewise Aggregate Approximation** APCA is a modification of PAA that allows arbitrary length frames in contrast with PAA that divides a time-series into N frame of equal length. This allows indexing using standard multi-dimensional index structures by mapping each mean value to each dimension. APCA differs from PAA in that by allowing arbitrary length frames, an extra value must be stored to identify the length of each window. This led researchers to believe APCA did not allow indexing [45]. Chakrabarti et al. [7] showed that APCA could indeed be indexed using multi-dimensional indexing structures. An APCA representation requires two numbers per window.

## 3.3 Distance Measures

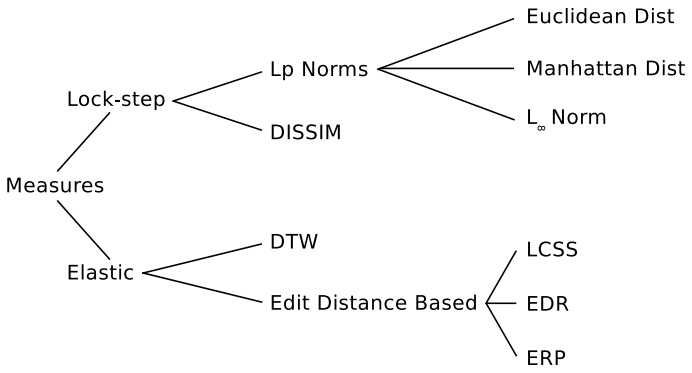


Figure 5: Basic hierarchy of distance functions.

Measuring the distance between objects and thus the similarity is essential for any data mining task. In a perfect case, the distance measure would be cheap to compute, invariant against all possible transformations and with the possibility to parametrize it to give results that are close to human perception of similarity on the given objects domain. Futhermore, it is desireable for the function to obey axioms of a metric space, because when those axioms are satisfied, we can use advanced indexing techniques and perform queries on the indexed data very effectively [47].

Metric space $\mathcal{M} = (\mathcal{D}, d)$ is defined as a set of objects $\mathcal{D}$ and the distances between them. The distance is computed by the function $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ that must satisfy the following porperties. $\forall x, y, z \in \mathcal{D}$ hold:

- $d(x, y) = 0 \Leftrightarrow x = y$ (identity)
- $d(x, y) = d(y, x)$ (symetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

The key axiom is triangle inequality which can be exploited to prune big parts of the indexed space and return the best matching objects very quickly. Common types of queries is a range query and kNN query. First one returns a set of objects that are within the given diameter $r$ from the query object $q$. The latter one returns $k$ nearest objects to the query object $q$.

#### Lock-step Measures

This class of distance functions is suitable only for time-series with the same length. It measures the distance between every particular part of the sequence and expects that these parts were taken in the same time interval.

**Lp Norms** Lp norms is a set of functions that assign strictly positive value to a vector in a Lp space. We will consider only two special cases of Lp norms. The ubiquitous Euclidean norm and Manhattan norm.

Although very simple and straightforward, Euclidean distance is one of the most used similarity measures in time series data mining. It is a classical lock-step similarity method so it can be used only for comparing segments of the same length. It is not robust against time shifting,

scaling and to any other transformations. As we said the Euclidean norm is a special case of Lp norm where $p = 2$, so the length of vector $x = (x_1, ..., x_n)$ is defined as:

$$\|x\| = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

It has been shown that even with the outlined disadvantages of Euclidean distance function it is very reasonable to it use in many fields of time series data mining. Good feature of this measure is that it satisfies properties of a metric function.

The second Lp norm we are about to mention is a Taxicab norm or Manhattan norm. In this case the $p = 1$.

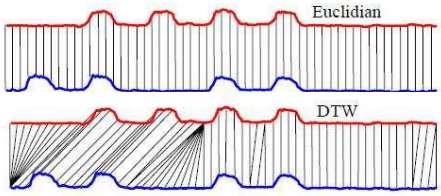


Figure 6: Euclidean distance vs. DTW.

**Elastic Measures**

These measures allows to compare sequence even if they have different lengths. It is allowed by the ability of time warping (See Fig. 6).

**Dynamic Time Warping** Dynamic Time Warping (DTW) ads sort of elasticity to the process of the time series comparison. It allows warping of sequences in a time to eliminate scaling or gaps on the time axis. Semantically, it means that for example in an audio retrieval process, we can identify a spoken word even if it was spoken in a different tempo than is the tempo of the indexed reference word. On Fig. 6 you can see the difference in a way of pairing values of the series being compared by fixed step Euclidean Distance and elastic DTW. Generally, DTW tries to find an optimal match between two sequences. More formally, Dynamic Time Warping is a linear programming method for finding a minimum cost path in an accumulation matrix.

The problem of DTW is a performance and sometimes inappropriate behavior, when it applies a lot of warping. The performance problem of DTW is also given by the fact, that it is not a metric function and thus cannot be indexed as a metric space.

The problem of performance and misbehavior was addressed by the introduction of Sakoe-Chiba band [40] and Itakura Parallelogram [34] which ideas were to constraint the search space only to a part of the matrix along the diagonal path (See Fig. 7).

To enhance the performance of the data retrieval applications that were using the DTW method, the lower bounding functions for the DTW that would be indexable were proposed. The lower bounding function introduced by Kim et al. [26] (known as LB_Kim) extracts 4 numbers as a feature vector for each sequence. The features are the first and last elements and the minimum and maximum values.

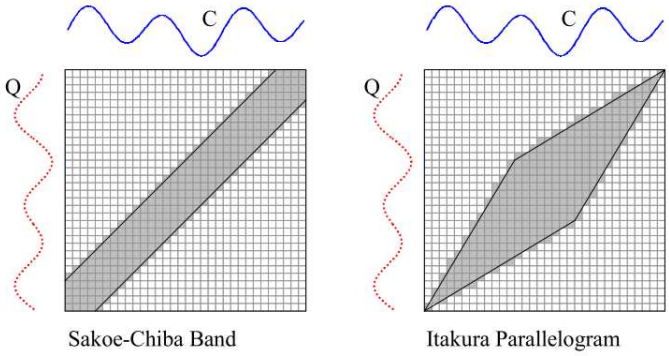


Figure 7: Global constraints limit the scope of the warping path, restricting them to the gray areas. The two most common constraints in the literature are the Sakoe-Chiba Band and the Itakura Parallelogram [25].

Another lower bounding function LB_Yi was introduced by Yi et al. [46] and it exploits the fact that all points in one sequence that are larger (smaller) than the maximum (minimum) must contribute at least squared difference of their value and the maximum (minimum) value of the other sequence in the final DTW distance.

Although, they both added the desirable property of lower bounding, the lower bound was still too loose. This issue was addressed by Keogh et al. in the work [25] introducing LB_Keogh, which idea is to construct envelopes (based on the idea of global constraint of Itakura parallelogram) around the original signal and use PAA representation for both *Upper* and *Lower* envelope borders. On these reduced envelope representations the special lower bounding function LB_PAA (computes the ED between the envelopes) is computed, and Minimum Bounding Rectangles (MBR) covering the original sequence are created. Then with the aid of $\mathrm{MINDIST}(Q, R)$ function, where Q is the query sequence and R is the MBR, the lower bounding distance is computed.

To our best knowledge, LB_Keogh is the best lower bounding technique for DTW approaches.

**Edit Distance Based Functions** To overcome the fact that DTW, in its pure form, is not eligible for exact indexing, i.e. indexing in metric spaces, the inspiration from edit distance based approaches gave a birth to the Edit Distance on Real Sequence (EDR) [11] and Edit Distance with Real Penalty (ERP) [9]. They both exploit the observation that the warping in one sequence can be seen as a gap addition in the other sequence, which takes the problem of warping to the level of edit distance like problem. The main difference between the DTW approach and edit distance based approaches is the penalization for warping which the DTW approach lacks.

# 4 Subsequence Matching strategies

So far have we discussed representations and similarity models that are used for general time series data mining tasks but we have not mentioned anything about subsequence matching yet. In general, subsequence matching is a specific problem in the area of sequence matching. Therefore, all the outlined methods are substantial to both sequence matching and subsequence matching. The latter differs in indexing and retrieval strategies but somewhere in the process it goes down to one of the mentioned similarity measuring methods on the given representations.

A specific application is heavily determined by its data meaning and interpretation. So prior to the inventing subsequence matching strategies one must ask few substantial questions. What is the meaning of the data? Do we seek for trends or patterns? Is it possible to compare data with different lengths? Is time warping a desirable feature during search? What will be the size of the window? What will be the volume of the indexed data? etc. As you can see the process has many variables that we must take into account before we start tailoring the subsequence matching approach for the specific problem.

This section covers basic problems and techniques used in subsequence matching applications. We will discuss usage of sliding and disjoint windows during indexing and querying, the effect of the chosen windows size on the performance of the application, discussion on where to employ dynamic time warping and where to use Euclidean distance, and finally specific problem of finding motifs in the time series.

In the following text we will use this notation:

| Symbols | Definitions |
|---|---|
| $Len[S]$ | The length of sequence $S$ |
| $S[k]$ | The k-th value of the sequence $S$ |
| $S[i:j]$ | Subsequence of $S$ including values between $S[i]$ to $S[j]$ inclusive. |
| $D(Q,S)$ | Distance between two sequences $Q$ and $S$ |
| $s_i$ | The $i$-th disjoint window of sequence $S$ |
| $\omega$ | Length of the (sliding/disjoint) window |
| $\epsilon$ | User defined tolerance |

## 4.1 Sliding And Disjoint Windows

For the sake of subsequence matching, we need to extract subsequences both when indexing and querying data. There are two related major techniques for creating subsequences:

- disjoint window – it divides the given sequence into disjoint windows. where one starts where the other ends,
- sliding window – it creates all possible windows of given length $\omega$ that can be extracted from the given series. In other words, if one starts at position $i$ within the sequence, the other start at position $i+1$.

Both, sliding and disjoint windows are used in subsequence matching techniques to achieve the state, where each window created from the query can be compared to each window in an in-

dex. First time, this concept was used in field opening paper [15] by Faloutsos et al. (FRM in short). There, they use sliding windows for indexed data and disjoint windows for queries. To reduce the amount of subsequence to subsequence distance computations, not all subsequences are indexed in the database, but minimum bounding rectangles representing a bunch of windows (MBR) are used instead. So in the case the range query is initiated, query disjoint windows are first compared to the indexed MBRs and only those MBRs, where at least part of MBR is in the range, are selected as a candidates for the answer. Then all disjoint windows from the candidate MBRs are taken and all false alarms (windows not in range) are found and dismissed from the precise answer. Although, this lower bounding technique ensures no false dismissals, it is not very efficient and produces a lot of false alarms.

The opposite way of using sliding and disjoint windows was introduced by Moon et al. [32]. They divide data sequences into disjoint windows and the query sequences into sliding windows. Hence, this approached exploits duality in constructing windows, it was called DualMatch. It has been shown that the duality based approach is correct (i.e. it incurs no false dismissals) and that it outperforms FRM. DualMatch reduced drastically the number of points that need to be stored to $1/\omega$ of that of FRM. The disadvantage of DualMatch is the upper bound for window size which in return caused additional false alarms due to the *window size effect* (see section 5.2)

Another method, GeneralMatch, was brought by Moon et al. in 2002 [31]. It is based on a generalization of constructing windows. The authors introduced the concept of J-sliding and J-disjoint windows with the specified J sliding factor and they defined them as follows:

**Definition 1.** *A* J*-sliding window* $(1 < J < \omega)$ $s_i^J$ *of size* $\omega$ *of the sequence* S *is defined as the subsequence of length* $\omega$ *starting from* $S[(i-1)*J+1](1 < i < \frac{Len(S)-\omega}{j} + 1)$.

Intuitively speaking, if we have $\omega = 16$ and $J = 4$, we construct windows by shifting subsequence of length 16 and by 4 entries, and thus, the starting points of the 4-sliding windows are $S[1], S[5], S[9]$, ..., respectively.

**Definition 2.** *A* J*-disjoint window* $(1 < J < \omega)$ $q^J_{(i,j)}$ *of size* $\omega$ *of the sequence* Q *is defined as the subsequence of length* $\omega$ *starting from* $Q[i+(j-1)*\omega](1 \le i \le J, 1 \le j \le \frac{Len(S)-i+1}{\omega}))$ *in* Q

Again, if we have an example where $\omega = 16$ and $J = 4$, we construct windows $Q[i : i+\omega-1], Q[i+\omega : i+2\omega-1]$,... by dividing Q[i:Len(Q)] into disjoint windows for every i where $(1 < i < 4)$.

**I/O Issues** Logically, when we divide the sequences into many subsequences, the amount of comparisons during the query processing rises. This also implies that subsequence matching algorithms do lots and lots of I/O operations needed for fetching particular subsequences one by one again and again. One of the advanced subsequence matching methods addressed this problem by introducing *deferred group subsequence retrieval* [19]. This technique tries to cope with the fact that, because of excessive disk I/O operations and bad buffer utilization, we might have to read same data pages repeatedly from the disk. The basic idea of the proposed solution is to delay a fixed size set of subsequence retrieval and enable batch retrieval.

## 4.2 Window Size Effect

The effect of a window size on a performance was first discussed in previously mentioned Dual-Match approach [32]. The windows size effect is caused by the application of two lemmas that were introduced in FRM:

**Lemma 2.** *When two sequences S and Q of the same length are divided into p windows $s_i$ and $q_i$ ($1 \leq i \leq p$) respectively, if S and Q are in $\epsilon$-match, then at least one of the pairs $(s_i, q_i)$ are in $\epsilon/\sqrt{p}$-match. That is, the following equation holds:*

$$D(S, Q) \leq \epsilon \Rightarrow \bigvee_{i=1}^{p} D(s_i, q_i) \leq \epsilon/\sqrt{p}$$

**Lemma 3.** *If two sequences S and Q of the same length are in $\epsilon$-match, then any pair of subsequences $(S[i : j], Q[i : j])$ are also in $\epsilon$-match. That is, the following equation holds:*

$$D(S, Q) \leq \epsilon \Rightarrow D(S[i : j], Q[i : j]) \leq \epsilon$$

Application of Lemmas 2 and 3 for long query sequences causes false alarms. That is, when two sequences S and Q are divided into p windows $s_i$ and $q_i$ ($1 \leq i \leq p$) respectively, although a pair $(s_i, q_i)$ is in $\epsilon/\sqrt{p}$-match, the distance between S and Q may be greater than $\epsilon$. To reduce this kind of false alarms, it is reasonable to use as large windows as possible. For example, let the window size of the method A be twice as large as that of the method B. Then, by Lemma 2 or 3, a candidate subsequence of the method A must also be a candidate of the method B. However, the inverse does not hold. This effect was defined as the *window size effect* [32]. The size of the window, however, must be less than or equal to the length of the query sequence; thus, the maximum window size depends on the length of the query sequence. Typically, the algorithm have the problem that the performance decreases as the difference between the query sequence length and the window size increases. This problem was addressed in [27, 29]. The proposed techniques employ creating multiple indexes and their usage based on the query parameters. The dependency of the performance degradation on the number of queries and query vs. window size was revealed and it was claimed that the need for multiple indexes tailored for the variable query lengths is crucial. Both mentioned articles introduced heuristic methods for determination of the window sizes for particular indexes based on the distribution of the query sequence lengths.

## 4.3 When We Need To Warp?

Euclidean Distance and Dynamic Time Warping based techniques have both many advocates. Euclidean distance is mostly favored for its simplicity, lower bounding options and the fact that it could be easily computed. On the other side stands much more sophisticated linear programming approach of dynamic time warping. During the past few decades the latter pushed the former out of many applications and it was claimed that the dominance of DTW over ED is inevitable. There are areas where this has become true like in speech recognition and query-by-humming where the warping ability is crucial to match the spoken terms with words from the database or where one can expect that the humming will not have the same tempo as the indexed data that we want to match with the query.

Despite the fact that DTW outperforms ED in these areas, arguments in the favor of ED were raised in the literature [38, 37]. It was argued that the bigger the data collection, the lesser the

need for DTW because it often degrades to the same result as the ED measure. This is caused by the fact that the probability, that there are some data in the collection that are very similar to the possible query, grows with a size of the indexed data collection. So the usage of DTW and ED in this case would bring very similar results but ED is cheaper to compute.

On the other hand, this reasoning cannot be applied generally. We believe that for some domains the warping feature of the DTW is crucial even if the collection is large enough. Again, on the example of the spoken term detection, we do not want to find only the one best match, but to find possibly all variations of the spoken term and this is the case where DTW, or other functions with the warping ability, are irreplaceable. In the case that the warping would be needed only because the different offset of the data, than, when considering subsequence matching, it is better to use ED. We do not need DTW in this case because the offset difference of the sequence can be overcome by the sliding/disjoint windows.

## 4.4 Finding Motifs In Time Series

We can say that finding motifs [30] is specific sub-problem of time series data mining. It aims on finding regular patterns in the time series, usually omitting the exact values and taking only the development or shape of the time series signal. Sometimes this problem is also referred as a rule discovery.

One of the common approaches is to have a predefined set of primitive shapes and than to classify the windows of the original sequence into these shapes. The set of primitives can be found by clustering of windows of predefined length. Each sequence is sliced into disjoint windows and than all the windows are clustered into the set of shape primitives. Each primitive is represented by a symbol and than each sequence can be represented by the string of these symbols [13].

# 5 Subsequence Matching Framework

As we have outlined in the previous sections, the ubiquity of the time series and its usability in a wide range of applications led the research community to the invention of many various approaches. It was shown that the time series whole matching approach is not sufficient in many areas. Therefore the subsequence matching problem had risen and has been addressed by many researches since the first paper on this topic was published [15]. Faloutsos et al. have introduced GEMINI framework, an application model for dealing with subsequence matching which in short we can explain in four steps that the application has to perform:

- slicing the time series sequences into smaller subsequences (originally using sliding window for the indexed data and disjoint window for the query)
- maping each time series subsequence in a lower dimension. We can call this step segmentation (originally using Fast Fourier Transform)
- indexing them in multi-dimensional indexing structure (originally R-tree)
- performing search with a distance function that obeys the rule of lower bounding lemma (originally Euclidean Distance).

Despite the fact that the application model is almost two decades old, it is still vital and suits many cases where the subsequence matching procedure has to be employed. Works that followed that Faloutsos's paper usually tried to enhance only the part of the problem mainly by introducing new methods for dimensionality reduction, new distance functions or a bit of a variance in an indexing and slicing strategy for the subsequence bits. We have mentioned the most significant ones in the previous sections. Those new approaches enhanced the performance of the GEMINI framework but a lot of them were tested only on a constrained, and often artificially created, datasets. It has been shown in [14] that the comparison of those methods is not as clear as some of the authors were trying to declare and so that the results can not be taken as a ground truth. It was argued [24] that the experimental results were computed in various environments with various data sets and with various implementations. This problem was previously called *implementation and data bias*.

The need for the unified testing dataset was settled and fulfilled by the authors of [14] and a collection of testing datasets has been maintained since then, and the authors of new papers were asked to perform the tests on this collection to bypass at least the *data bias* mentioned above. On the other hand, the *implementation bias* often remains.

We would like to address this problem and to present a versatile subsequence matching framework that would be usable for rapid prototyping and testing of the current, but mainly new approaches and would serve as trusted platform for performing these tests. By framework, we mean a complex implementation framework for quick but efficient realization of a wide range of subsequence matching related applications and approaches. It will contain a library of subcomponents, a way to combine them, and an option to create new ones.

We are aware that this goal is not easy one to achieve, because the balance of the boundaries of the framework and the real usability and extensibility must be found so that the developers and scientists would adopt it. Nevertheless, we believe that we can deliver such a framework that can helpful to the community. We can offer the experience with building the MESSIF framework

[3], part of the MUFIN [2] project that allows fast prototyping and testing of general similarity search applications literally on any data domain. Until now, from this paper's point of view, these technologies allowed only dealing with the vectors whole matching problem. What we did was that we used these technologies that were proven by the time and many projects and enriched them with the ability of the subsequence matching.

To prove that our solution is really usable for many applications areas, we have decided to demonstrate it on the application related in speech recognition and analysis that covers many problems and many approaches to solve them and where it may not be clear on the first look that the related applications can be solved or enhanced by the state of the art subsequence matching solutions. Later in the text, we will discuss the demo application that demonstrates the combination of state of the art technologies both from the world of speech analysis and subsequence matching.

## 5.1 Idea

Our Subsequence Matching Framework relies on the MESSIF library that is designed to help to implement similarity search applications in general. We provide several basic classes and an easy understandable architecture for rapid development and testing of subsequence similarity search applications. We have tried to identify the common sub-problems of the subsequence matching process. We have mentioned some in the rough description of the GEMINI framework but we added some new and still let the architecture opened for implementing brand new approaches. To achieve this we build the framework in a modular way. You can imagine *modules* dealing with:

- normalization of the time-series/sequence data,
- transformation for data sequences,
- storage index for subsequences,
- slicing of the sequences into the windows,
- distance function implementations,
- lower bounding techniques.

The concept of *modules* (Fig. 8) should allow good re-usability of the code and will help to overcome the *implementation bias* because once the module responsible for solving some part of the problem is implemented, it should be used by others. The architecture allows to implement subroutines with functionality independent of the specific data type. As you can see on the Fig 8, another substantial part of the framework is the definition of algorithms. The algorithms are responsible for the logic of the subsequence matching, indexing and the execution of the queries. Each algorithm declares its parameters. You can imagine that as slots in which the modules are plugged in. So in the example, the Simple algorithm is working with two slicing modules, one transformation module, one distance function that obeys the lower bounding lemma and two indexes – one for the original sequences and one for the subsequences. Under the definition of the algorithm we can see two instances altering in the usage of modules. This is possible because the implementation of the modules is independent of the data types (to some level of abstraction) and the algorithm itself only uses the interface common to the modules from the same class.

These features allow to quickly prototype and alter the algorithm by changing the modules that are used. So when we have the mechanism of the algorithm implemented using such an interface to the the modules declared as a parameters of this algorithm and have implemented a bunch of modules for these classes, we can easily perform lots of combinations and gather the results.

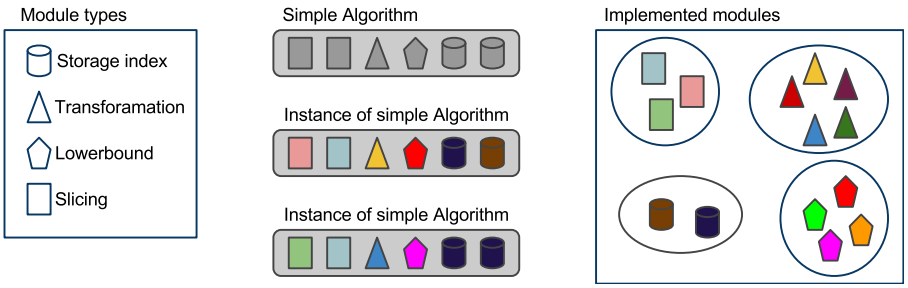


Figure 8: SMF principles.

Another great feature of the framework is that altering these combinations does not require to re-compile the code. With the usage of the MESSIF *batch files* one can instantiate new algorithm with new parameters/modules (also instantiated through the *batch file*) and script the whole series of tests in a very comfortable and fast way.

The binding with the MESSIF framework also brings support for performing various similarity queries including classical range queries, KNN-queries etc.

We can also leverage many of state of the art metric indexes previously developed for MESSIF applications like M-index [33].

## 5.2 Implementation

As mentioned above, the subsequence matching framework extends the MESSIF platform and is a part of the MUFIN project. The whole MUFIN ecosystem and the SMF are developed using Java and it works with related technologies like Remote Method Invocation (RMI), Java Server Pages (JSP) and use the latest Java features like reflection. We can say that the Subsequence Matching Framework layer extends the MESSIF core layer and that the applications written with aid of the framework can use the MESSIF-UI module for creating the web application demos. The MESSIF offers lots of functionality when developing any similarity search application like indexing and querying in a distributed environment or constructing multi-layered indexing network or composing combined queries.

So far we have developed basic skeleton of the subsequence matching framework and now we are trying to proof the concept by implementing various subsequence matching applications with different data domains and different demands. This should help us to find cons of the framework and make the architecture better and more usable for different purposes. The last application domain that has drawn our attention is the speech analysis and generally the audio retrieval.

# 6 Use Case: Spoken Term Detection

To prove the applicability of the SMF, we are in a process of implementing an ASR application that would be dealing with the problem of query-by-example spoken term detection. Besides the framework itself, the application will employ one of the state of the art metric indexes. Similarly as in [20], the application will use the indexing based on posteriogram templates that will be extracted with the aid of BUT phonetic recognizer [42]. In contrast with [20] we will be using other distance function than DTW, which can be seen as an origin for the performance issues related to the computational demands of DTW and the fact that the DTW itself cannot be indexed as a metric space. We believe that we can achieve very similar result in the terms of quality with the usage of some edit distance based metric function like ERD or ERP which allow better indexing in metric structures like M-index [33] and that we can achieve better performance results. We would also like to investigate the possibilities for approximate search in the spoken term detection area.

The concept of phonetic posteriogram is based on the knowledge of posterior probabilities of every phoneme in the phonetic vocabulary for every time frame. It can be represented as graph of phoneme posterior probability in a time (see Fig. 9). For the evaluation of our approaches we have obtained TIMIT [16] and NIST Spoken Term Detection Development Set [18] corpora.

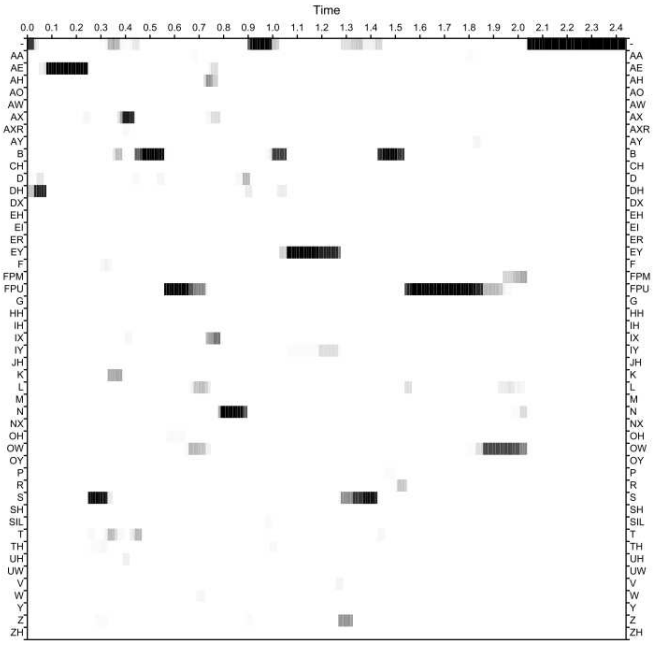


Figure 9: An example posteriorgram representation for the spoken phrase "basketball and baseball" [20].

# 7 Conclusion And Future Work

We have overviewed basic problems and approaches in audio retrieval area and time-series, or generally sequence, processing. We have outlined that we can apply solutions from subsequence matching area suitable for problems from the audio retrieval area. Our framework should contribute to the solutions in an audio retrieval process by employing subsequence matching techniques. In the further research we would like to search for more possibilities for enhancing the

performance aspects of the audio retrieval by using metric space approach and advanced subsequence matching techniques. Things like tuning index parameters or deciding on the best subsequence matching strategy for the particular application are yet to be properly investigated.

## Acknowledgments

This work was supported by national research projects VF20102014004, MSMT 1M0545, GACR 103/10/0886, and GACR P202/10/P220.

# Bibliography


[1] J Batke, G Eisenberg, P Weishaupt, and T Sikora. A Query by Humming system using MPEG-7 Descriptors. In *116th AES Convention*, pages 588–601. AES, 2004.

[2] Michal Batko, Vlastislav Dohnal, David Novak, and Jan Sedmidubsky. MUFIN: A Multi-feature Indexing Network. *2009 Second International Workshop on Similarity Search and Applications*, pages 158–159, 2009.

[3] Michal Batko, D Novak, and P Zezula. MESSIF : Metric Similarity Search Implementation Framework. *Lecture Notes in Computer Science*, 4877(102):1, 2007.

[4] Louis Bosch and Bert Cranen. A computational model for unsupervised word discovery. *Order A Journal On The Theory Of Ordered Sets And Its Applications*, pages 1–4, 2007.

[5] Lou Boves, Rolf Carlson, Erhard Hinrichs, David House, Steven Krauwer, Lothar Lemnitzer, Martti Vainio, and Peter Wittenburg. Resources for Speech Research: Present and Future Infrastructure Needs, 2009.

[6] Yuhan Cai and Raymond Ng. *Indexing spatio-temporal trajectories with Chebyshev polynomials*. SIGMOD '04. ACM Press, New York, New York, USA, 2004.

[7] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems (TODS)*, 27(2):188, 2002.

[8] Kin-Pong Chan and Ada Wai-chee Fu. Efficient Time Series Matching by Wavelets, 1999.

[9] Lei Chen and Raymond Ng. On the Marriage of L_p-norms and Edit Distance, 2004.

[10] Lei Chen and M. Tamer Özsu. Using multi-scale histograms to answer pattern existence and shape match queries, 2005.

[11] Lei Chen, M. Tamer Özsu, and Vincent Oria. *Robust and fast similarity search for moving object trajectories*. SIGMOD '05. ACM Press, New York, New York, USA, 2005.

[12] James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297, April 1965.

[13] Gautam Das, King-ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule Discovery From Time Series, 1998.

[14] Hui Ding, Goce Trajcevski, Xiaoyue Wang, and Eamonn Keogh. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures, 2008.

[15] Christos Faloutsos, M Ranganathan, and Yannis Manolopoulos. Fast Subsequence Matching in Time-Series Databases, 1994.

[16] John S Garofolo. TIMIT Acoustic-Phonetic Continuous Speech Corpus. *Linguistic Data Consortium, Philadelphia*, 1993.

[17] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C Smith. Query by Humming: Musical Information Retrieval in an Audio Database. In *ACM Multimedia*, pages 231–236. ACM Press, 1995.

[18] NIST Multimodal Information Group. 2006 NIST Spoken Term Detection Development Set. *Linguistic Data Consortium, Philadelphia*, 2011.

[19] Wook-Shin Han, Jinsoo Lee, Yang-Sae Moon, and Haifeng Jiang. Ranked subsequence matching in time-series databases. *Very Large Data Bases*, pages 423–434, 2007.

[20] Timothy J. Hazen, Wade Shen, and Christopher White. *Query-by-example spoken term detection using phonetic posteriorgram templates*. IEEE, December 2009.

[21] Marijn Huijbregts, Mitchell McLaren, and David van Leeuwen. UNSUPERVISED ACOUSTIC SUB-WORD UNIT DETECTION FOR QUERY-BY-EXAMPLE SPOKEN TERM DETECTION. *Language and Speech*, pages 4436–4439, 2011.

[22] Eamonn Keogh. A Decade of Progress in Indexing and Mining Large Time Series Databases. *Management*, (1994):1994–1994, 2006.

[23] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases, 2000.

[24] Eamonn Keogh and Shruti Kasetty. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration, 2002.

[25] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, May 2004.

[26] S Kim, S Park, and W Chu. An index-based approach for similarity search supporting time warping in large sequence databases. *Proceedings of International Conference on Data Engineering*, pages 607–614, 2001.

[27] Hyun-Gil Koh, Woong-Kee Loh, and Sang-Wook Kim. *Innovations in Applied Artificial Intelligence*, volume 3533 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, June 2005.

[28] Flip Korn, H. V. Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD Record*, 26(2):289–300, June 1997.

[29] Seung-Hwan Lim, Heejin Park, and Sang-Wook Kim. Using multiple indexes for efficient subsequence matching in time-series databases. *Information Sciences*, 177(24):5691–5706, December 2007.

[30] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. *Time*, pages 53–68, 2002.

[31] Yang-Sae Moon, Kyu-Young Whang, and Wook-Shin Han. General match: a subsequence matching method in time-series databases based on generalized windows. *International Conference on Management of Data*, page 382, 2002.

[32] Yang-Sae Moon, Kyu-Young Whang, and Woong-Kee Loh. Duality-Based Subsequence Matching in Time-Series Databases. *Proceedings of the 17th International Conference on Data Engineering*, page 263, 2001.

[33] David Novak and Michal Batko. Metric Index: An Efficient and Scalable Solution for Similarity Search. In *Second International Workshop on Similarity Search and Applications (SISAP 2009)*, volume 9, pages 65–73. IEEE, 2009.

[34] Lawrence O'Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*, volume 103 of *Prentice Hall signal processing series*. Prentice Hall, 1993.

[35] Alex S Park and James R Glass. Unsupervised Pattern Discovery in Speech. *IEEE Transactions On Audio Speech And Language Processing*, 16(1):186–197, 2008.

[36] J W Picone. Signal modeling techniques in speech recognition. *Proceedings of the IEEE*, 81(9):1215–1247, 1993.

[37] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about Dynamic Time Warping is Wrong, 2004.

[38] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining SDM'05*, volume 21, pages 506–510. Citeseer, 2005.

[39] K. V. Ravi Kanth, Divyakant Agrawal, and Ambuj Singh. Dimensionality reduction for similarity searching in dynamic databases. *ACM SIGMOD Record*, 27(2):166–176, June 1998.

[40] H Sakoe and S Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics Speech and Signal Processing*, 26(1):43–49, 1978.

[41] Phillipe Salembier and Thomas/Manjunath Sikora. Introduction to MPEG-7: Multimedia Content Description Interface. June 2002.

[42] P Schwarz, P Matejka, and J Cernocky. Towards Lower Error Rates in Phoneme Recognition. 2004.

[43] Dirk Von Zeddelmann, Frank Kurth, and Meinard Müller. Perceptual Audio Features for Unsupervised Key-Phrase Detection. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing ICASSP*, Lecture Notes in Computer Science, 2010.

[44] Han Wei, Chan Cheong-Fat, Choy Chiu-Sing, and Pun Kong-Pang. An efficient MFCC extraction method in speech recognition. In *2006 IEEE International Symposium on Circuits and Systems*, page 4. IEEE, 2006.

[45] Byoung-Kee Yi and Christos Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp Norms. pages 385–394, September 2000.

[46] Byoung-Kee Yi, H.V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. *Proceedings 14th International Conference on Data Engineering*, pages 201–208, 1998.

[47] Pavel Zezula, Giuseppe Amato, Michal Batko, and Vlastislav Dohnal. *Similarity Search: The Metric Space Approach*. Springer US, 2005.

[48] Yunyue Zhu and Dennis Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD '03*, page 181, New York, New York, USA, June 2003. ACM Press.