Symbola

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF PHYSICS

GRAMMAR OF TIME
TIME SERIES DATA MINING IN STEROIDS

HOW TO TRANSLATE TIME?

JOHN VERY LONGNAME DOE

Master/BSc in Name of Previous Degree

DOCTORATE IN STUDY PROGRAM NAME

NOVA University Lisbon
month, year

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF PHYSICS

GRAMMAR OF TIME
TIME SERIES DATA MINING IN STEROIDS

HOW TO TRANSLATE TIME?

JOHN VERY LONGNAME DOE

Master/BSc in Name of Previous Degree

Adviser: Mary Doe Adviser Name
Full Professor, NOVA University Lisbon

Co-advisers: John Doe Co-Adviser Name
Associate Professor, NOVA University Lisbon

John Doe other Co-Adviser Name
Full Professor, NOVA University Lisbon

Examination Committee

Chair: Name of the committee chairperson
Full Professor, FCT-NOVA

Rapporteur: Name of a rapporteur
Associate Professor, Another University

Members: Another member of the committee
Full Professor, Another University

Yet another member of the committee
Assistant Professor, Another University

DOCTORATE IN STUDY PROGRAM NAME
SPECIALIZATION IN SPECIALITY NAME

NOVA University Lisbon
month, year

Grammar of Time Time Series Data Mining in Steroids

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.14) [**novathesis-manual**].

Dedicatory lorem ipsum.

Acknowledgements

Acknowledgments are personal text and should be a free expression of the author.

However, without any intention of conditioning the form or content of this text, I would like to add that it usually starts with academic thanks (instructors, etc.); then institutional thanks (Research Center, Department, Faculty, University, FCT / MEC scholarships, etc.) and, finally, the personal ones (friends, family, etc.).

But I insist that there are no fixed rules for this text, and it must, above all, express what the author feels.

"You cannot teach a man anything; you can only help him discover it in himself." (Galileo)

Abstract

Regardless of the language in which the dissertation is written, a summary is required in the same language as the main text and another summary in another language. It is assumed that the two languages in question are Portuguese and English.

The abstracts should appear first in the language of the main text and then in the other language. For example, if the dissertation is written in Portuguese the abstract in Portuguese will appear first, then the abstract in English, followed by the main text in Portuguese. If the dissertation is written in English, the abstract in English will appear first, then the abstract in Portuguese, followed by the main text in English.

In the LaTeX version, the NOVAthesis template will automatically order the two abstracts taking into account the language of the main text. You may change this behaviour by adding

```
\abstractorder(<MAIN_LANG>):={<LANG_1>,...,<LANG_N>}
```

to the customization area in the document preamble, e.g.,

```
\abstractorder(de):={de,en,it}
```

The abstracts should not exceed one page and, in a generic way, should answer the following questions (it is essential to adapt to the usual practices of your scientific area):

1. What is the problem?
2. Why is this problem interesting/challenging?
3. What is the proposed approach/solution?
4. What results (implications/consequences) from the solution?

Keywords: Keyword 1, Keyword 2, Keyword 3, Keyword 4, Keyword 5, Keyword 6, Keyword 7, Keyword 8, Keyword 9

Resumo

Independentemente da língua em que a dissertação esteja redigida, é necessário um resumo na mesma língua do texto principal e outro resumo noutra língua. Pressupõe-se que as duas línguas em questão sejam o português e o inglês.

Os resumos devem aparecer primeiro na língua do texto principal e depois na outra língua. Por exemplo, se a dissertação for redigida em português, o resumo em português aparecerá primeiro, seguido do resumo em inglês (*abstract*), seguido do texto principal em português. Se a dissertação for redigida em inglês, o resumo em inglês (*abstract* aparecerá primeiro, seguido do resumo em português, seguido do texto principal em inglês.

Na versão LaTeX o template NOVAthesis irá ordenar automaticamente os dois resumos tendo em consideração a língua do texto principal. É possível alterar este comportamento adicionando

```
\abstractorder(<MAIN_LANG>):={<LANG_1>,...,<LANG_N>}
```

à zona de customização no preâmbulo do documento, e.g.,

```
\abstractorder(de):={de,en,it}
```

Os resumos não devem ultrapassar uma página e, de forma genérica, devem responder às seguintes questões (é essencial adaptá-los às práticas habituais da sua área científica):

1. Qual é o problema?
2. Porque é que é um problema interessante/desafiante?
3. Qual é a proposta de abordagem/solução?
4. Quais são as consequências/resultados da solução proposta?

Palavras-chave: Palavra-chave 1, Palavra-chave 2, Palavra-chave 3, Palavra-chave 4

Contents

List of Figures

List of Tables

Glossary

This document is incomplete. The external file associated with the glossary 'main' (which should be called `template.gls`) hasn't been created.

Check the contents of the file `template.glo`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

If you don't want this glossary, add `nomain` to your package option list when you load `glossaries-extra.sty`. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "template"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "template"
  ```

Then rerun LaTeX on this document.
This message will be removed once the problem has been fixed.

Acronyms

This document is incomplete. The external file associated with the glossary 'acronym' (which should be called `template.acr`) hasn't been created.

Check the contents of the file `template.acn`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "template"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "template"
  ```

Then rerun LaTeX on this document.
This message will be removed once the problem has been fixed.

Symbols

This document is incomplete. The external file associated with the glossary 'symbols' (which should be called `template.sls`) hasn't been created.

Check the contents of the file `template.slo`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "template"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "template"
  ```

Then rerun LaTeX on this document.
This message will be removed once the problem has been fixed.

Chemical Symbols

This document is incomplete. The external file associated with the glossary 'chemical' (which should be called `template.chs`) hasn't been created.

Check the contents of the file `template.cho`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "template"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "template"
  ```

Then rerun LaTeX on this document.
This message will be removed once the problem has been fixed.

1 Introduction

1.1 Time Series and Challenges of a Data-Driven Society

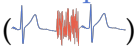
(THIS SHOULD HAVE SOMETHING LIKE: WE BELIEVE THERE IS A LACK OF TOOLS THAT CAN PUT THE INTUITION OF THE USER INTO THE ANALYSIS PROCESS OF TIME SERIES. SEVERAL METHODS ARE ALREAY AVAILABLE BUT WE SHOULD MAKE THEM AVAILABLE AS FUNCTIONAL TOOLS. I HAVE NOT DONE THAT, BUT THIS WOULD BE MY ULTIMATE GOAL AND MOVE TOWARDS THAT.

In recent years, the continuous increase in accessible wearable technology has contributed to a significant amount of data available. The continuous production of data from wearable devices through the usage of mobile phones, smartwatches, hearables, wristbands and other non-invasive wearable sensors has provided a valuable quantity of information. This data often comes as time series, being one of the most common data type in nature [**puttinghuman**]. As reported in *Tankovska et al.*, the wearable devices usage has more than doubled in the interval between 2016 and 2019, reaching 722 million [**tankovska_23_2020**] [**novathesis-manual**], leading to a large volume of time series data being gathered in all possible scenarios, by monitoring patients in healthcare institutions [**cpd_medical_1**, **cpd_medical_2**, **cpd_medical_3**, **cpd_medical_4**, **dataset6**, **dataset7**], tracking everyday activities of humans [**cpd_har_1**, **cpd_har_2**, **review_1**], recording machines in industrial processes or workers motion while performing their tasks [**antonio**, **sara**]. It has never been so easy to gather data about any aspect of our life, work, education, society or industry. Of course, having relevant information about a subject is beneficial, but the overwhelming amount of data brings tremendous challenges in the ability to save, process, analyze and retrieve interpretable and meaningful information from which we can act upon[**bigdata**]. Ultimately, it becomes even harder to have data well structured and labeled, considering that it is a sensitive and time consuming process, which complexity increases with data quantity. This is particularly problematic when developing machine learning applications (remember that Garbage-in Garbage-out - *GIGO*) [**roh2019survey**]. In the work of *Roh et al.* is mentioned that data scientists only

rely on a small portion of the available datasets because it is too expensive to label all the data available [**roh2019survey**], and this is just an example of how much data can be unused.

We believe that we should do more with the data we have and for that, tools should be available to support and help analysts to accelerate the process of information retrieval from time series by making it more expressive and intuitive. In this thesis, we propose several novel methods that contribute to the information retrieval problematic. These methods are designed to help in (1) tell the story behind the time series by means of a visual representation that highlights its structure and organization, (2) make the search of patterns and events with more expressive queries and (3) move towards distance measures that are more human readable. These contributions are part of a general work born with this thesis, called *grammar of time*.

1.2 Linguistic Nature of Time Series

Time Series are a visual domain, from which humans can create a good intuition. It is inherent to our ability to see relevant structures and patterns. The reader can imagine a recurrent shape, such as the QRS complex of an electrocardiogram (ECG) signal that is interrupted by a noisy segment (). When interpreting this signal, we see that it has 3 representative segments and that the first is very similar to the third one. We could then represent the signal by *A B A*. Some shapes may be harder to distinguish, for instance, consider an accelerometer signal of a subject while walking and shifting to jogging regime (). Or a change in the shape of the arterial blood pressure (ABP) signal when there is a change in the subject's posture (). In both cases, the signal has 2 structures of a similar representative periodic pattern (A B).

This visual intuition is also very clear when a (non-)experience analyst is searching for specific shapes or patterns in time series. The reader may agree that scientists or other professionals often resort to describe the shape they are looking for. For instance, a physician may say "*I am searching for the T-wave, that represents the* `large peak`" () or "*I am searching for the QRS complex, that looks like a* `sharp peak followed by a sharp valley` ()". This visual intuition also happens when analysts are trying to find differences between classes of signals. For instance, the following shapes (1)  and (2) are different because "*shape 1 has a* `peak` *where shape 2 doesn't*".

Time series are carriers of information and the presence of a change in the regimes of a time series or the presence of a specific shape in a segment of a time series may be associated with a specific occurrence in the physical world and be attributed a meaning. This notion of structure and meaning is a good approximation of what represents the foundation of a language: grammar and meaning [**grammar**].

Grammar is generally defined as the book of rules that constitutes the structure of a language, and is modeled by the morphology and syntax [**grammar**]. The first is the

structure of words, how these are built or morphed based on context, while the latter consists in organizing words in sequences to form larger linguistic units, such as sentences. Such as a language has morphological and syntax rules that represent its structural information, time series are also organized by a formal structure of ordered subsegments with specific morphological characteristics, organized to build larger segments. This introduces why this thesis is designated *grammar of time* and also introduces the reader further to the problematic that will be explored in this work. We will demonstrate how the developed solutions are helpful to several domains, with a special interest in showing how these can be helpful and meaningful in the context of occupational health.

1.3 Context and Relevance in Occupational Health

Work-related musculoskeletal disorders (WMSDs) prevail as the most common occupational disease in the European Union. These have a global impact on the well being of individuals and their quality of life in a range of working sectors [**Irastorza2010**], accounting for the second largest responsibility to disability worldwilde [**Luttmann2003**]. These are specially prevalent as upper limb or neck disorder (with 42% of all WMSDs cases reported) [**Seidel2019**] in several industry sectors, such as textile and automotive, where production processes with pre-defined motions and actions have a repetitive/cyclic nature. This has a negative impact on the risk to develop musculoskeletal disorders, with tremendous consequences to both workers and companies, leading to absenteism, early retirement and loss of productivity [**Trabalhadores**, **Varandas19**].

Several strategies have been implemented to identify, regulate and prevent occupational risk in manufacturing industries, such as (1) the inclusion of job rotation schedules, which promote a variation of the exposure throughout the working day [**jobrotation**] and (2) screening tools, for the assessment of occupational risk exposure, e.g. OCcupational Repetitive Action (OCRA), Rapid Upper Limb Assessment (RULA) or the Ergonomic Assessment WorkSheet (EAWS) [**ocra**, **rula**, **eaws**]. Nevertheless, these strategies are not optimal because they (1) are not automated, relying in observational methods and dedicated personal to inspect video records; (2) are not objective measures; (3) do not take into account differences among the worker's population, as anthropometric, age and experience variability; and (4) present single scores, being insufficient to explain the factors that contributed to this risk. With the advent of Industry 4.0, more companies are using modern strategies that follow digital solutions to provide direct and objective quantitative measures [**romero**]. An example of these incentives is the usage of wearable inertial devices for motion and posture tracking of workers.

Using inertial motion units (IMUs), time series can be collected, and relevant information can be directly measured, e.g. position and velocity of each body segment, postural angles between joints and gait parameters, making these important for ergonomics studies Caputo2019, Hang19. There are some limitations of using IMUs, mostly related with the long term bias (sensor drifting) arising from long acquisitions and the empirical

process to fine tune sensor fusion techniques. Other systems can be used for motion capture, such as camera-based methods, but these rely in fixed setup of cameras, which is unmanageable in real industrial scenarios [**sara**].

SE CALHAR, INCLUIR NA MOTIVACAO QUE EM ALGUMAS EXPERIENCIAS VERIFICAMOS DIFERENCAS ENTRE GRUPO ANTROPOMETRICOS

The usage of time series in this context can play an important role in supporting the decision of ergonomists and other professionals of the industry. In order to develop systems that can use motion and postural data for direct risk assessment and reporting, several challenges arise in the time series data mining domain. For instance, considering the periodic nature of most manufacturing tasks, risk factors are calculated by working cycle. Therefore, methods should be developed to identify working cycles with some variability in their periodicity. In addition, real occupational scenario might have interruptions or changes in the working behavior, due to abrupt production stoppage, shift breaks or even changing to another workspace that has a different motion pattern.

Other questions also arise by ergonomists, such as "*can we find a pattern that has a sharp rise in the IMU from the arm?*" or "*when the worker is using a hand tool to make screwing, can we see a periodic pattern on the IMU from the hand?*, which represent specific patterns with a descriptive shape that can be seen on the signals and are specific of a task. These events can be relevant to study their precise impact on the worker's occupational exposure. Having ways to detect these patterns is of great relevance as well. In this study, we will show how the proposed solutions can have an impact in these problems, and how they contribute to provide relevant visual feedback for information retrieval from the occupational data and make the search of specific patterns more intuitive and expressive, even for non-experienced data analysts, such as ergonomists.

1.4 Research Questions

The previous sections introduced our main motivations related with the development of methods for information retrieval, provided context regarding the grammar of time framework and how the proposed solutions can have a significant contributions in occupational health assessment.

This project addresses all the range of topics of time series, from the moment data is acquired (*sensing*), processed for information retrieval (*analysis*) and how it is used to act upon (*decision making*). The main objectives are related with the development of methods for information retrieval (*analysis*) from time series for better decision making.

1. **Sensing** - Explore in depth the available technology to measure motion and postural variables in occupational scenarios for risk assessment. This will take into account which variables are associated with a risk, based on ergonomic standards. These measures are returned as time series, which are processed in the topic *analyzis*;

2. **Analyzis** - In this topic, several research paths are explored. **A -** study (1) how to perform structural information retrieval in time series for segmentation based on change points and periodic points and (2) how are the segments related. For this, we applied a feature-based transformation of the time series and similarity based measures to make a meaningful visual representation, from which the segmentation points can be extracted. **B -** explore symbolic representations of time series, studying how these can be used for more expressive and intuitive pattern search with the help of regular expressions and ultimately natural language. **C -** From the textual representation of time series, study if we can make a higher leveled distance measure, following standard text mining methods. The resulting outputs of these methods can be used to get relevant information to take better decisions, namely in the occupational domain;

3. **Decision Making -** Study meaningful summarization techniques and explore several real-life examples in how the developed methods can help analysts be more aware of the data and move towards a more *democratized* usage of data mining tools for information retrieval in time series.

With this work, we intend to contribute to the state of the art in time series data mining with tools that provide more meaningful representations of time series, from which information can be retrieved with more meaning and at a higher level of abstraction, closer to the human intuition and visual interpretative abilities. This contributes towards more expressive methods and a democratization of these tools to accelerate the analysis process by experts in data mining and make non-experts capable of making high-level analysis.

1.5 Thesis Structure

This thesis provides a detailed description and explanation of the research work developed during the PhD program. It is organized in blablabla...

Figure X illustrates a guideline of the structure of this work, with a short description of each Chapter's content and description.

Chapter 1 introduced the main motivations, goals and context for the development of this thesis. Chapter 2 introduces theoretical concepts necessary to have a complete understanding of the work developed. It covers an introduction to motion and postural sensors used in occupational settings, time series, standard methods for its representation and analysis and text mining concepts. Chapter 3 presents the most recent works related with what we developed, namely in the topics of segmentation, summarization, pattern/event search and dictionnary based classification. On Chapter 4 we start describing the data we used, explaining its source for both acquired data and publicly available, for what it was used and how. It includes a detailed description of the protocol used to acquired workers' motion data in real industrial scenarios. The algorithm developed for time series

structural information retrieval is explained in Chapter 5, while Chapter 6 covers the symbolic representation of time series. In this chapter is explained the exploratory path to use this novel representation in query search and classification tasks. Chapter 7 shows the application of the previous methods to an exhaustive set of examples, namely from the occupational scenario, and major results are presented. In addition, this chapter also provide a general discussion about the usage of these methods for decision making. Finally, Chapter 8 gives an overall remark over the outcomes of this thesis and a reflection over the contributions that the developed methods have in making time series preparation and data mining more expressive, quicker and more practicle for an ever increasing number of data available. Each chapter will have a short introduction to situate and contextualize the reader.

2

Sensing the Physical World

2.1 Sensing the Physical World

- Bring an overview of the sensors used for a general set of things - motion, physiological, etc...

- These measure the physical world and retrieve physical changes in what they measure

- These changes can be related with something meaningful and relevant that occurred and can be seen on the data

- In this work, we apply the methods in all kinds of scenarios to show their agnosticism to domains, but focus our attention to a specific context where introducing sensing technology can have a strong impact.

2.2 Occupational Variables in the Industry

- Occupational variables that affect worker's health have long been studied and already defined in several screening tools from standard ergonomic guidelines. We can name EAWS, OCRA, RULA, etc...

- These worksheets are a reference for ergonomists in identifying the variables of interest to measure the risk of each activity performed by a worker.

- The multiple set of actions of a workstation can be analyzed

- Typically, these variables are related with motion and posture of body segments. Several scenarios are studied and variables extracted are frequency, intensity and duration of activity. Study specific activities, measure the risk based on vibration from machine or tools (https://www.cdc.gov/niosh/topics/ergonomics/ergoprimer/default.html)

- All these variables, being related with motion, should be studied

definition 1 - workstation

definition 3 - intensity, duration and frequency

definition 2 - vibration

- hand - is a special case. We should measure the vibration on the grip

2.3 Sensing Worker's Health

The type of variables that are required to perform a risk assessment are related with motion, posture and vibration. These are physical variables that can be quantified by means of inertial sensors, such as accelerometer, gyroscope and magnetometer. These three sensors are used together to compensate limitations of each other in error accumulation from sensor drifting.

- With these sensors, we can measure the orientation of body segments in terns of other body segments or standard body planes (sagital or frontal plane).

3

Time Series Fundamentals

The content of this thesis is diverse and covers several different topics. Therefore, the reader will appreciate that we set the foundations that are necessary to fully capture the essence of this work. For this, we provide an introduction to each of the topics addressed, the global definitions and used notation in this work. We start by explaining occupational domain variables and corresponding sensors used to monitor these. The data of interest in this work is *time series* and the global definitions and notations are provided. Standard pre-processing methods, representation forms and distance measures are also explained. In this chapter, only global definitions will be made. Each further chapter will have additional and more contextualized definitions when needed.

3.1 Global Definitions

The information gathered by sensors are physical quantities that vary with time. These are called *time series* and are the main topic of this work.

Definition 1 - Time Series (T): A time series is a sequence of real values ordered in time with length $n \in \mathbb{N}$: $T = (t_1, t_2, ..., t_n)$. Several domains of data rely in the acquisition of multiple time series from multiple axis of the same sensor (e.g. the 3-axis accelerometer) or from multiple sources (e.g. IMU as a fusion of three different sensors), creating a *multi-dimensional time series*.

Definition 2 - Multi-Dimensional T (MT): A MT is a set of $k \in \mathbb{N}$ time series belonging to the same acquisition: $\{T_1, T_2, ..., T_k\}$.

Segments of interest are often searched inside a *time series*. A segment is called a *subsequence*:

Definition 3 - Subsequence (sT): A *subsequence* is a segment of the time series with size $w \in \mathbb{N}$ and starting from a given position i and ending at position $i+w$ from the T or MT. A *subsequence* is delimited by two instants in time. This sample that segments a *subsequence* can be considered an *event*.

Definition 4 - Event (E): Following the definitions of [**event_def1**, **event_def2**], which state that "*an event is a dynamic phenomenon whose behavior changes enough over time to be considered a qualitatively significant change*" and "characterized by an interval of measurements that differs significantly from some underlying baseline", we consider that an *event* is an instant in time e that indicates the presence of a relevant occurrence in the time series. Multiple *events* segment the time series into several *subsequences* of different lengths. Therefore, *event* detection is often considered time series segmentation [**cpd_alan**].

A common strategy used in time series data mining to find relevant *subsequences* or *events* is the moving window.

Definition 5 - Moving Window (MW): A *moving window* is a process of sliding along a time series T to apply a specific method on each *subsequence* it hovers. The window has, such as the *subsequence* a predefined size $w \in \mathbb{N}$, which starts at a given position i and ends at position $i+w$. The process is iterative and can be made overlapping windows or not. The next window will start at $i+o$, being o the overlapping size.

With a *MW*, each *subsequence* can be filtered, features can be extracted or distances can be measured. We will show several utilities of this technique further when introducing methods used to pre-process a raw time series and standard distance measures.

Depending on the context and which conditions the data is gathered, the raw information can contain disturbances or should be transformed into another dimension to extract what matters. The set of tasks taken to prepare the *time series* to enhance information retrieval is called *pre-processing*. The pre-processing steps we will discuss involve filtering, normalization and transformation.

After preparing the data, information retrieval techniques are employed, which typically rely on distance measures. In that sense, the standard distance measures are explained. These distances will not only be performed on the numerical domain, but also on the text domain. Therefore, an introduction to the *textual abstraction* of time series will be made in this section as well.

3.2 Filtering

Time series have multiple sources of disturbance. This disturbance is usually called *noise* and is defined as an unwanted form of energy, but it can have multiple interpretations. It can be caused by internal sources inside a device, such as *white noise*, or be due to external sources, such as motion artifacts, wandering baseline, sensor detachment or the magnetic field from surrounding devices []. Any of these disturbances will affect the analysis stage and should be detected or removed.

3.2.1 Spectral Filtering

Several methods can be used to reduce the influence of noise in the analysis. Standard filtering methods, such as low-pass, band-pass and high-pass filters can be used to reduce

the presence of specific frequency bandwidths that are not relevant. There are many configurations for these types of filters, being one commonly used the *Butterworth* filter.

3.2.2 Smoothing

Another often used method that has the purpose of reducing the presence of noise and represents a variation of a low-pass filter is the smoothing technique. Several variations of this technique exist, being the simplest one a moving average, which uses a moving window, calculating the mean in each iteration.

3.2.3 Wandering Baseline

Another type of disturbance on the data that is usually removed is a wandering baseline. An example typically occurs in ECG signals, where the respiration creates a wandering baseline on the signal. This type of disturbance has a very low frequency compared to the meaningful information on the data and can be removed by subtracting a *smoothed* version of the original data, or applying a high pass filter.

3.3 Normalization

Normalization of data is an important step in any data mining process. It is essential for data uniformization and scaling, while keeping the morphology and shape of the time series. Several methods can be used for this purpose, namely:

$$\overline{T} = \frac{T}{max(|T|)} \tag{3.1}$$

the normalized signal ($\overline{T}$) is scaled by the absolute maximum of T. It is the simplest approach to normalization and guarantees that values are scaled linearly and their modulus cannot be higher than 1.

A variation of this process is the normalization by the range of amplitudes, which is as follows:

$$\overline{T} = \frac{T - min(T)}{max(T) - min(T)} \tag{3.2}$$

here the signal T is normalized to range between [0,1]. Another normalization method, called *z-normalization*, is very commonly used and relies on the distribution of its values:

$$\overline{T} = \frac{T - \mu_T}{\sigma_T} \tag{3.3}$$

where the time series T is subtracted by its mean, μ_T and scaled by its standard deviation, σ_T. The resulting values represent how many standard deviations the signal is away from the mean.

3.4 Transformation

In information retrieval, data has often to be re-scaled, simplified, approximate or represented into another data type. Each can contribute in their own way to capture the most relevant and meaningful information, or discover a new type of information that once was hidden in the original data. Dozens of methods exist for time series representation, such as Singular Value Decomposition (SVD) or wavelet transform, but only the ones relevant for this thesis will be explained.

3.4.1 Spectral Transformation

One of the first and most well known techniques suggested for time series transformation was the Discrete Fourier Transform (DFT) fourier. The idea behind this concept is that any signal, of any complexity, is a decomposition of a finite number of sine waves. Each wave is represented by a complex number, known as the Fourier coefficient, transforming the signal from the time domain to the frequency domain [**fourier2**]. This transformation allows to see the signal in a different manner, highlighting which frequencies concentrate more or less energy. It unveils the presence of specific types of noise or artifacts, or periodic shapes. Figure **??** shows the transformation of a signal into the frequency domain.

3.4.2 Feature-based Representation

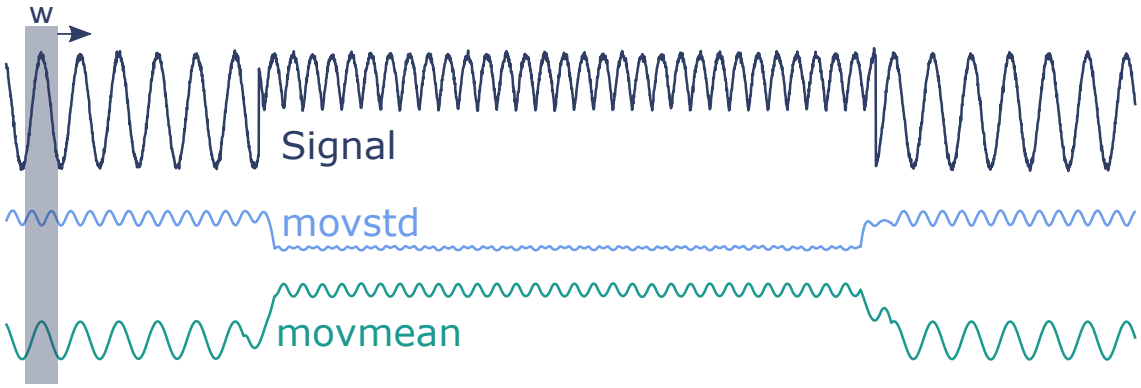


Figure 3.1: Moving window used to extract features with total overlap. The mean and standard deviation are extracted from the signal.

Frequency properties are very relevant to characterize a time series, but others can also be used to get a full characterization of the signal. The process of feature extraction is also a transformation method commonly employed. It is performed by a moving window from which features are extracted. For each feature, f, a feature series is computed.

Definition 5 - Feature Series (F): A *feature series*, F, is a feature representation of a time series with size m that depends on the overlap size $o \in \mathbb{N}$ of the sliding process, making

the size of the resulting feature series $m = \frac{n}{w-o}$. Considering the existence of a MT, the *feature series* becomes a *multi feature series* of stacked *feature series*, with size $f_{k,m}$.

When extracting more than one feature, these are grouped into a *feature matrix*.

Definition 6 - Feature Matrix (F_M): A *feature matrix*, F_M, is the set of r features extracted for k time series, with size $r \times (k \times M)$.

On Figure **??** is showed a time series from which the average (movmean) and standard deviation (movstd) are computed with a moving widow of size w=100.

3.4.3 Piecewise Aggregate Approximation

Another common used transformation method to simplify a time series and reduce its dimension is the piecewise aggregate approximation(PAA) [**paa**]. The new representation space will have size $1 < N \leq n$, in which N is a factor of the original size n. The searches to keep the average of the N equi-sized subsequences in which the original signal with length n is segmented, which results in $\overline{T} = \overline{t_1}, \overline{t_2}, ..., \overline{t_N}$, such that [**paa**]

$$\overline{t_i} = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} t_j \tag{3.4}$$

An example is showed in Figure **??**, where a ?? is converted to ?? with sizes 2 and 20, respectively.

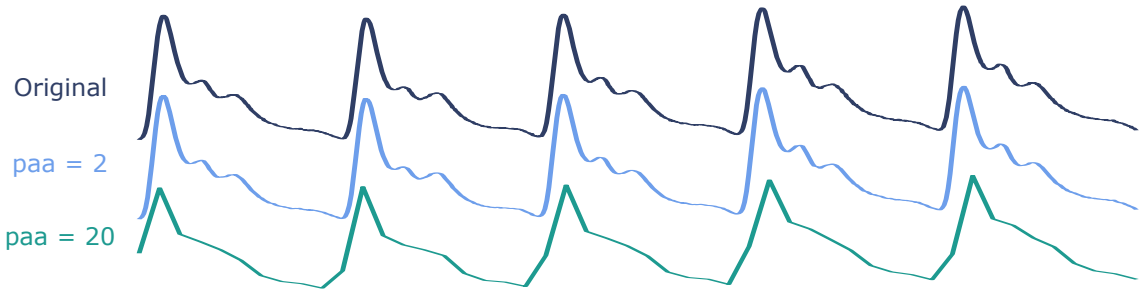


Figure 3.2: PAA representation of a ?? signal, with window sizes of 2 and 20, repsectively.

3.4.4 Symbolic Aggregate Approximation

From this method, a new representation technique was born, transforming the signal from the numerical to the symbolic domain. It is called Symbolic Aggregate approXimation (SAX). This method applies PAA to a z-normalized time series and indexes a *character* to each sample of the simplified signal based on the distribution of its amplitude values. The signal's amplitude values are separated in bins with equal probability. The number of bins is equal to the size of the *alphabet* chosen. Figure **??** shows an example of the signal

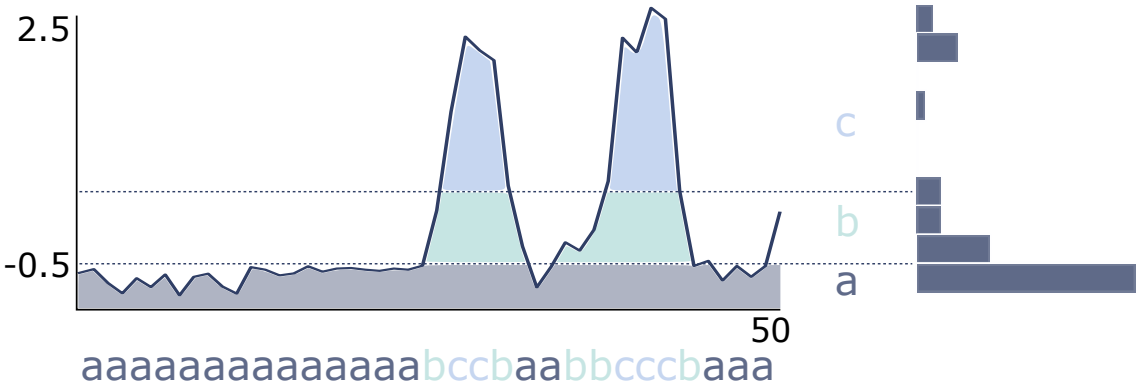


Figure 3.3: SAX representation of a power consumption signal from a Dutch Company, with window bin size of 3.

transformed into a string with 3 letters in its alphabet. Such as the DFT, SAX opens doors to analyze time series in a completely different manner, profiting from the much acquired knowledge in text mining.

In this thesis we will use feature vectors for several purposes. We also propose a novel symbolic representation technique for time series that is used for expressive pattern search and classification. In order to perform search or classification, we have to be able to calculate the difference/similarity between two time series or *subsequences*.

3.5 Distance Measures

There is an exhaustive number of distance measure for time series, but two of the classical standard measures still provide state-of-the art results in most time series data mining tasks, namely the euclidean distance (ED) and the dynamic time warping (DTW).

3.5.1 Euclidean Distance

The ED is the most straightforward distance measure for time series. Let us consider two time series, Q and C, of length n, so that

$Q = q_1, q_2, ..., q_i, ..., q_n$
$C = c_1, c_2, ..., c_i, ..., c_n$

The distance between these two time series under the ED is:

$$ED(Q, C) = \sqrt{\Sigma_{i=1}^{n}(q_i - c_i)^2} \tag{3.5}$$

which represents the square root of the sum of the squared amplitude differences between the samples of each signal. Although the distance measure is simple to compute, it is highly susceptible to typical distortions on time series. When using ED, these

distortions must be removed, otherwise, other methods, invariant to these distortions, should be used. Examples of distortions are the amplitude and offset distortion, phase distortion, and local scaling ("warping") distortion. The first can be compensated by the z-normalized ED:

$$z\_ED(Q,C) = \sqrt{2m(1 - \frac{\Sigma_{i=1}^{m} Q_i C_i - m\mu_Q\mu_C}{m\sigma_Q\sigma_C})} \tag{3.6}$$

where μ_Q and μ_C are the mean of the time series pair and σ_Q and σ_C are the standard deviation.

The *warping* distortion can be solved with an elastic measure. For this purpose, DTW is typically used.

3.5.2 Dynamic Time Warping

The DTW distance measures the alignment between two time series. Let us consider two time series, Q and C, of length n and m, respectively:

$Q = q_1, q_2, ..., q_i, ..., q_n$
$C = c_1, c_2, ..., c_j, ..., c_m$

The alignment is measured by means of a distance matrix with size n-by-m, where the (i^{th}, j^{th}) cell of the matrix contains the $d(q_i, c_j)$ between the two points q_i and c_j, being $d = (q_i - c_j)^2$ [**dtw**]. Figure **??** shows an example of a distance matrix between two time series. The matrix fully describes the difference between the two time series and maps where these align. The mapping is made by a warping path, W, that represent the set of matrix cells that minimize the warping cost, also defined as the cumulative distance of these cells [**dtw**]

$$W = w_1, w_2, ..., w_k, ..., w_K; \qquad max(m,n) \leq K < m + n + 1 \tag{3.7}$$

$$DTW(Q,C) = min\sqrt{\Sigma_{k=1}^{K} w_k} \tag{3.8}$$

The cumulative distance $\gamma(i,j)$ is calculated as $d(q_i, c_j)$ of the current cell added to the minimum distance adjacent to that cell:

$$\gamma(i,j) = d(q_i, c_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \tag{3.9}$$

When two time series with the same length have a linear warping path, such that $w_k = (i,j)_k, i = j = k$, we have a special case of the ED. DTW has a time and space complexity of $O(nm)$ while the ED has linear complexity ($O(n)$).

Figure **??** shows an example of applying the ?? and ?? on two different PQRS complexes from different ??s.

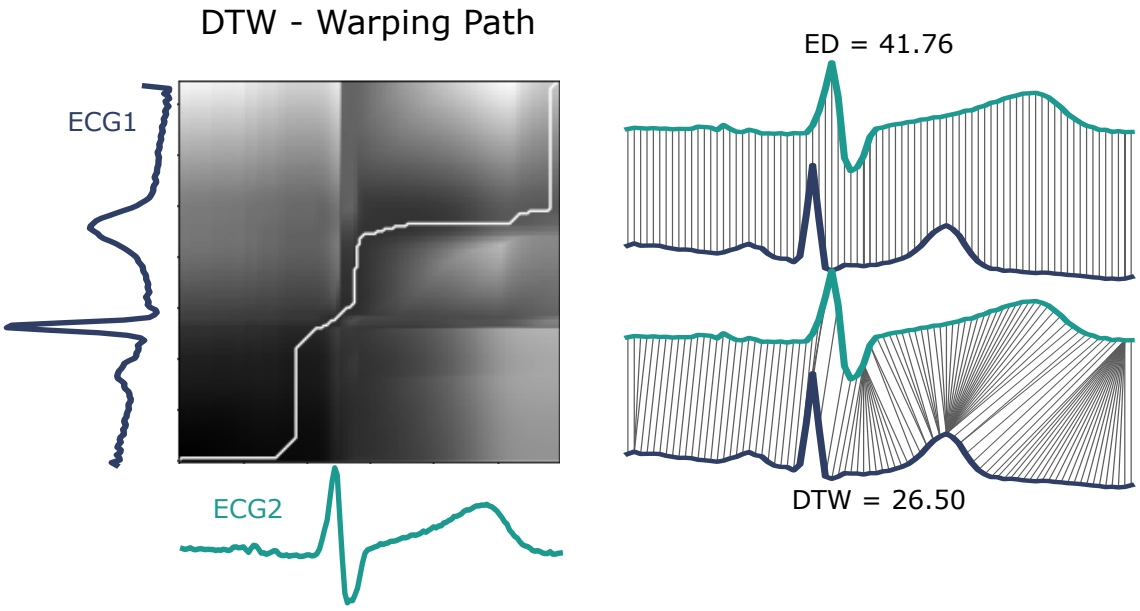


Figure 3.4: DTW and ED distances on two different ECG signals.

3.5.3 Complexity Invariant Distance

A different type of distance measure is also used to cope with complexity invariance. This distance uses a complexity correction factor (CF) with an existing distance measure, such as ED [**complexity**]:

$$CD(Q,C) = ED(Q,C)(Q,C) \tag{3.10}$$

The CF is defined as [**complexity**]:

$$CF = \frac{max\{CE(Q), CE(C)\}}{min\{CE(Q), CE(C)\}} \tag{3.11}$$

where CE represents the complexity estimate of a time series. This estimate is calculated based on the intuition that if we could "stretch" a time series until it becomes a straight line, this line would be as long as the complexity of the signal. It can be computed as the sum of the $n-th$ discrete differences along the time series[**complexity**]:

$$CE(Q) = \sqrt{\Sigma_{i=1}^{n-1}(q_i - q_{i+1})^2} \tag{3.12}$$

These distance measures are performed on the original representation domain of time series. As we showed above, other representation techniques can be employed, creating opportunities for other types of approaches. In this work, we explore other representation techniques to create novel ways of exploring time series. Then, we find that the reader will appreciate that we describe other distance measures employed, namely in the feature-based domain.

3.5.4 Feature-based Distance

As mentioned, a feature series F can be computed from the original time series to represent it based on a specific feature. If the size of the *moving window* is equal to the size of the time series, than F is represented by a single value. Otherwise, each *subsequence* highlighted by the *moving window* is characterized by the feature and the F is computed as an array. When multiple features are extracted, each *subsequence* is characterized by a set of features, creating a feature vector $\vec{f}$ with r feature values. Vector based distance measures can be used with these feature vectors to compare different time series or *subsequences*. There are several vector-based distance measures, including the already mentioned euclidean distance or the manhattan distance, but we will only describe the cosine similarity/distance.

The cosine similarity is a measure of the angle between two vectors determining if these are pointing in the same direction. Consider two feature vectors $\vec{f_A}$ and $\vec{f_B}$. Their cosine similarity is computed as their normalized dot product [**cosine**]

$$CS = \frac{\vec{f_A} \cdot \vec{f_B}}{||\vec{f_A}||||\vec{f_B}||} \tag{3.13}$$

being $||\vec{f_A}||$ and $||\vec{f_B}||$ the euclidean norm of each feature vector, defined as $\sqrt{\Sigma_{i=1}^{r} f_{Ai}}$ and $\sqrt{\Sigma_{i=1}^{r} f_{Bi}}$, respectively [**cosine**].

3.6 Applying Distance Measures

Measuring distances between any time series gives the ability to compare them. It is the fundamental instrument for most time series data mining tasks. With a distance measure, we are able to compare groups of time series for classification purposes or compare *subsequences* with a query template and find if it occurs in the time series. Another relevant application of distance measures is its usefulness to retrieve relevant structural information of a time series by comparing each of its *subsequences* to all other *subsequences*. In this subsection, relevant methods applied with the help of the presented distance measures are explained to retrieve information from a time series. We will start with distance/similarity matrices.

3.6.1 Self-Distance Matrices

A time series can reveal relevant information when each *subsequence* is compared to all the other *subsequences* of the same time series. The result is a pairwise distance matrix that unveils *homogeneity*, *repetition* and *novelty* on the time series. Each are relevant assets for segmentation and summarization tasks.

Let X be a sequence with size N that can be a time series or a representation of a time series in the *PAA* or feature space, such that $X = (x_1, x_2, ..., x_i, ..., x_j, ..., x_N)$. Each element

of X can either be a single value or a vector with r features. Independently of that, a matrix S with size $N \times N$ can be computed, such that:

$$S(i, j) = d(x_i, x_j) \tag{3.14}$$

being d a distance measure between elements $x_i, x_j \in X$ for $i, j \in [1 : N]$. $S(i, j)$ represents a cell of S that contains the distance value. When $i = j$ the distance should be zero, therefore the diagonal of S has the lower values. Besides the main diagonal, other relevant structures can be found in S. These include *homogeneous blocks* and *paths* [**fmp**, **muller**].

Areas with lower distance are highlighted as *homogeneous* structures. These give an indication of *homogeneity* and *novelty*. *Homogeneity* because a *block* along the diagonal means that the time series has a constant behavior during the segment delimited by the *block*. *Novelty* because when S has multiple *blocks* along the diagonal, it shows that the time series shifted its behavior/regime. The moment there is a transition between *blocks* is a potential segmentation point.

When the time series has repeating *subsequences*, *paths* show up on S. The reason for it can be illustrated with the mentioned ?? measure. With ??, the z-normalized euclidean distance matrix between two time series is computed and the optimal path is computed as the final cumulative distance. This *path* is a perfect diagonal if the time series are exactly the same, but can be slightly distorted if these are slightly different. The same type of *paths* appear in S indicating a low distance between two different *subsequences* of the time series.

3.6.2 Matrix Profile

As mentioned from the previous subsection, measuring all the distance pairs of a time series provides the ability to retrieve relevant structural information. Recently, a strategy was proposed to compute a one dimensional distance profile for a time series based on a z-normalized euclidean distance matrix. By keeping the *nearest neighbor* of each *subsequence*, we retrieve the *matrix profile*. The result gives the minimal distance pair of each *subsequence*, meaning that minimum values are *motifs* and maximum values are *discords*.

3.6.3 Template-based Search

The presented distances can also be used to retrieve a distance profile from a *template*. This type of mechanism belongs to the class of query-based search problems. The process to compute this distance profile involves sliding the template along the signal and applying a distance measure to each iteration. The result should indicate which *subsequences* are more (dis)similar to the used template.

In this thesis, we will propose other methods to perform query-based searches on a time series. Instead of using a *subsequence* template, we will be using text patterns. REVER

3.6.4 Classification

- show dendogram - explain 1-euclidean nn

3.7 Text Mining on Time Series

The representation of a time series into the symbolic domain makes possible the usage of text mining methods to analyze time series. The reader will appreciate that an association regarding time series and text is made, and are also introduced relevant methods from the text mining domain, used in this work.

3.7.1 Time Series Textual Abstraction

In *SAX*, the signal is transformed into a sequence of symbols. For this, each sample of the *PAA* representation is converted into a character, which can then form *words* and *sentences*. Other symbolic representation techniques exist on the literature and this work proposes a novel symbolic representation. In that sense, it is relevant to give the general background that makes this association between the original time series, a symbolic time series and text notation. Note that this is an introductory explanation that will be further contextualized when needed throughout this thesis on the corresponding sections.

Definition 8 - Character (Char): A *character* is an unit symbolic element that represents a sample or *subsequence* of a time series. Each sample of a time series is transformed into a *character* to form a *symbolic time series*. Sequences of characters form *words*.

Definition 9 - Symbolic Time Series (ST): A *symbolic time series* is a sequence of *characters* ordered in time with length $n \in \mathbb{N}$: $ST = (st_1, st_2, ..., st_n)$. A specific sequence of *characters* of a ST can form a *word*.

Definition 9 - Word (W): A *word* is the concatenation of a sequence of *characters*, giving a textual representation of a *subsequence*. Putting *words* together forms a *sentence*.

Definition 10 - Sentence (S): A *sentence* represents a group of *subsequences*. It is formed by joining sequences of symbolic *words*.

Definition 11 - Document (D): The set of *sentences* in a time series are called a *document*. It represents the entire time series.

Definition 12 - Corpus (GD): The *corpus* is a collection of text material (group of documents). It represents the higher level of textual information. This collection is typically annotated and used for machine learning tasks. In this case, a corpus will be represented by the set of documents that describe a time series dataset.

Definition 13 - Vocabulary (V): The *vocabulary* comprehends the set of all different words present in all time series.

3.7.2 Text Features

From the textual representation, text mining methods can now be applied. Here are introduced traditional methods applied for feature extraction of text data, namely the ?? and ??.

Definition 14 - Bag of Words (Bow): A BoW is a feature matrix representation of a corpus, being the feature the number of occurrences of each *term*, called the term-frequency (*tf*):

$$bow(t, d) = tf_{t,d} = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{3.15}$$

being t the term that exists in a document, d the document and t' the term that belongs to document d. Here t can be a single *word* or an *n-grams*.

Definition 15 - N-grams: It is a span of followed *words* that are counted in the ??/??. Possible *2-gram* from Figure **??** would be `aa`, `ca` or `ac`. This strategy resembles a *moving window* on time series, but for text. It makes the ??/?? model more robust since it makes it rely in more than single *word* statistics. Regarding time series, this method is relevant because it takes into account time dependencies between *words*, which reflects the time dependency seen in time series between *subsequences*.

The ?? is commonly used to vectorize the textual representation of each symbolic time series, but there is common knowledge in the text mining community that if a *term* occurs in all *documents* it is less relevant. To counteract this limitation, the ?? matrix is used.

Definition 15 - Term Frequency Inverse Document Frequency (TF-idf): The *Tf-idf* matrix increases the relevance of t by means of the t_f, while reducing its importance in proportion to the number of *documents*, d that contain the term t. The model is defined by being a ratio between the t_f and the *inverse document frequency* (idf), which is calculated as follows:

$$idf(t, D) = \log \frac{L}{|\{d \in D : t \in d|\}} \tag{3.16}$$

L is the total number of documents ($L = |D|$). The final equation of the *tfidf* model is the following:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{3.17}$$

Both *bow* and ?? are matrices that have a vector representation of each *document*, where each element of the vector is the relevance of the *term*. That means that the cosine distance can be used to compute the difference between *documents*.

Due to the probabilistic nature of the ?? and the fact that it contains discrete features, it is suitable to use in naive bayes classifiers. In the other end, the ?? is typically used with linear support vector machine (SVM) classifiers.

3.7.3 Text Pattern Search

When writing or reading a document, we often have to use the *cntrl+F* key to search for specific words or expressions. This option let us search for direct matches, characters that belong to a word or expression, or even, use ??.

The last method is a parsing technique that is convenient to write text patterns, being more flexible than direct matches. It is based on regular languages, following a specific set of rules, and contains a set of meta-characters.

In order to understand the way that regular expressions work, some of the most used characters and ?? primitives are presented as follows [**regex2**]:

MC	Description	Example of Match
*	The preceding item will be matched zero or more times	*eve*nt* → [evnt, event, eveent]
+	The preceding item will be matched one or more times	*su+b* → [sub, suub, suuub]
?	The preceding item is optional and will be matched, at most, once	*team?*→ [tea, team]
.	Matches any character	$s.m \rightarrow [ssm, sam, sim]$
[]	Matches anything inside the brackets	$wom[ae]n \rightarrow [women, woman]$
|, &	Boolean operators - or, and	*tr(i|a)p* → [trip, trap]
(?=<)	Positive lookbehind - The string matches the item that is preceded by the pattern inside the lookbehind without making it part of the match	*(?=<http://)* → any URL
(?<!)	Negative lookbehind - The string matches the item that is not preceded by the pattern inside the lookbehind	*?<!*.+ → [~~10th~~, th]
(?=)	Positive lookahead - The string matches the preceding item that is followed by the pattern inside the lookahead without making it part of the match	*(?=www\.)*→ *matches web protocol*
(?!)	Negative lookahead - The string matches the preceding item that is not followed by the pattern inside the lookahead	a(?!b) → [~~ab~~, ac]

3.7.4 Keyword Extraction

https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

3.8 Performance and Validation Measures

In this thesis are mainly discussed three time series data mining domains, namely classification, segmentation and event detection. In order to perform a validation of the work developed, standard procedures are already available. In this section are explained which procedure are typically used to evaluate the performance of algorithms used in these domains.

3.8.1 Classification Problems

(FALTA A ACCURACY)

One of the most common strategies to evaluate algorithms from the machine learning field are *precision* (P), *recall* (R) and *f1-score* (F1). These measures are calculated based on *true positives* (TP), *false positives* (FP) and *false negatives* (FN). Understanding what these metrics represent depend on the problem. These measures are used in this thesis to evaluate the performance of classification and event detection algorithms.

$$P = \frac{TP}{TP + FP} \tag{3.18}$$

$$R = \frac{TP}{TP + FN} \tag{3.19}$$

$$F1 = 2 \times \frac{P * F}{P + R} \tag{3.20}$$

In terms of time series classification problems, a labeled dataset with training and testing time series is typically used. The algorithm is trained on the training set and validated on the testing set. In order to perform the validation of the algorithm, the ground truth labels are compared with the labels predicted by the algorithm. This comparison gives the number of TP, FP and FN.

- TP_c - If the label predicted by the algorithm is equal to the target class (positive when positive);
- FP_c - If the label predicted by the algorithm is falsely classified as the target class when it should not (positive when negative);
- FN_c - If the label predicted by the algorithm is not labeled as the target class when it should (negative when positive).

These measures were initially performed in binary classification problems, but can be adapted for multi-classification ones. For this, each class (the target class) is compared to all the other class, being the target class the *positive* and all the other classes *negative*. All measures are calculated for each class. Finally, a macro and micro average of these metrics can be calculated.

$$macroP = \Sigma_{i=0}^{c} \frac{P_c}{c} \tag{3.21}$$

$$macroR = \Sigma_{i=0}^{c} \frac{R_c}{c} \tag{3.22}$$

$$microP = \frac{\Sigma_{i=0}^{c} TP_c}{\Sigma_{i=0}^{c} TP_c + \Sigma_{i=0}^{c} FP_c} \tag{3.23}$$

$$microR = \frac{\Sigma_{i=0}^{c} TP_c}{\Sigma_{i=0}^{c} TP_c + \Sigma_{i=0}^{c} FN_c} \tag{3.24}$$

These metrics are typically visualized on a *confusion matrix*, which displays a 2D-map of the ground truth labels VS predicted labels.

3.8.2 Event Detection

Regarding event detection problems, the process involves finding the sample that corresponds to the ground truth event. Considering that it would not be fair to calculate the performance of an event detection algorithm by searching if the ground truth sample is found, we calculate the TP, FP and FN based on an error margin. From these measures, metrics from Equations **??**, **??** and **??** are calculated. The estimated events are considered one of the following categories:

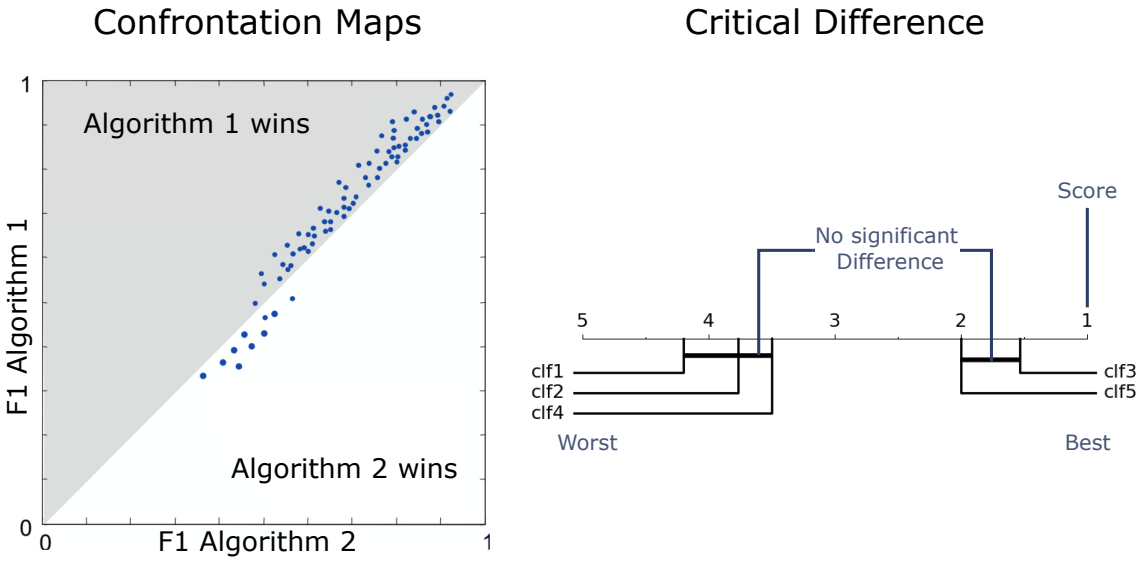
- TP_e - is counted when the estimated event is in the margin around the ground-truth event;
- FP_e - is counted whenever it is out of a margin around the ground-truth event, or when there is more than one estimated event inside the margin;
- FN_e - is counted when there is no estimated event inside the margin of the ground-truth events.

Additionally, the distance of the *TP* events from the ground-truth events can be calculated with several distance-based metrics, namely the mean-absolute-error (MAE), the mean-squared-error (MSE) and the mean-signed-error (MsE):

$$MAE = \sum_{i=1}^{k} \frac{|g_i - e_i|}{k} \tag{3.25}$$

$$MSE = \frac{1}{k} \sum_{i=1}^{k} (g_i - e_i)^2 \tag{3.26}$$

$$ME = \frac{1}{k} \sum_{i=1}^{k} (g_i - e_i) \tag{3.27}$$



The precision measure is relevant to indicate if the method is able to only estimate events that belong to the ground-truth category, while the recall measure is an important indication of how many ground-truth events are missed in the estimation of the method. Both measures are combined in the F1-measure.

The distance-based metrics evaluate how far are the TP from the corresponding ground-truth events (MAE and MSE) and which is the direction of estimation of events (if before or after the ground-truth events - ME).

3.8.3 Comparing Algorithms

In classification problems, a good procedure is to compare the proposed algorithm with existing state of the art solutions, so that the reader can understand how the results presented are unbiased from the data. In this work, for each strategy proposed in the domains of classification and event detection, we will compare it with existing solutions. There are two typical ways of displaying this comparison: *confrontation maps* and *critical difference maps*, with examples displayed on Figures **??**.left and **??**.right, respectively.

3.8.3.1 Confrontation Maps

When the proposed algorithm is applied to a dataset and compared with another algorithm, we might be tempted to display an overwhelming quantity of information in a table. Although this is a valid approach that should be made to give a full picture to the reader, there should also be a straightforward way of displaying the same information but that can be read in a glance. With confrontation maps we can compare the performance of two algorithms just to very intuitively understand if there is a significant difference in their performance. Typically, this map is a scatter plot comparing the *F1-score* or *accuracy* of algorithm 1 *VS* algorithm 2. Figure **??**.left displays an example of it taken from

[**keogh_presentation**]. Each dot of the plot is a dataset. When it is above the diagonal, it is better classified by algorithm 1, while if below, it is better classified by algorithm 2. In this example, algorithm 1 is better in most datasets.

3.8.3.2 Critical Difference

Critical difference maps are a way of comparing the performance of multiple algorithms or variations of the same algorithm. The plot is a representation of a statistical test over the performance result of each algorithm. The test evaluates if the difference in the performance is significant (critical difference) or not. For instance, on Figure **??**.right, the plot compares 5 different classifiers (*clf1* to *clf5*) and highlights that the difference in performance is not significant between *clf1, 2* and *4*, neither between *clf3 and 5*. However, *clf3* and *5* have a much better performance than the other classifiers. The closer the classifiers are from the right (1), the better they are. The bold bar connects classifier with performances that are not significantly different.

In this work, we will use an implemented critical difference method from [**critical_dif**].

4

State of the Art

State of the art in the topics mentioned previously as well as what has been done considering the advances in time series data mining.

4.1 Information Retrieval from Time Series

4.1.1 Event Detection

Most of the works available in event detection are focused in change point detection or segmentation. The found strategies are categorized based on (1) their ability to be used online or offline, (2) being univariate or multivariate, (3) based on a model or non-parametric and (4) being unsupervised or supervised [**cpd_alan**, **review_1**, **review_2**]. Regarding supervised methods, there are multi-class, binary and virtual classifiers, optimized for the purpose of detecting change points [**review_cpd_1**]. The advantage of supervised methods is to not only detect the change point, but give the nature of the change as well. Another example uses neural networks with transfer learning for segmentation [**pedromatias**]. However, supervised methods rely in very brittle training sets and class imbalance, since there are more in-state sequences than change point sequences [**review_cpd_1**]. Additionally, a problem reported by [**cpd_alan**] is that most algorithms were validating the performance of their algorithms in synthetic data, which given the nature of the application was not optimal. In that sense, a benchmark is now available for change point detection [**cpd_alan**], where methods can be compared on real-data. The proposed work uses this benchmark to compare itself with other non-supervised and offline methods.

Existing non-supervised methods include older but with state of the art performance in change point detection, such as the *Baysian Online* method (BOCPD) [**bocpd**], the *binary segmentation* (BINSEG) method [**binseg**] and the *segmentation neighborhoods* (SEGNEIGH) method [**segneigh**]. These methods have been reported successful in several domains [**cpd_alan**], however, the BOCPD only achieved good results when parameters were hypertuned, and the BINSEG and SEGNEIGH are not used in multidimensional domains. In addition, these methods are not reported to cope with a multi-time scale change

[**cpd_alan**]. An available repository provides an implementation of some of these offline methods [**review_2**], but these lack a visual output that might give the user an intuition over where a change point might be.

Another method, called FLOSS [**eamonn1**], relies in searching change points based on the nearest neighbors of subsequences, being very successful in real data domains. As it searches for nearest neighbors, the similarity between segments might be compared and used for summarization, but nothing is reported regarding multi-dimensional time series.

The ?? has been used for change point detection in the audio domain, based on a feature representation of the audio signal [**MuellerZ19_FMP_ISMIR**]. The advantage of using the ?? is the amount of information it provides for a specific time scale. In this work, we profit from these ideas applied in the audio domain, but extend its usage to other time series domains. The tool we propose can be used to detect events with context, associating the estimated events with patterns, (dis)similarities, periodicity and novelty. In addition, if being able to extract the information available in the ??, this tool can be extended to summarization tasks. Finally, although the search mechanism is based on a specific time scale, the process can be made recursive to perform multi-time scale searches recursively.

The proposed method highlights itself for being domain agnostic, work with both uni and multidimensional time series, give events with context by means of the visual information available, but also by the similarity measures in the matrix, that help in associating an event as a change or a periodic segment, and how similar are the segmented subsequences. It is unsupervised and works offline. It can be extended to work in multi-time scale problems with a special interest in time series summarization. We will demonstrate in this work how this method can bring novelty to the problematic of event detection, with a direct application to labelling and time series summarization.

The problems regarded in this work involve essentially the identification of cyclic information and anomalies. Typically, algorithms developed for these purposes may resort to (1) supervised machine learning (ML) methods, which require a certain level of annotation beforehand and (2) unsupervised methods, which are based on the similarity analysis of the signals or their features, without any prior information. Several methods found, employed in the analysis of inertial data, are used in the context of human activity recognition (HAR). The list of supervised ML methods is extensive and promising works are found to achieve this purpose. The application of neural networks [**Lara2013**], hidden Markov models [**Zhu2009**], decision trees [**Jatoba2008**], bayesian networks [**Jatoba2008**], and semi-automatic process [**duarte1**], among others, are algorithms capable of detecting and classifying various human actions. Nonetheless, most of the work done in this context only looks to identify previously defined actions like lying, standing, sitting down, move upstairs, etc., that might not be cyclic and rely on a significant amount of labelled data.

Several works that use unsupervised methods for the identification of cyclic information and anomalies are also found. The most simple method of cycle detection is the use of point references on the workplace to describe when a cycle starts and ends. Which is usually considered a system subject to flaws with a requirement for further adjustments

steps [**Bauters2014**, **Bauters2018**]. Other more reliable alternatives analyze features of the signal and search for periodic motion in those. An automated algorithm of segmentation was able to separate complex and multidimensional data into smaller segments that can be described through harmonic models. This algorithm revealed to be significantly useful to identify cyclic movement without any *a priori* knowledge of the input data, using a combination of a recursive least squares segmentation algorithm, a model fitting of damped harmonics, and in the end, a clustering analysis to classify the events [**Lu2004**, **Lu2003**]. The usage of features is of great relevance in unsupervised works, and methods are found to select adequate features for detection and classification tasks, such as in [**machado2015**]. Another example is the use of four-pass UKF (unscented Kalman filter) to produce an unified model with kinematic parameters. These may then be segmented by analyzing the parameter's zero crossing velocity and in the end uses a clustering algorithm to identify repetitive segments [**Wang2015a**].

Other methods rely on a self-similarity approach, namely [**neuza**], where cyclic information is segmented by searching for minimums, in the convolution of a segment of the signal with itself. The *Matrix Profile (MP)*, which is a method that compares all sub-sequences of a given time series with themselves through an euclidean distance, has also revealed promising results. In the end, it returns the minimum value distance for each segment, highlighting the moments of the time series which are similar within themselves [**Yeh2018**]. Additionally, autocorrelation revealed itself an useful tool, as the search over maximum values can infer the cyclic nature of the data [**Bauters2014**]. Finally, for anomaly detection in industrial scenarios, an interesting work applies an unsupervised method based on the clustering of time series segments to detect the execution of improper movements [**duarte2**].

The following work is inspired over an algorithm for the detection of musical structures on audio signals [**Foote2000**, **audiolabs1**, **audiolabs2**] by means of a *Self-Similarity Matrix* (*SSM*). This sort of analysis of self-similarity to collect information about the periodicity has also been performed over video datasets. This type of analysis usually consists on a framework where a Fourier analysis is performed on an *SSM* to characterize and highlight the periodicity of the data from the video [**Cutler2002**, **Cutler2000**, **Cutler1999**].

4.1.2 Text based Query Search

There is a large literature on time series similarity search, see [26] and the references therein. However, in most cases it is assumed that the query comes from a downstream algorithm, not a human. As such, there has been relatively little attention paid to the ability of humans to formulate meaningful queries. In principle one could do "query-by-sketching" and invite the user to draw the pattern she is interested in finding [15,16]. The recent "Qetch" system is a prominent example of this approach [15]. However, there are two possible limitations to such an approach: First, it is not clear that most people have the ability to sketch their query. For example, many people cannot even draw an accurate

circle [25]. Secondly, as Figure 1 hinted at, classic distance measures may be too literal and limited in expressiveness to retrieve the desired pattern. As a simple example, suppose that a user wishes to retrieve all highly symmetric patterns. There is simply no way to do this with Euclidean distance or similar distance measures. Other researchers have noted these issues and proposed more flexible queries languages for time series. For example, the SDL (shape definition language) of [11]- allows the user to formulate "blurred" queries. However, we believe that most such systems are not accessible for the typical user. For example, in our proposed system, a 3-point-turn can be successfully queried by noting that the surge axis will exhibit three consecutive "bumps" and formulating the query Surge: [peak peak peak]. In contrast, SDL would require: (Shape triplespeak (width ht1 ht2 ht3) (in width (in order spike (ht1 ht2 ht3) spike (ht1 ht2 ht3)))). Several similar query systems based on regular expressions or SQL-like languages have been proposed, but none seem suitable for general use [17,20]. There have also been a handful of other attempts at natural language querying for time series [6,7]. None of these works seem to have been adopted by practitioners. We feel that this is because they probably suffer from too broad an ambition, proposing completely domain independent search. While domain independence is a worthy ambition (and our eventual research goal), it is clearly challenging. Even the word "spike" can have a different meaning for neuroscientists, economists, epidemiologists, and astronomers. In this work we take advantage of the fact that driving is a familiar, even quotidian, activity for most people, and therefore a domain for which most people have strong intuitions for. Moreover, this domain has a near unique property that allows a user to model the behavior they wish to find. We found that, in many cases we could glean intuition as to how a driver's behavior would reveal itself in telemetry by simply "puppeteering" a smartphone equipped with an app that shows its acceleration and gyroscope readings. For example, by modeling a 3-point turn by sliding the smartphone on a desk, we can see that this behavior best revels itself on the surge axis as three consecutive bumps.

4.1.3 Summarization

Very few strategies are found to make compact and meaningful representation of time series. The works that can be highlighted refer to time *snippets* and time series *bitmaps* [**snippets**, **bitmap**]. The first highlights the limitation of current methods in providing a satisfactory solution to time series *summarization*. It proposes a method that is able to segment the *k* most *representative* sub sequences of a time series, and use these elements as the summary. This strategy answers several of the discussed demands aforementioned in Section **??**, namely the segmentation and similarity. Regarding the time series *bitmap* representation, the strategy is able to provide a coded bitmap with information on cluster, anomaly and other regularities on data collection. These bitmaps were used as folder icons, and also answer several of the aforementioned characteristics, such as *similarity* and *events*. An example of both strategies can be seen on Figure **??**.

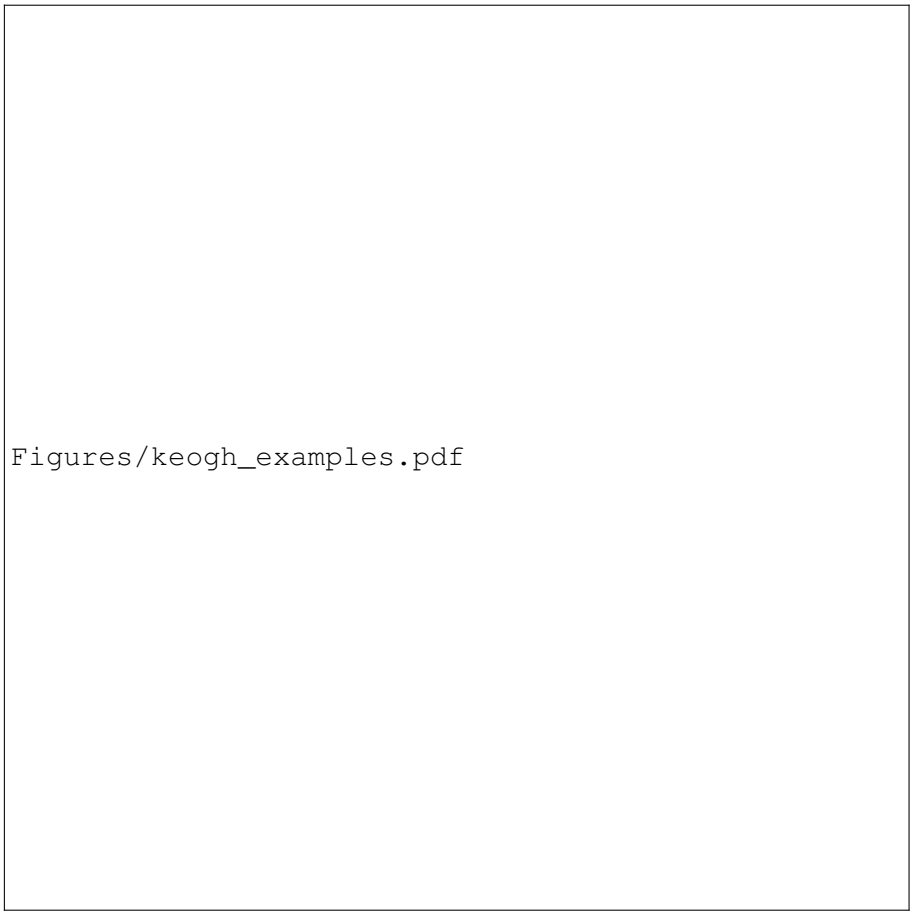


Figure 4.1: Strategies for time series summary found on the literature. These images are taken from the works from [**snippets**, **bitmap**]

Time series *shapelets* are also a method that could provide interesting results. However, the strategy is *supervised*, and the point of the proposed method is to have *no apriori* knowledge about the structure of the data, except the time scale in which the summarization is performed.

Other interesting strategies provide a transformation of time series into text and could be used for time series summarization, but are not able to suitably summarize a time series from the textual representation [**ssts**, **sax**].

Strategies that are typically used to present information in a compact way are found in several domains. In text analysis, for instance, the relationship between repeating sequences is illustrated with arc diagrams [**bitmap**, **arcplots**]. These show where repeating sequences occur in a very concise way. This has a range of applications that include, for example text and DNA sequence analysis.

One domain that has a particular relevance in data visualization is genomics. Graphical genome maps are found to concatenate a significant amount of information in a very compact way. Genome features and sequence characteristics are assessed with this visual strategy. An example can be found on Figure **??**. This visualization strategy can provide increasing circular layers of information. Although we are used to look at time series from

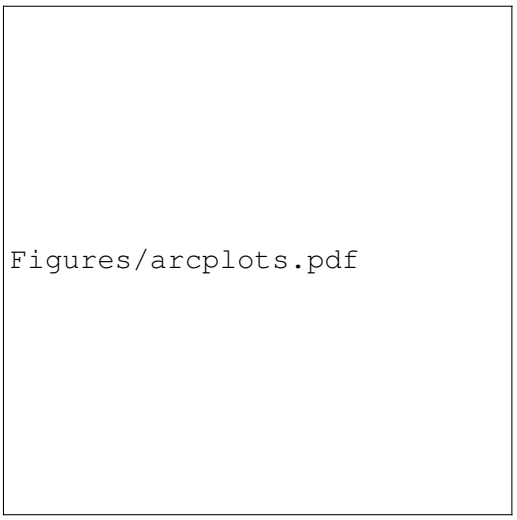
left to right, a circular representation can have benefits to concatenate the information we want to include.

In the musical domain, strategies have also been developed that summarize audio time series with segmentation techniques. One of the strategies that is common to be used involves detecting novelty instances on a similarity matrix representation of the audio signal, called *Self-Similarity Matrix* (SSM). This data structure provides a significant range of information that can be used to retrieve structural information, such as block and periodic structures [**fmp1**, **fmp2**, **audiolabs1**, **audiolabs2**]. This method will inspire our visualization strategy, which will be explained further.

4.1.4 Classification

Current available methods for time series classification are categorised as shape-based and structure-based. Existing approaches until the last decade were focused in shape-based similarity methods, while during the mid 2010's, methods that would seek the analysis of higher-level features started to be developed [**Keogh2004**].

Shape-based methods focus their attention in performing local comparisons between time series. Examples of well-known methods are the Euclidean distance (ED) or Dynamic Time Warping (DTW) [**jlin2013**]. Although both work well with short-length time series, the first has the inconvenient of needing time series with the same length, while also being sensitive to time misalignments. The latter is able to counteract this problem by means of determining the best alignment between two time series [**Keogh2004**, **jlin2013**]. These distance measures are usually combined with a k-Nearest Neighbour (k-NN) classifier to



(a)

(b)

Figure 4.2: A - Diagram for string association. This image is taken from the works from [**arcplots**]; B - Circular plot by OmicCircos. Several layers (Circular tracks) identify genome position, expression heatmaps, correlation between expression and CNV, among other features. The image is taken from the works from *Ying Hu, et al.* [**genomics**].

solve TSC tasks. The limitations of these techniques come with problems that include the presence of noise or long time series with characteristic sub-structures [**BOSS**].

In the other end, structure-based methods rely on broader aspects of time series such as the presence of specific morphological structures or patterns, being useful to classify long and noisy time series [**BOSS**]. Dictionary based methods fit into this category and have recently been used with great success. These techniques rely in a transformation of the time series into a symbolic feature vectors by means of a specific method, such as the *Symbolic Aggregate approXimation* (SAX) [**SAX**] or the *Symbolic Fourier Approximation* (SFA) [**SFA**]. The first approach proposed for TSC with symbolic representations was the work of Jessica Lin *et. al* with the *Bag of Patterns* (*BoP*)[**jlin2013**]. Further proposed methods were conceptually inspired on the *BoP*, using the same reasoning. Techniques such as *Bag of SFA Symbols* (BOSS) and Word ExtrAction for time SEries cLassification (WEASEL), from the same authors, use a similar reasoning but employ the SFA instead [**BOSS**, **weasle**].

Using syntactic methods has already been successful for several time series data mining tasks, mostly related with query search and classification. Besides, these methods, being dictionary-based, can be used to show similarity between subsequences by looking into the distribution of word counts. However, current methods rely mostly in incomprehensible sets of characters, such as *aaa*, which are hard to associate with a specific subsequence of the time series, therefore providing limited interpretability. In this work, we propose a method that literally translates the time series into sentences, such as that if a human was to describe a time series with text, it should be possible to separate these time series with the written words. We have seen natural language being used to include the human in the loop for more intuitive and meaningful query searches in time series [**hil_naturallanguage**]. Such as with SSTS, the purpose is to increase the expressiveness. This kind of descriptive power can be used to provide more intuitive feedback and increase interpretability to understand why a time series is different than others.

There is an existing method that is capable of providing visual interpretability of differences between time series, which is a structure-based method called *shapelets* [**shapelets**]. Shapelets are representative subsequences of the time series, which characterize a specific class. The advantage of this method is the higher interpretability because relevant shapes from the class can be highlighted [**shapelets**].

All the mentioned methods are a reference in TSC tasks with innovative concepts that merge ideas from the text-mining domain into TSC domain. One of the advantages of structure-based methods that rely in a dictionary-based concept is to use the words extracted as an interpretable model to differentiate time series. The histogram of words generated gives the user an understanding of which patterns better represent the time series and give an intuition over patterns that differ between classes of time series. This provides a feedback and explanation over why a class is different than the other. However, dictionaries can be confusing, and the words generated are not intuitively associated with the patterns these represent in the time series. One method that went beyond

the previously mentioned methods in that aspect is the SAX-Vector Space Model (SAX-VSM). This method used a weighted word vector representation of the time series and showed which are the relevant words for the classification process and what patterns these represent in the time series, demonstrating that the classification process can be interpretable by measuring the importance of the patterns found for each class of signals [**sax_vsm**].

The proposed method is built upon the same ideas as the BoP method but uses the *SSTS* Tool to promote the inclusion of the human reasoning in the classification process and provide more interpretable representations, as inspired by the work of SAX-VSM.

The method has been conceptually designed focusing in providing a solution that copes with (1) enabling the human intuition in the classification process, (2) be invariant to size, (3) have awareness of the order at which structures appear on the time series, (4) be domain agnostic, (5) have a flexible pre-processing to increase the representational power and (6) increase the readability. This method brings novelty by using literal natural language sentences to perform classification of time series, which can be customized by an analyst and moves towards a more readable output on distinguishing time series both visually and with keywords.

4.1.5 Search by Query

There is a large literature on time series similarity search, see [26] and the references therein. However, in most cases it is assumed that the query comes from a downstream algorithm, not a human. As such, there has been relatively little attention paid to the ability of humans to formulate meaningful queries. In principle one could do "query-by-sketching" and invite the user to draw the pattern she is interested in finding [15,16]. The recent "Qetch" system is a prominent example of this approach [15]. However, there are two possible limitations to such an approach: First, it is not clear that most people have the ability to sketch their query. For example, many people cannot even draw an accurate circle [25]. Secondly, as Figure 1 hinted at, classic distance measures may be too literal and limited in expressiveness to retrieve the desired pattern. As a simple example, suppose that a user wishes to retrieve all highly symmetric patterns. There is simply no way to do this with Euclidean distance or similar distance measures. Other researchers have noted these issues and proposed more flexible queries languages for time series. For example, the SDL (shape definition language) of [11]- allows the user to formulate "blurred" queries. However, we believe that most such systems are not accessible for the typical user. For example, in our proposed system, a 3-point-turn can be successfully queried by noting that the surge axis will exhibit three consecutive "bumps" and formulating the query Surge: [peak peak peak]. In contrast, SDL would require: (Shape triplespeak (width ht1 ht2 ht3) (in width (in order spike (ht1 ht2 ht3) spike (ht1 ht2 ht3)))). Several similar query systems based on regular expressions or SQL-like languages have been proposed, but none seem suitable for general use [17,20]. There have also been a handful of other

attempts at natural language querying for time series [6,7]. None of these works seem to have been adopted by practitioners. We feel that this is because they probably suffer from too broad an ambition, proposing completely domain independent search. While domain independence is a worthy ambition (and our eventual research goal), it is clearly challenging. Even the word "spike" can have a different meaning for neuroscientists, economists, epidemiologists, and astronomers. In this work we take advantage of the fact that driving is a familiar, even quotidian, activity for most people, and therefore a domain for which most people have strong intuitions for. Moreover, this domain has a near unique property that allows a user to model the behavior they wish to find. We found that, in many cases we could glean intuition as to how a driver's behavior would reveal itself in telemetry by simply "puppeteering" a smartphone equipped with an app that shows its acceleration and gyroscope readings. For example, by modeling a 3-point turn by sliding the smartphone on a desk, we can see that this behavior best revels itself on the surge axis as three consecutive bumps.

4.2 Occupational Health Sensing and Problems

5

Data Description and Management

Explain our sources of data to explore the methods developed in this work. Show why you chose this type of data and which purpose it has been used.

5.1 Public Datasets

5.1.1 Classification Benchmark - UCR

The University California Riverside (UCR) Time Series Archive was introduced in 2002 and is one of the most used benchmarks for time series data mining tasks, specially for classification. The datasets are diverse in terms of data type, data domain, difficulty, number of classes and dimensionality[**ucr**].

Several *python* distributions make it available to download. In this thesis, all UCR archive datasets from the *pyts* distribution were used for classification tasks. These represent 107 datasets from 18 different data types (??, ??, ??, ??, ??, ??, etc...), which also are from many different domains, such as medical, financial, motion, entomology, etc...[**ucr**]

The list of datasets can be found on the following link [**ucr_site**].

5.1.2 UCI Machine Learning Repository

The University California Irvine (UCI) Machine Learning is another well known benchmark for machine learning tasks. It does not focus solely on time series data mining, having datasets for time series, image, text or categorical data. This archive was created as an ftp in 1987 by a studen at UC Irvine, containing now 607 datasets.

From this dataset, we focused in time series data for the validation of the proposed methods. More specifically, this dataset contains time series that can be used to validate the proposed novelty and cyclic segmentation. We mostly used multidimensional time series related with human activity recognition tasks.

The following datasets were used:

- **Dataset 2 - Smartphone Dataset for Human Activity Recognition in Ambient Assisted Living** This dataset was gathered from an experiment on 30 volunteers. Each subject was wearing a smartphone on the waist while performing several activities: *(1) Walking, (2) Walking Upstairs, (3) Walking Downstairs, (4) Sitting, (5) Standing and (6) Laying*. The activities were performed for approximately 60 seconds. The device recorder the internal accelerometer (ACC) and gyroscope (GYR) data at a constant rate of 50 Hz. Each activity has been categorized and labelled on the acquired data [**dataset2**, **dataset2_2**].
  Purpose: This dataset was used in the context of change point detection, to test the method in estimating transitions between the performed activities from the accelerometer data.

- **Dataset 3 - Smartphone Based Recognition of Human Activities and Postural Transitions**
  The dataset was built in the context of human activity recognition experiments. These were carried out with a group of 30 volunteers that performed a protocol with six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). Additionally, the experiment also included postural transitions that occurred between the static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. The data was collected with a smartphone (Samsung Galaxy S II) mounted on the waist of each subject. The data from a 3-axis accelerometer and 3-axis gyroscope was gathered at a constant rate of 50Hz. The experiments were video-recorded to label the data manually [**dataset3**].
  Purpose: This dataset was used in the context of change point detection, to test the method in estimating transitions between the performed static and dynamic activities from the accelerometer and gyroscope data.

- **Dataset 4 - Wireless Sensor Data Mining (WISDM) Smarphone and Smartwatch Activity Biometrics Dataset**
  The raw data from the accelerometer and gyroscope sensors is collected from a smartphone and smartwatch at a rate of 20Hz. This experiment was conducted on 51 participants as they performed 18 activities, each for a duration of 3 minutes. Each sample of the data was labelled based on the activity it corresponds to. The activities are diverse and include dribbling, eating, jogging, sitting, walking on stairs, standing, walking, among others [**dataset4**].
  Purpose: This dataset was used in the context of change point detection, to test the method in estimating transitions between the performed activities from the accelerometer and gyroscope data from both devices.

- **Dataset 5 - EMG Data for Gestures**
  This dataset has EMG signals for recording patterns, by using a MYO Thalmic

bracelet worn on a user's forearm. The bracelet is equipped with eight sensors equally spaced around the forearm that simultaneously acquire electromyographic signals. The dataset has raw EMG data from 36 subjects while they performed series of static hand gestures. The subject performs two series, each of which consists of six basic gestures. Each gesture was performed for 3 seconds with a pause of 3 seconds between gestures. The data was collected with a fixed sampling frequency of 200 Hz [**dataset5**].
Purpose: This dataset was used in the context of change point detection, to test the method in estimating transitions between the activation and relaxation of the muscular activity.

- **Dataset 8 -** The dataset comprises several time series from real-world data. It was built from a project of the Alan Turing Insitute to have a repository for the evaluation of change point detection algorithms. The available benchmark page was used to get access to the datasets, and run the performance metrics developed. This way, the performance of the proposed method is compare with the performance of several existing methods [**cpd_alan**].

  Purpose: This dataset has ground-truth events for each time series. The performance of several existing methods is available and used to compare with the performance of the proposed method on the same time series.

5.1.3 Kaggle

Dataset 1 -
The dataset comprises data from 15 subjects that were performing 7 activities while wearing a sole wearable accelerometer mounted on the chest. The data was acquired at a constant rate of 52 Hz. The categories of activities are: *(1) Working at computer, (2) Standing Up, Walking and Going Up/Down stairs, (3) Standing, (4) Walking, (5) Going Up/Down Stairs, (6) Walking and Talking with Someone, (7) Talking while Standing*. Each sample of the data gathered has a corresponding label from the performed activity [**dataset1**].

Purpose: This dataset was used in the context of novelty segmentation, to test the method in estimating transitions between the performed activities from the accelerometer data.

5.1.4 Physionet

Dataset 6 - The dataset comprehends 12 half-hour ECG recordings and 3 half-hour recordings of noise typical in ambulatory ECG recordings. The noise recordings were made using physically active volunteers and standard ECG recorders, leads, and electrodes. The three noise records were assembled from the recordings by selecting intervals that contained

predominantly baseline wander (in record 'bw'), muscle (EMG) artifact (in record 'ma'), and electrode motion artifact (in record 'em'). Two clean ECG signals were selected and noise was added with different signal-to-noise ratios (SNR) [**dataset6**, **PhysioNet**].

Purpose: This dataset was used in the context of change point detection, to test the method in estimating transitions to and from noise sections of the signal.

Dataset 7 - This dataset has short duration ECG signals, which were recorded from a healthy 25-year-old male performing different physical activities (standing, walking and single jump) to study the effect of motion artifacts on ECG signals and their sparsity. The dataset was acquired with a sampling rate of 500 Hz and 16 bit resolution. For this exercise, only the records with jump were used [**dataset7**, **PhysioNet**].

Purpose: This dataset was used in the context of change point detection, to test the method in estimating transitions to and from noise sections of the signal.

5.1.5 Alan Turing CPD Benchmark

The Alan Turing Change Point Detection Benchmark is a recent dataset used to have a standard reference for change point segmentation tasks. The benchmark was created in 2020 and provides 42 datasets, being 42 univariate time series and 4 multi dimensional time series. It contains data from real scenarios that varies from finance, weather, society and medical:

Dataset 8 - Alan Turing CPD Benchmark
The dataset comprises several time series from real-world data. It was built from a project of the Alan Turing Insitute to have a repository for the evaluation of change point detection algorithms. The available benchmark page was used to get access to the datasets, and run the performance metrics developed. This way, the performance of the proposed method is compare with the performance of several existing methods [**cpd_alan**].

Purpose: This dataset has ground-truth events for each time series. The performance of several existing methods is available and used to compare with the performance of the proposed method on the same time series.

5.2 Private Datasets

5.2.1 Aritrokinematic Dataset

For this thesis, there were allowed access to inertial acquisitions made in the context of human activity recognition. The database used was obtained on the scope of the *Arthrokinemat* project whose main objective was the development of a learning adaptive

sensor-based measurement system to prevent osteoarthritis[**arthrokinemat**]. More specifically, the recording was made for the work of [**Liu2019**], which introduces a human activity recognition system able to recognize between a list of several daily activities.

A set of multiple Biosensors were used, with various internal characteristics. Beginning with two *8 channel PLUX hubs* a device which allows for the wireless acquisition of biosensors via Bluetooth, with them being part of the *biosignals plux Research Kits*. From both *plux hubs* there were used two 3-axial ?? sensors, 4 sets of ?? sensors and an electrogoniometer, with more details about these sensors being displayed in Table **??**.

Adding to these sensors there were also used 4 other types of biosensors: one airborne microphone, one piezoelectric microphone, two 3-axial gyroscopes and one force sensor. From here, 18 activities were performed recursively, with all of these being listed in Table **??**

Table 5.1: *Plux Hub* sensors descriptions, for the ?? databases[**Liu2019**]. The sensors are divided in half by a horizontal dash line with the first group of sensors using a single *Plux Hub* device and the other sensors using the second device. The information provided by each sensor include also the positioning of the sensors as well as it's sampling frequency.

Sensor	**Position / Muscle**	**Sampling Frequency**
3 axial Upper Accelerometer	Thigh, proximal ventral	100 Hz
3 axial Lower Accelerometer	Shank, distal ventral	100 Hz
Eletrogoniometer	Knee of the right leg, lateral	100 Hz
EMG1	Musculus vastus medialis	1000 Hz
EMG2	Musculus tibialis anterior	1000 Hz
EMG3	Musculus biceps femoris	1000 Hz
EMG4	Musculus gastrocnemius	1000 Hz

The activities **sit-to-stand** and **stand-to-sit** were moments of transition between the activities stand and sit. Moreover the activities **curve-left** and **curve-right** were interposed between the activity of **walking**. **curve-left/right** are also divided into **step** or **spin**, depending if it is a big 90turn with several walking steps or a fast 90turn of the entire body like a parade command, respectively.

Another set of activities that may also require further explanation are **lateral-shuffle-left/right**, a motion usually done in sports that describes the subject's lateral movement of the left/right foot, with the other foot following along and continuing the shuffling in the same direction. **V-cut-left/right** means that the subject changes his direction by 90at jogging speed. The remaining activities are self explanatory.

This database was used to study how the algorithm could be used to detect periodic events with different levels of motion cycle complexity.

5.2.2 Industrial Job Dataset

This database has greater importance in comparison with the previous ones, as it will serve to test if this methodology is indeed valid for processing data acquired from a work

Table 5.2: Activities description, of the ?? database[**Liu2019**]. For each activity is displayed the number of repetitive occurrences each activity is performed as well as the total time length and the minimum and maximum length of each occurrence.

Activity	**Occurrence**	**Total Length**	**Min Length**	**Max Length**
sit	47	123.75s	0.86s	4.69s
stand	46	127.70s	1.36s	4.73s
sit-to-stand	45	30.94s	0.15s	1.30s
stand-to-sit	53	72.90s	0.56s	3.10s
stair-up	55	190.45s	1.59s	4.93s
stair-down	57	181.96s	1.37s	4.86s
walk	220	554.07s	1.18s	4.78s
curve-left-step	57	143.09s	1.10	3.87s
curve-left-spin	46	109.15s	1.25s	3.39s
curve-right-step	51	67.51s	0.54s	3.19s
curve-right-spin	48	41.50s	0.28s	1.88s
run	97	151.27s	0.64s	2.74s
v-cut-left	53	43.76	0.29s	2.08s
v-cut-right	55	61.75s	0.35s	2.37s
lateral-shuffle-left	53	97.54s	0.73s	4.11s
lateral-shuffle-right	52	90.42s	0.75s	3.98s
jump-one-leg	59	61.36s	0.33s	2.85s
jump-two-leg	63	63.40s	0.51s	1.63s
Total	**1157**	**2212.52s**	**0.15s**	**4.93s**

environment. With this said, two main requirements need to be presented. Firstly, the data should have been retrieved in a real-working setting (i.e. not in a laboratory controlled environment), and secondly, it had to be the result of a direct sensor acquisition system that could provide information about the motion of the various workers.

The data acquisition was made in a previous thesis project from [**Santos2019**], whose main objective was to calculate the ergonomic risk through direct measuring data. The technological setting was provided by a sensing framework named Internet of Things in Package (IoTiP), designed and provided by Fraunhofer AICOS. IoTiP is a system that intends to combine hardware, firmware and software components to promote the field "Internet of Things"[**FraunhoferAICOS**]. For this work, the technology setting consisted on 4 9-?? ?? sensors (composed internally by a triaxial ??, triaxial ?? and triaxial ??) and an Android wireless communication system. The last one was made through an application called Recorder, also developed by Fraunhofer AICOS.[**Santos2019**].

The acquisition was made in a Volkswagen Industrial assembly line where 12 manufacturing workers performed their work tasks while having attached 4 ?? sensors in their bodies. During the acquisition, the subjects performed various tasks in multiple workstations. Relevantly to this thesis, the database comprehends three different workstations from *Bodyshop assembly line*, a section where cars' doors were assembled: 1) Liftgate workstation, where back doors are mounted; 2) Fender workstation, involving front door

tasks and 3) Doors workstation, which demanded tasks on the front doors and in the cars' hood[**Santos2019**]. The acquisitions made involved a total of 6 ?? with each one performing at least 2 different ??. The various acquisitions were simultaneously filmed and to synchronize the ground manual annotations of the data, in the beginning and end of the acquisition the subjects were asked to stay, firstly, in a neutral anatomic position and then perform a T pose (calibration position) as represented in Figure **??**. There were also registered some details regarding the anthropomtetric characteristics of the various subjects, as displayed in Table **??**.

The mentioned study was centered on the ergonomic assessment of the dominant arm. For this reason, the ??s were attached in: 1) the posterior side of the hand, 2) posterior side of the forearm and 3) posterior side of the arm and a final one 4) placed in the anterior side of the thorax area. All of the devices were attached with elastic bands, in such a way that all had their Y-axis pointed up while in a neutral anatomical position, as illustrated in Figure **??**. The last device is unique as in its original work it allowed an assessment of the movement of the arm in relation to the entire body. Moreover, it is also different from the other since its ?? sensor was incorporated in a smartphone. Smartphones can indeed work as ?? devices, since these can sense the acceleration, angular momentum and magnetic field. Each one of these 4 ?? devices had incorporated within them 3 triaxial sensors (??, ??, ??), with each producing a multivariate time series of 3 dimensions correspondent to the 3 coordinate orientations of x, y and z.

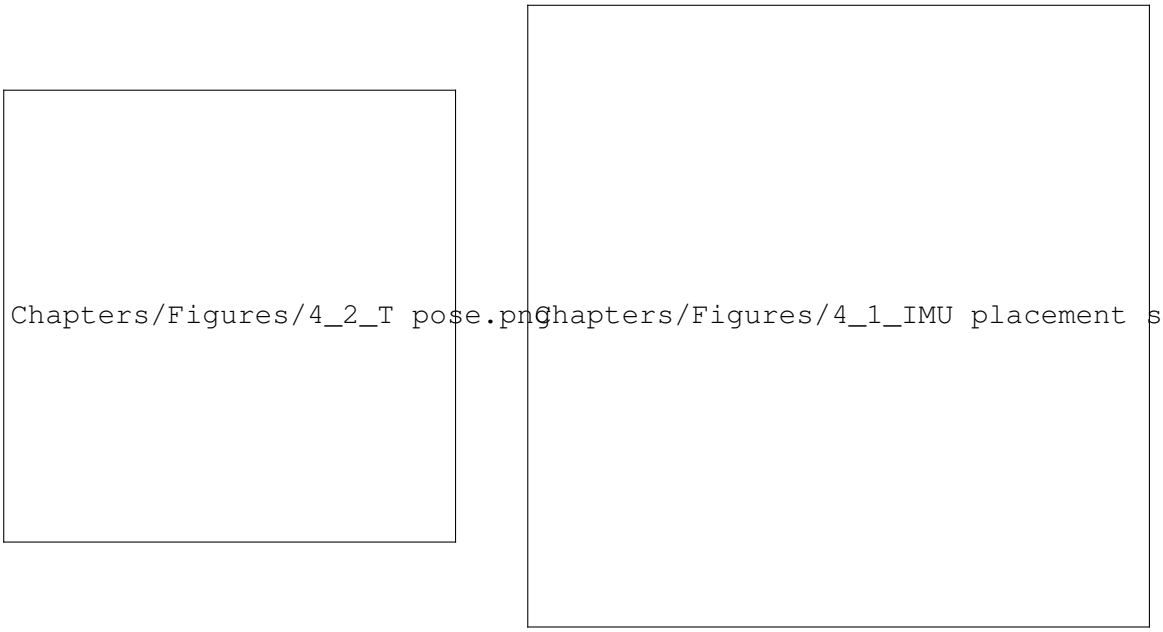


Figure 5.2: Schematic placements of IMU sensors over the acquisition of *Industrial* Database.

The data is hierarchically divided, as showed in **??**. The Database is firstly divided into 12 groups corresponding to 6 subjects with each working at two different workstations;

followed by 4 other databases: Hand, Forearm, Arm and Torso; and then with all of these databases being further divided into 3 other Time Series: Accelerometer, Gyroscope. These time series are multivariate with 3 dimensions representing the 3 orientations of x, y and z of the sensors (Figure **??**). The first level represents the various subject acquisitions made, the second level corresponds to the data retrieved from different ?? devices and the third level represents the data retrieved from different sensor types. Considering this grouping method is relevant due to the data from different databases having different time properties as it will be seen in Section **??**.

This database will help in studying the application of the algorithm to detect (1) Working Periods, (2) Periodic Working Cycles and (3) Search by example.

Some important notes:

1. Despite the actual acquisition having three sensors ??, ?? and ??, only the ?? and ?? sensors were considered for this study as this had the best behaviour, and the ?? was acting in an erratic manner.

2. The two workstation of ??2 were made during the same acquisition, resulting in a single time serie, where the subject performed two different types of active work motions.

3. In ??2 ??1&2 the torso was not considered, due to malfunction.

5.2.3 Office Job Dataset

6

Unveiling the *Grammar* of Time Series

In this chapter is described the process to unveil the structure of a time series. The next sections will start by giving an explanation of the relevant information that has to be retrieved and demonstrates how to perform it. The main method is inspired by what is done in *music structure analysis* for segmentation and audio-thumbnailing. The process follows the steps of building a similarity matrix by means of a feature-based representation of a time series. While having already been extensively studied for audio signal structure analysis[**Mueller15_FMP_SPRINGER**, **audiolabs1**, **audiolabs2**, **cpd_audio**], this knowledge has not yet been extended to other types of *time series* domains, which could greatly benefit from it [**muller_music_health**].

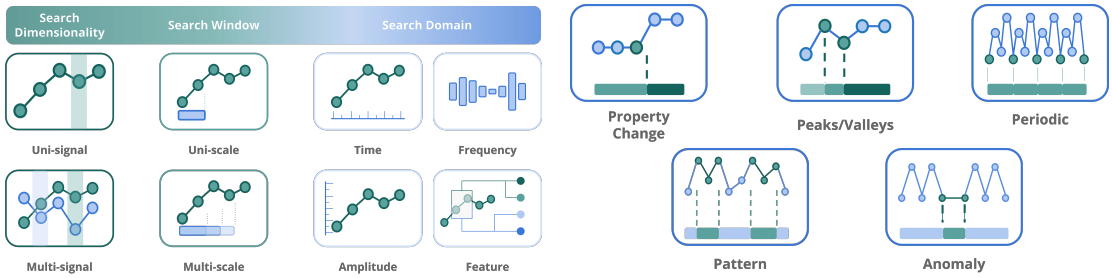
6.1 The Problem

Defining what is relevant in a time series highly depends on the context and purpose of the analysis, but globally, for any type of time series, there is a general interest in understanding how the signal is structured, specially for tasks related with data annotation/labeling. The structure of a time series is built of *segments* delimited by *events*. The problem is the search for *events* that are significant.

From definition 4 of Chapter **??**, we highlight two primary considerations for the detection of events: (1) an event is a change in the behavior of the time series, and (2) it has to be significant both *statistically* and *qualitatively*. The *qualitative* aspect indicates subjectivity from the analyst because of the domain or context of the problem. Considering this, we will start by explaining the dimensions of the problem: (1) search and (2) type of significance.

6.1.1 Search Dimension

Figures **??** and **??** are an illustrated summary of the two dimensions of the problem. Regarding the *search* dimension, it is formed by three layers: *(1) dimensionality*: the search can be made in one or multiple time series. In the multidimensional space, events can occur simultaneously in several time series, but other events can be specific for each of them

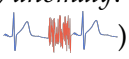


(a) (b)

Figure 6.1: (a) Categories of search of events. In this case are shown Dimension, Window and Domain. (b) Examples of different type of events that can be considered significant in a time series.

(e.g. an accelerometer signal has 3 dimensions, but some gestures might be noticeable in only one of them); *(2) time scale*: *events* might occur in different time scales (e.g. when looking in a *TS* of 1 hour long, we might see some relevant events, but when looking for a *subsequence* of 10 minutes (zooming-in), other events are revealed); *(3) domain*: the search procedure might be made directly on the time series by means of time properties, a distance measure (e.g. Euclidean distance), or can be made on the representation level, such as the feature domain.

6.1.2 Type Dimension

In what regards the *event* type, we show in Figure **??** examples of events that are considered significant in a time series: *(1) Property change:* when the change of a property or set of properties is greater than a threshold, such as changes on the mean (FIND THUMBNAIL IMAGE) or frequency (), *(2) Peak/Valley*: peaks and valleys can typically be associated with significant physical changes (e.g. the peaks of an ECG signal ), *(3) Periodicity*: if a signal is periodic, the moment each period starts is considered relevant (e.g. the cycles of a BVP signal ), *(4) recurrent pattern*: re-occurrences of similar *subsequences* with a certain shape or *(5) anomaly*: very dissimilar *subsequences* are relevant to indicate (e.g. noise in a clean signal ).

6.1.3 Proposal

In order to fill as much ground as possible in this problematic, we started by defining the search space considering that if the time series would be transformed in the feature space, any change in any of the features would be relevant, for instance, we might be searching for changes in the mean, standard deviation, mean frequency or other property. By characterizing the signal into the feature space, we are able to explore changes in all feature representations. Additionally, an event should separate two different behaviors. The notion of *difference* in time series can be associated with *distance/similarity*. This would enable to find change points, recurrent patterns, anomalies and periodic shapes.

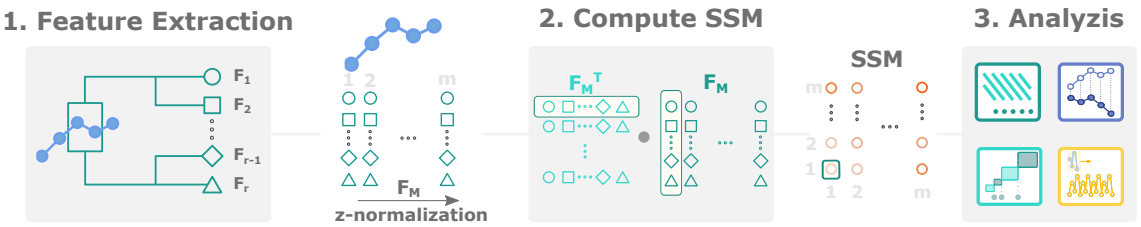


Figure 6.2: Main process to reach the ??. The information needed to calculate the ?? are the record and the input parameters: the window size (w) and the overlapping percentage (o). The first stage involves the feature extraction process, based on w and o values. Features are extracted on each window (1, 2..., N), being N the total number of windows. From the first window (w_1), are extracted features (f1, f2..., f_K, being K the number of features used. The feature number is also associated with a shape (circle, triangle, etc...). The features can be extracted on multivariate records, being M the number of records used. Each feature is positioned as a row on the F_M. Then follows the ?? computation.

Therefore, we propose an unsupervised methodology that searches for events in (1) uni and multi dimensional space, (2) with a fixed time scale, but with potential to be used in multi-time scale and (3) on a ?? computed by a feature space representation of the time series. The events that will be searched are any relevant change in the matrix related with a change point and/or periodic event.

We provide evidence that the proposed method is reliable for the detection of the mentioned events, supporting our claims with several examples in multiple time series domains (it is type agnostic) and comparing the results with state-of-the-art methods. Besides, we highlight that these events are all extracted from the same source of information (*SSM*), while also providing some insights in how this could be expanded for multi-time scale search, used for summarization and labelling.

6.2 Building the SSM

In this section, we explain the steps of the proposed method. The extraction of relevant events from time series starts by computing the ??. As explained in Chapter **??**, this matrix has relevant structural information to retrieve *events*, namely *blocks* and *paths*. Figure **??** summarizes the steps involved in calculating the ??.

6.2.1 Feature Extraction

The structural information present on the SSM depends on the richness of the set of features into translating the changes and disruptions of the signal. Behavioral changes might be related with a variate set of features. As a feature may be sensitive for a type of change, the type of features should be diverse to identify a multivariate set of events and scan all types of signals. For this purpose, we used the available features from the TSFEL **[barandas_tsfel_2020]** Python library presented in the Feature Table **xxxx**.

The features are extracted in a moving window with a size w, specified by the user, with an overlap of size o. These two parameters have a large influence on the results. The first defines the time scale at which features are extracted, therefore the wider the window, the more *zoomed-out* will be the search. The second parameter defines the pixel-resolution of the resulting feature series, decreasing the amount of information (down-sampling) with a smaller overlap.

The extracted features are grouped into a feature matrix (F_M), where the rows represent each feature and the columns the corresponding *subsequence*, described by all features. Features extracted from a multidimensional record are ordered in the F_M as rows as well. The total number of rows can be, at maximum: $r \times k$, being k the number of time series being analyzed and r the number of features extracted, as illustrated in Figure **??**.

6.2.2 Feature-based Self Similarity Matrix

After grouping all the features extracted, the next stage applies a distance measure to the feature space and computes the ??. This process consists in comparing each *subsequence* with the all the others within the time series record. Since each column of the F_M is the feature characterization of each *subsequence* by the entire set of features, the comparison between segments is achieved by calculating the dot product between the z-normalized transposed F_M and itself:

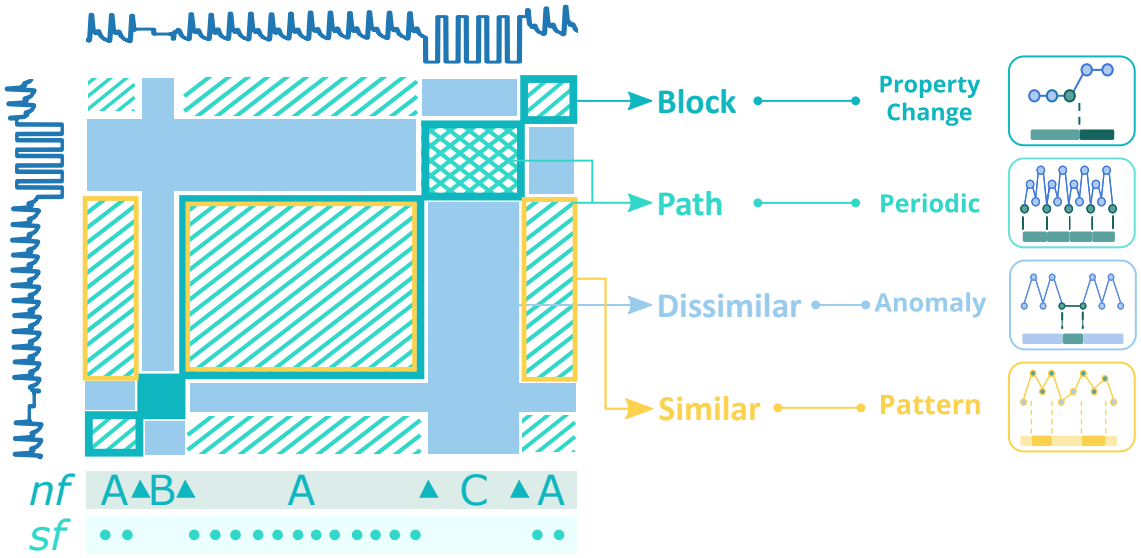
$$SSM = F_M^T F_M \tag{6.1}$$

The dot product gives a similarity score based on the feature values of each *subsequence*. Cells of the ?? with higher similarity scores indicate that the corresponding *subsequences* have similar feature values [**audiolabs1**, **audiolabs2**]. As a result, the ?? provides rich visual information, highlighting structures, such as blocks and paths, that describe the signal's morphological behavior over time and its structure.

In Figure **??**, the main structures are illustrated and highlighted in an example of an *??* [**audiolabs1**]. These structures are illustrated on the time series of example 1 (Figure **??**. As mentioned in Chapter 5, the main structures are *blocks* and *paths*. Our proposed methods for annotation takes advantage of these main structures to extract the desired information.

Paths show recurrence of patterns, which is an indication of matching the morphology between corresponding *subsequences*. The example highlights circles in the *sf* layer, indicating recurrence. In *block* "C" are also visible *inverse-paths*, which indicate symmetry, which means that the corresponding *subsequences* are similar in reverse. The cross-path in *block* "C" means that the *subsequences* are periodic and symmetric.

Differently, *blocks* are square shaped structure that indicate homogeneous areas of the ??, which translate as constant behavior in the time series. The change between block structures along the main diagonal indicates a relevant change of morphology and behavior on the time series. In Figure **??**, the ?? is segmented into several blocks on layer



.

Figure 6.3: Description of informative structures of the *SSM*. Based on the ?? presented in Figure **??**, we show a simplified view with highlights on the relevant structures. The record has 4 main structures: A - homogeneous segment, which corresponds to the BVP periodic signal; B - homogeneous segment of missed data; C - homogeneous segment with detachment of the sensor. The boxes (blue) highlight homogeneous blocks while the sub-diagonals (orange) highlight periodicity in the segment. *nf* and *sf* indicate that the novelty and similarity functions are computed based on this information. Segment C has a cross-pattern, which indicates periodicity and symmetry

nf. The triangular shapes indicate the change point that separate blocks "A", "B" and "C". Besides *paths* and *blocks*, the ?? provides distance measures between *subsequences*, which can be used to highlight dissimilar segments, such as anomalies or highlight very similar *subsequences*, such as motifs (pattern).

Several strategies were applied on the ?? to extract the mentioned information. Further are explained the approaches used.

6.3 Information Retrieval

The ?? is a powerful visual tool *per se*, highlighting relevant information that could be missed if looking at the raw time series. However, being the information on the ??, it should be possible to retrieve it automatically. As Figure **??**, here are explained how to extract three layers of information, namely (1) change points in a time series (*block* transitions), (2) periodic segmentation of patterns (*paths*) and (3) how similar are the *subsequences* segmented by the previous methods (*distance profiles*). In addition, we will also demonstrate how the ?? can be used to search queries in a periodic time series.

Figure 6.4: Information retrieval topics explained in this section.

6.3.1 Novelty Search

The search for *novelty* is inspired by a method used in musical structure analysis and presented by *Foote et al.* [**foote2000**]. The process involves searching for transitions between *blocks* using a moving checkerboard square matrix. The result is a 1 dimensional function designated *novelty function - nova*.

As showed on Figure **??**, block transitions along the diagonal are represented by a checkerboard pattern. Detecting such patterns can be made by correlating a standard checkerboard matrix with the diagonal of the ??. For this, a sliding squared matrix, designated *kernel*, is used. As illustrated in Figure **??**, the kernel has a checkerboard pattern and is combined with a Gaussian function to add a smoothing factor. The kernel, K_N, is a combination of two different kernels: K_H and K_C. The first is responsible for identifying the homogeneity of the ?? in each side of the center point along the diagonal. The higher the homogeneity, the higher will be the values in these sections. The latter measures the level of cross-similarity, returning higher values in cases of high cross-similarity. Therefore, when sliding the kernel K_N along the diagonal, a higher correlation value will be returned when it reaches a segment of the ?? with a similar checkerboard pattern. The result is the mentioned *nova* [**Dannenberg2008**, **Mueller15_FMP_SPRINGER**, **MuellerZ19_FMP_ISMIR**].

As showed in Figure **??** (left), the kernel in position **A**, which is placed on an area of high homogeneity, returns a value close to 0 when summing the product between it and the section of the ?? it overlaps. In the other end, in position **B**, the kernel reaches a segment with low cross-similarity and high diagonal similarity, which results in high correlation values with a checkerboard pattern. The *nova* is high in these transition segments [**Dannenberg2008**, **Mueller15_FMP_SPRINGER**, **MuellerZ19_FMP_ISMIR**].

Each section of the kernel has the same size, L, being the total kernel size configured by $D = 2 \times L + 1$, with $L \in \mathbb{N}$. The kernel has an odd size to adapt zero values in centered points. It also has total size D × D, being K_N defined by the following function [**Mueller15_FMP_SPRINGER**, **MuellerZ19_FMP_ISMIR**]:

$$K_N(i, j) = (a_i) \cdot (b_j) \tag{6.2}$$

being $a, b \in [-L : L]$ and "" representing the sign function, which indicates the sign of the value (1, 0 or -1). A radially symmetric Gaussian function is used to smooth the Kernel, with the following equation [**Mueller15_FMP_SPRINGER**, **MuellerZ19_FMP_ISMIR**]:

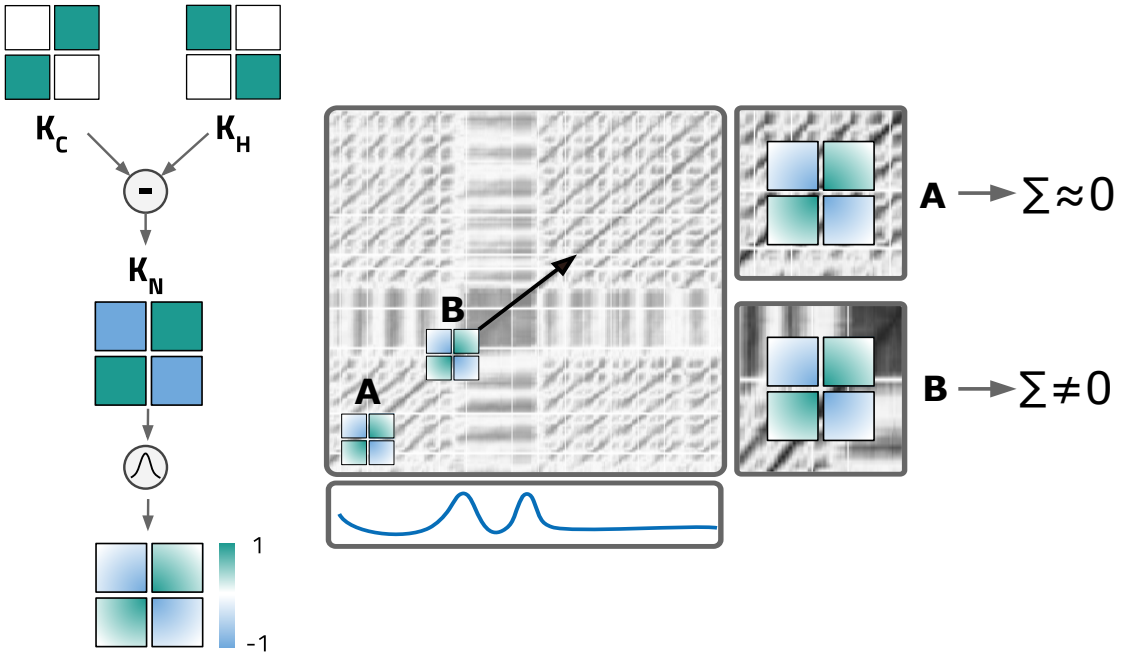


Figure 6.5: (left) Description of the matrix (kernel) used to compute the *novelty function*. The checkerboard pattern is achieved by combining kernel K_H - measure of homogeneity; and K_C - measure of cross-similarity. The resulting kernel (K_N) is combined with a Gaussian function to generate K_G. The Figure is based on the works of *Mueller et al.* [**Mueller15_FMP_SPRINGER**, **MuellerZ19_FMP_ISMIR**]; (right) The process to compute the novelty function is described. Kernel K_G is slided along the diagonal of the ?? to compute the *novelty function* presented as the bottom sub-plot. Positions A and B show the effect of block transitions on the *novelty function*. Figure based on the works of [**Dannenberg2008**, **Mueller15_FMP_SPRINGER**, **MuellerZ19_FMP_ISMIR**].

$$\phi(p,u) = \exp(-\frac{1}{2L\sigma^2}(p^2+u^2)) \tag{6.3}$$

being σ the standard deviation, equal for both x and y dimensions of the matrix, L the size of each kernel's section, and p and u the position in the x and y dimensions, respectively. The final kernel is computed by point-wise multiplication with the Gaussian function:

$$K_G = \phi \cdot K_N \tag{6.4}$$

The *nova* is calculated by correlating the kernel with the diagonal of the matrix:

$$nova(m) = \sum_{i,j=0}^{2L+1} K_G(a_i, b_j) SSM(m+a_i, m+b_j) \tag{6.5}$$

being the sample of the novelty function $m \in [0-N]$ and $a, b \in [-L:L]$. The change point events are represented by local maxima (peaks) in *nova*, which can be detected by standard peak finding strategies.

6.3.2 Periodic Search

As aforementioned, *paths* indicate the presence of similarity and reoccurring patterns can be visualized on the ??. The moment in time the *paths* start indicates the position at which the period of the pattern begins. In order to find the periodicity, we compute the similarity function, s_f, which is calculated by summing the values of the ?? column-wise (either column-wise or row-wise would work, since the matrix is symmetric), being each element of the sf calculated by:

$$sf(x) = \sum_{i=0}^{m} SSM_{ix} \tag{6.6}$$

where i is the column position for the sum, sf_j the sample of the function at position j and m the size of one of the dimensions of the ??, which is equal to the feature-series size. As segments with similar morphology will be similarly described by the extracted features, the columns will have a similar representation, hence a similar value on the *sf*. In cases where the time series is periodic, the similarity function will enhance this behavior by having valleys at the moment the *path* starts. The identification of events related with the periodicity of a time series is then possible by searching for local minima (valleys) on the similarity function.

6.3.3 Similarity Profiles

The main elements, *blocks* and *paths*, are essential sources of information for the segmentation of the time series. Besides these, the ?? also provides the pairwise similarity values between all *subsequences* of the time series. This is an important measure that gives an understanding of how close together are each *subsequence* and can be used to cluster them. In order to use the similarity values of the ?? to compare *subsequences* we compute *similarity profiles*.

The comparison between *subsequences* could be made by directly using the values of the matrix in the segment delimited by both *subsequences*. Although this would be a legitimate process, we find that a stronger measure is to compare how much each of the two *subsequences* are similar/different to all the other *subsequences* of the time series. For this, *similarity profiles* ($P_s(c)$) are computed as the average similarity values of a section of the ?? delimited by the *subsequence* being profiled, with size w, and all the other *subsequences* of the time series, with size m:

$$P_s(c) = \frac{\Sigma_{i=0}^{w} SSM(i, c)}{w} \tag{6.7}$$

The *similarity profile* is computed column-wise, being each column c the average similarity value between the reference *subsequence* and the *subsequence* corresponding to c. The reasoning is that similar *subsequences* should have closer *similarity profiles*. Since the profiles have the same size, these can then be compared with the *euclidean distance* and

Figure 6.6: Profiles computed for each segment of the example signal used in Figure **??**.

clustered based on these distance values. This process is specially valuable to cluster the segments previously extracted with the *nova* and *similarity* functions.

A general example of applying this process to the segmented time series based on the *nova* function is showed in Figure **??**. Each segment category (A, B and C) extracted from the ?? of Figure **??** is computed into a profile (P_A, P_B and P_C) by averaging column-wise. These *similarity profiles*show how similar the segment is with all the other *subsequences* of the time series. All segments A will have a similar P_A, while being very different from profiles P_B and P_C.

6.3.4 Query Search

Another relevant application of the ?? is to make query search based on examples. The process follows the traditional methods of template-based search methods explained in Chapter

$$D(i) = \sum_{i=0}^{i=m} \sqrt{(SSM(i) - SSM_t)^2} \tag{6.8}$$

where $SSM(i)$ is the segment of the SSM over which the example, SSM_t, slides at moment i, up to the size of the ??. The resulting distance function has minimums at the position where the example is matched.

6.4 Experimental Evaluation in Selected Use-cases

After explaining the process to represent the time series into a feature-based similarity matrix and the methods used to retrieve information from it, we present selected use-cases from multiple domains to exemplify its universal usage.

6.4.1 Use-Case 1 - Human Activity

The example presented in Figure **??** shows the usage of the ?? on a record of Dataset 2. In this example, the method was applied to all the 3-axis of the accelerometer data. All are showed and described with the sequence of activities as captioned in Figure **??**.

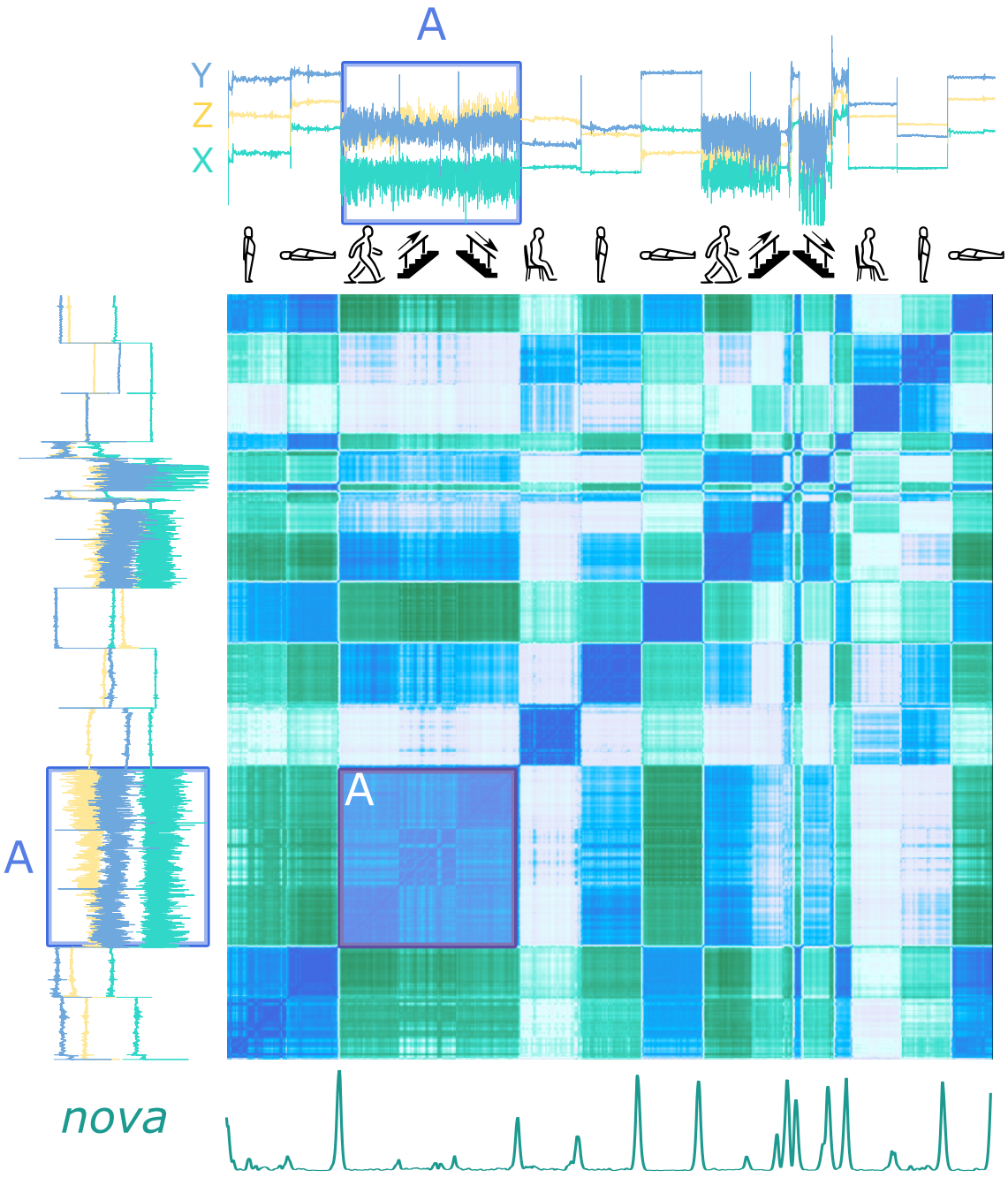


Figure 6.7: Change point event detection strategy applied on the SSM to search for change point events. The sequence of activities is presented as follows: $Sitting \rightarrow Laying \rightarrow Walking \rightarrow Upstairs \rightarrow Downstairs \rightarrow Sitting \rightarrow Standing \rightarrow Laying \rightarrow Walking \rightarrow Upstairs$. The input variables used are $time_{scale}$=250 samples, $kernel_{size}$=45 samples, overlap=95%

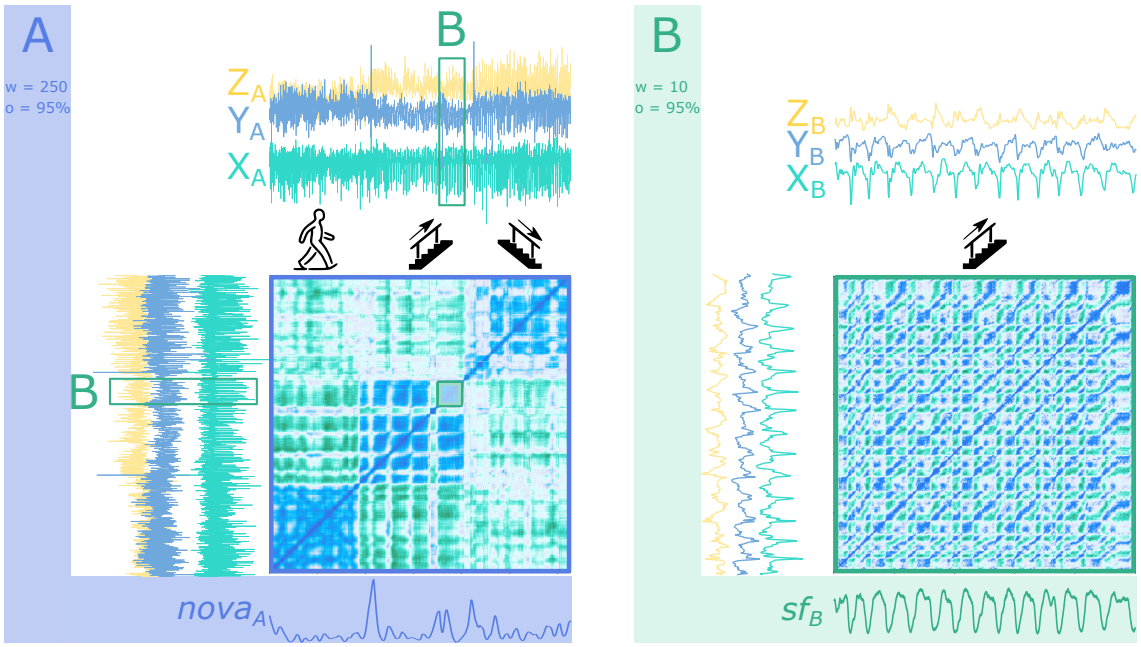


Figure 6.8: ABP signal change point detection. The parameters used were a size of 5000 samples, with an overlap of 75% and a kernel size of 25 samples.

The ?? was computed using a window size of 250 samples, and an overlap of 95 %. The color *blue* indicates segments with higher similarity. Along the diagonal, these blocks are visible and the events are estimated as the transition between these, highlighted by the *nova* function. The kernel used for this detection had a size of 45 samples.

In this example, we can identify that the detected change point events match with the activity transitions. Although all transitions are visible on the novelty function, the ones that correspond to transitions between similar segments of activities are harder to find, namely the transitions between walking activities. This is plausible since the properties of these segments are similar and the morphological difference is not as significant as when shifting between dissimilar activities (e.g. between *Laying* and *Walking*).

Any significant change in properties will be detected by the proposed method. As presented in Figure **??**, at the end of the time series, the period in which the subject was performing the *Walking upstairs* activity is affected by other changes in the time series. These are significant and also correspond to *block* transitions, which are also evident in the novelty function. The proposed strategy, being unsupervised, is sensitive to any change, as long as it is observed as a significant change in the signal's properties

When *zooming-in* the *SSM* into segment *A*, which shows transitions between walking behaviors, the checkerboard pattern that highlights change points is clearer and the three different walking patterns are easily segmented. The two major peaks in the corresponding *nova* function are from these transitions, as presented in Figure **??** (left). In addition, the reader might notice that the segments of the matrix related with *walking in stairs* are also segmented into smaller *blocks*. Although the information is not available on the dataset

description, we strongly believe these are flight of stairs.

Considering that the signal is a walking behavior, the reader might question the fact that the periodicity of the walking pattern is not exhibited on the matrix. The reason is that the window size used to compute the ?? of Figure **??** is too large. If features are extracted with a smaller window size, closer to the walking period, the *paths* that indicate the recurrence of shapes are visible. Figure **??** (right) shows the ?? built from the segment B of the original time series, with a window size of 10 samples and an overlap of 95 %. The matrix shows the *paths*, from which it is possible to extract the periods with the similarity function (sf_B).

6.4.2 Use-Case 2 - Medical domain

In the medical domain there are several examples of structural information to retrieve. Some signals are periodic, such as the ??, the ?? or the ?? signals. When acquiring this type of data, several instances might reflect unexpected changes, either because of physiological responses and medical disorders or due to the sensing process. Here we show two examples of physiological changes in two periodic signals.

The ?? signal can change due to postural changes. An experiment was conducted to study this effect and is available at Physionet [**tilt**, **PhysioNet**]. In Figure **??** is showed the process of segmenting the ?? signal based on postural changes, signaled with the ground truth. The change points are well perceived by the proposed strategy. The reader can notice that the shape of the ?? signal in each regime is very similar, being hard to notice by human eye where it happened. In addition, the periodicity of the signal is not visible on the matrix because the features were extracted with a window size of 5000 samples, which is much larger than the size period length.

The ?? of Figure **??**.a also shows which segments are similar to each other. The blue colors of the matrix indicate high similarity and it is clearly presenting that segments with positive *posture change* are more similar.

The same happens on the ?? signal from Figure **??**.right. It displays a condition called pulsus paradoxus.... Again, the reader can notice that the change point is hardly perceivable by human eye, but the proposed strategy is able to clearly show the difference in both regimes.

6.4.3 Use-case 3 - Multidimensional

The proposed method accepts both single and multidimensional records. The difference regards the number of features extracted. As presented on Figure **??**, the same set of features are extracted for each time series of the record and combined in the F_M.

Using a single time series of a multivariate record is optional and depends on the detection's purpose. In some cases, using a single time series from a multidimensional record can lead to missing relevant events undetected. An example of this can be seen on Figure **??**.a with record *"Occupancy"* from Dataset **??**.

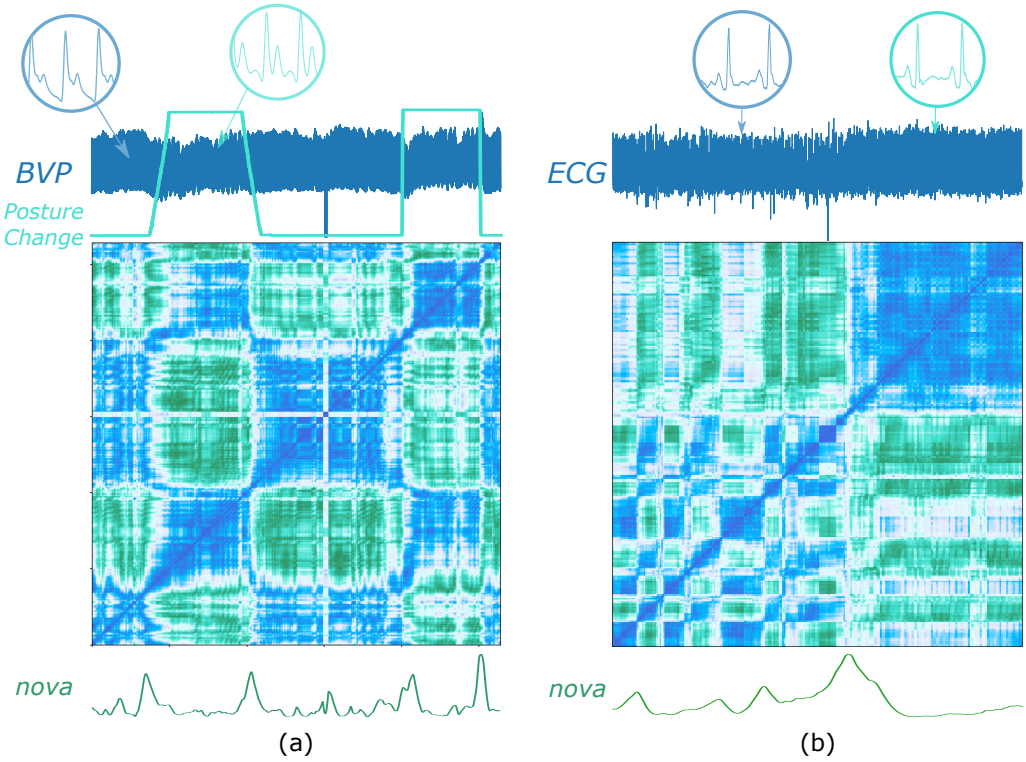


Figure 6.9: (a) ABP signal change point detection. The parameters used were a size of 5000 samples, with an overlap of 95% and a kernel size of 200 samples.

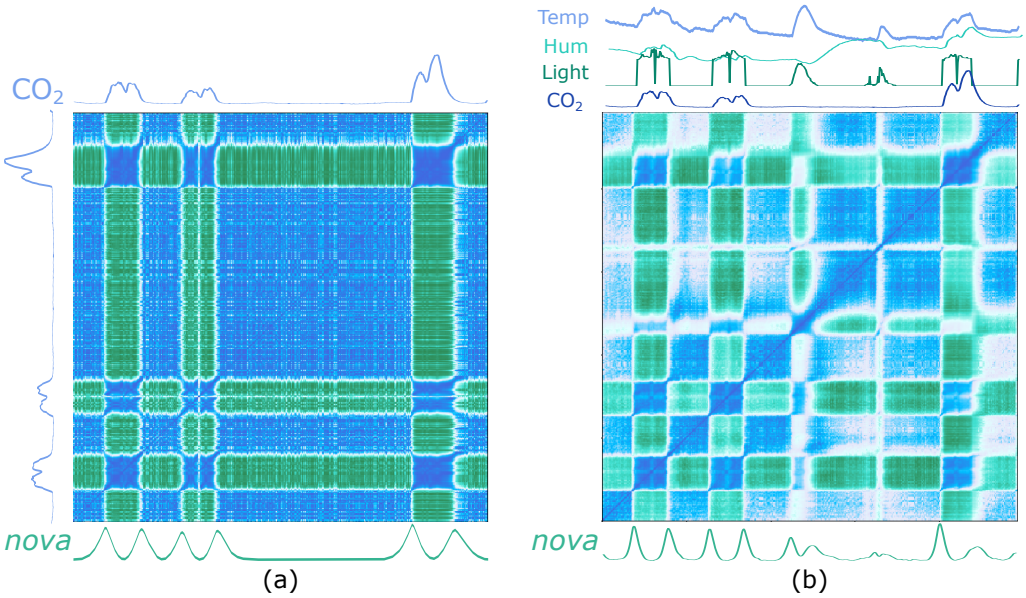


Figure 6.10: Proposed method applied on "*Occupancy*" record of Dataset **??**. (a) A single time series of the record is used to extract events; while in (b) the ?? is computed with features extracted from the four available time series.

The record is a multi-dimensional time series that measures room occupancy based on temperature, humidity, light and CO_2. All events can only be detected if using several time series of the record [**cpd_alan**]. On Figure **??**.a, a single time series was analyzed by the proposed method to detect relevant events, while Figure **??**.b is the result of using all the time series of the record.

6.5 Time Series Profiling

The aforementioned examples show how the proposed method can be used as a strong and reliable visual tool. It is possible to see how a time series is structured, how similar are segments and if these are periodic or not. The information available is quite relevant to support the labeling process of the analyst, but it also can be used to summarize a time series and give a meaningful report about it.

Following the presented work, we studied how to use it for a meaningful summary of time series. This process is inspired by methodologies that exist in other scenarios for data summarization techniques with statistical analysis, such as the available methods from the *pandas python library*: *pandas.profile()* and *pandas.describe()*. These methods are able to provide a summarization of a dataset (typically of categorical data) that is given as input. A similar method is not known for time series. In order to develop such a method, we should first understand what is meaningful and relevant to represent as a summary.

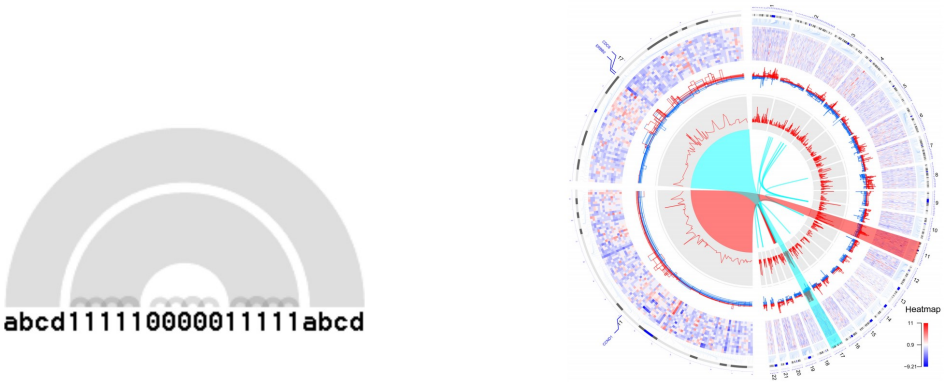
6.5.1 Elements with Relevance

In this section, we have been discussing which elements are relevant in time series, mostly associated with *events*. From *events* we can segment homogeneous *subsequences*, recurrent or periodic patterns and anomalies. The relationship between these segments is possible analyzing their similarity. In addition, a characterization of the segments is possible with a statistical analysis. In that sense, the relevant elements to summarize in a time series are (1) *homogeneous segments*, (2) *periodic patterns*, (3) *recurrent patterns*, (4) *anomalies*, (5) association based on similarity and (6) statistical characterization. Several examples from other domains can be used as inspiration in how to join all these elements in a compact, expressive and intuitive way.

6.5.2 Compact Design

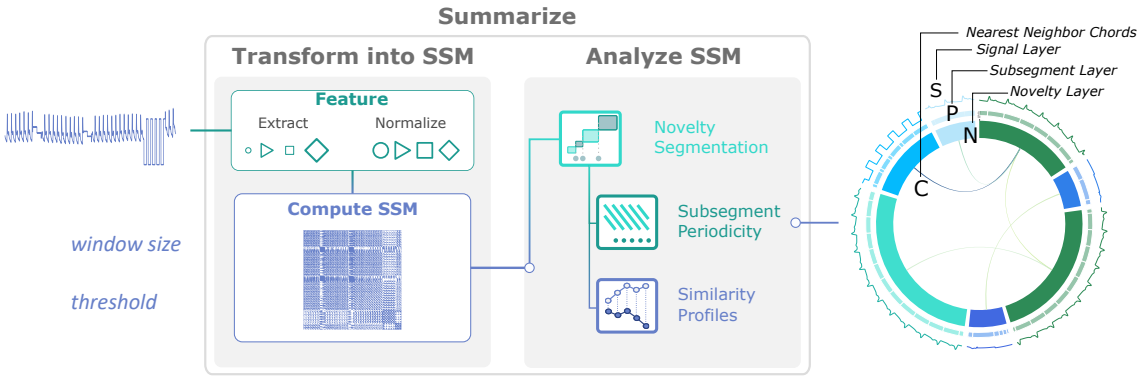
Strategies that are typically used to present information in a compact way are found in several domains. In text analysis, for instance, the relationship between repeating sequences is illustrated with arc diagrams [**bitmap**, **arcplots**]. These show where repeating sequences occur in a very concise way. This has a range of applications that include, for example text and DNA sequence analysis.

One domain that has interesting examples in data visualization is *genomics*. Graphical genome maps are found to concatenate a significant amount of information in a very



(a) (b)

Figure 6.11: A - Diagram for string association. This image is taken from the works from [**arcplots**]; B - Circular plot by OmicCircos. Several layers (Circular tracks) identify genome position, expression heatmaps, correlation between expression and CNV, among other features. The image is taken from the works from *Ying Hu, et al.* [**genomics**].

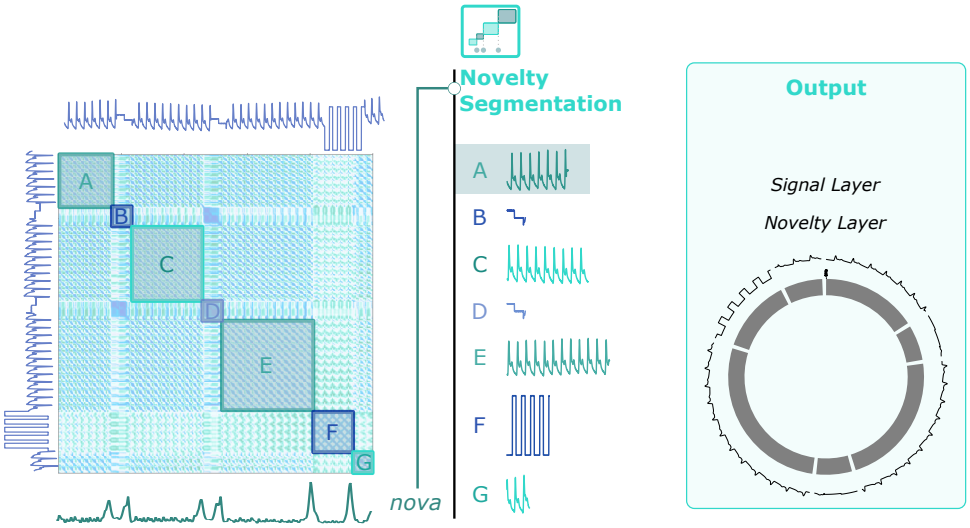


compact way. Genome features and sequence characteristics are assessed with this visual strategy. An example can be found on Figure **??**. This type of visualization inspired the summarization approach proposed for time series.

This visualization strategy has several elements that can be used to transform the ?? into a compact form of filtered information. The elements are (1) multi-layered colored *segments*, (2) *chords* that connect to nearest neighbor *segments* and (3) circular signals on top of segments. The transformation into this compact representation can be performed with the explained analysis methods above.

6.5.3 A step by step example

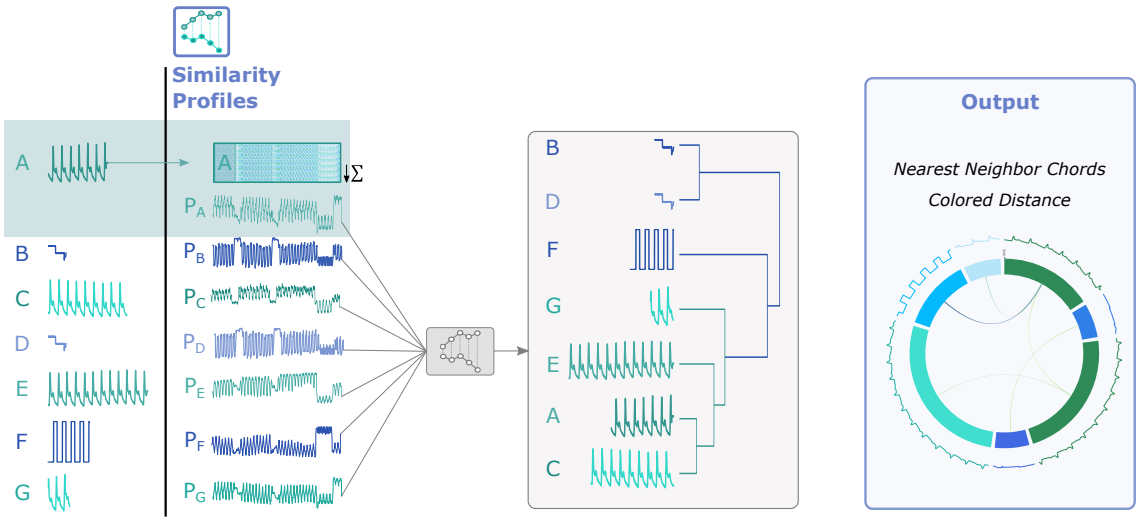
The steps are indicated in Figure **??** for the ?? signal example. After computing the ??, it is firstly analyzed to segment the signal based on the *nova* function. These segments are then compared based on the *similarity profiles*. Additional layers can be created by performing an iterative and multi-scale segmentation. With this process, the time series is segmented (*novelty layer*), subsegmented (*subsegment layer*), each segment is connected to the nearest

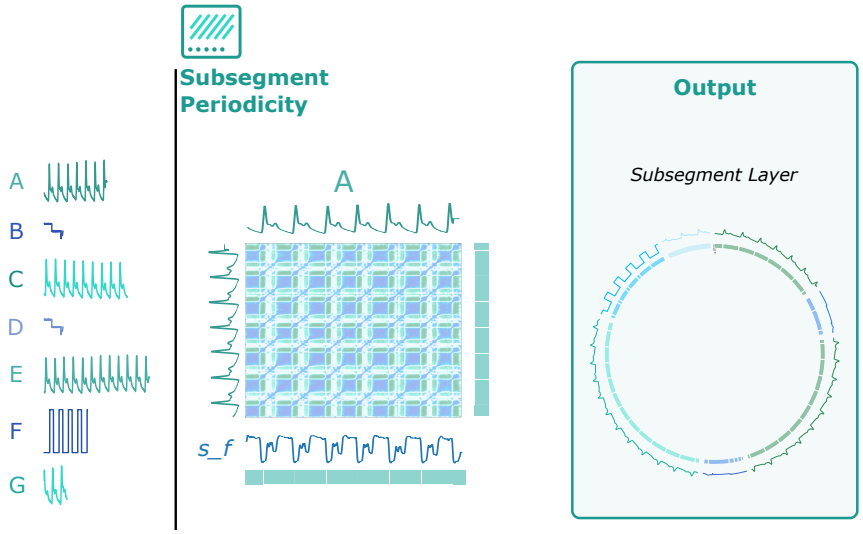


neighbor segment (*nearest neighbor chords*) and the colors for each segment is given based on their similarity to the first segment. Figures **??**, **??** and **??** show the step-by-step process to summarize the time series by analyzing the ??.

The *nova* function segments the time series into seven segments. The reader can notice that segments A, C, E and G are similar and separated by segments B, C and F, represented by a failure in the connection of the sensor. From this first segmentation, the *novelty layer* is created, indicating how structured is the signal, as presented in Figure **??**.

Further, the segments are compared with the *similarity profiles* of each segment. Figure **??** illustrates as example the rows of the ?? delimited by segment A and the column-wise average into P_A. The same process is applied to each segment. From this Figure, the reader may notice that the profiles P_A, P_C, P_E and P_G are more similar, while profiles P_B and P_D are more similar as well. The pairwise distance is computed between profiles, which can then be used to extract the nearest neighbor of each segment, as well as transform the distance values between segments into color.





The pairwise distance between profiles is also used to illustrate how the segments are ordered and clustered by the dendogram of Figure **??**. The dendogram shows that there are three main clusters, being cluster one represented by segments A, C, E and G; the second cluster has segment F and the third cluster groups segments B and D. It is important to note that typical distance measures, such as the ?? or ??, would not be able to directly sort these segments correctly. Using *similarity profiles* is more robust and invariant to the size and time distortions.

Finally, the process to summarize the time series can be iterative by adding *subsegment layers*. These layers can be added by performing the *novelty search* on the previously segmented time series with a smaller time scale, or segmenting the time series based on periodicity. In this case, the signal is periodic, therefore Figure **??** illustrates the *periodic search*, where periods are segmented by the minima of the s_f.

7

Language for Time Series Data Mining

In "Pattern Recognition: Human and Mechanical", Satosi Watanabe defines a pattern as a vaguely defined entity that is the opposite of chaos, to which a name can be given [**watanabe**]. The recognition of these entities immersed in chaos is well performed by the human brain by distinguishing or finding similarities in features that characterize a certain pattern, being an innate capability for decision making [**niesser**]. This capacity is also revealed in the search for patterns in time series, as data scientists are able to visually understand where these patterns occur with previous training and express it linguistically.

Human language is foremost a means of communication, in which the information is represented by sentences, composed by words that can be broken into sequences of symbols. Time series are, in their turn, carriers of information about a certain measure, as sequences of ordered real domain numerical data observed during a given temporal interval. For instance, analysts have a visual perception of the morphological behavior of time series and are able to describe it in such a way that another subject understands which portion of the time series he/she is referring to. As we mentioned in Chapter **??**, a physician may say "*I am searching for the T-wave, that represents the* `large peak`" () or "*I am searching for the QRS complex, that looks like a* `sharp peak followed by a sharp valley` ()". From this textual description of patterns, the reader can associate words with specific properties, such as `peak` can be related with *slope* and `high` related with *amplitude*.

The text mining domain already has several approaches to search for specific patterns based on text queries. The reader may already have used the command *cntrl+f* to search for specific words, or used regular expressions to search for specific patterns of words in text documents. Most probably, the reader also uses the *Google Search* toolbar with a simple list of keywords, sorting the list of documents according to these.

In this chapter, we intend to present the reader with solutions that promote a higher flexibility, cognitive-speed and expressiveness in searching for patterns in time series as we typically do with text. Two main strategies were explored: (1) ?? and (2) ??. The first profits of a symbolic characterization of the signal, introducing a novel representation and the usage of *regular expressions* to search for patterns on this representation. The second

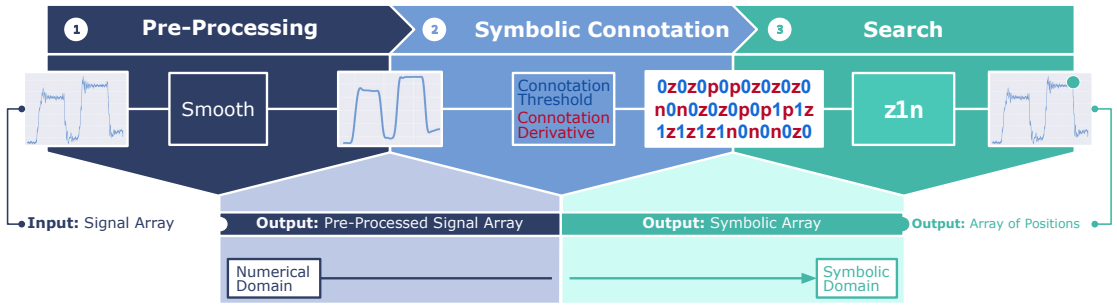


Figure 7.1: SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search.

uses a feature based representation of the signal, being each feature attributed to one word, which can be used with additional *operators* to search for the desired patterns with natural language.

The chapter will start by detailing how ?? works, showing its usage in several examples. In addition, a symbolic representation of the time series can also be used to differentiate classes of signals. Therefore, a proposed strategy towards interpretable time series classification with ?? is introduced and explained as well. This chapter ends with ?? and how a user can *"google"* patterns on time series with keywords.

7.1 Syntactic Search on Time Series

In Chapter 2 was presented the well-known ??, which is a symbolic representation of the time series based on the amplitude distributions. With ??, query with text was not considered, but ?? takes inspiration of this symbolic transformation by including additional attributes besides amplitude, such as the derivative, speed of the derivative, etc.... Each property characterizes each sample of the signal. The combination of these attributes into sequence of primitives is a symbolic characterization of the signal that enables the use of regular expressions for searching the desired pattern. The proposed tool focuses on the knowledge retrieved from the visual interpretation of the signal that enables the search of the desired patterns by writing a regular expression that parses the syntactic representation of the signal.

The proposed tool, ??, uses symbolic sequences to perform each of the three steps that compose it: *pre-processing* - prepare the signal to highlight the desired patterns and ease the search process; *connotation* - symbolic representation of time series into a set of primitives that give information about the shape and attributes of the time series over time and is governed by a set of grammatical rules; and *search* - a parser (??) to search for patterns in the symbolic representation. Both *pre-processing* and *connotation* steps rely on *tokens* to call specific methods.

The next sections explain each step of the ?? process and will be accompanied by an example to elucidate how it works. The represented signal is the z axis of an accelerometer

placed in the wrist of a subject while performing a rotating task in different angles. The purpose of this example is to find the time intervals where the plateaus above a certain threshold begin to decrease. Each step of this problem's resolution is shown in Figure **??** and will be thoroughly explained throughout the next sections. We take the liberty to start introducing some of the processing tokens (Sm is the smooth function, A is the Connotation Threshold and $D1$ is the Connotation Derivative) to give a base complete example of the usage of the ??.

7.1.1 Pre-Processing - Preparing the Data

The pre-processing stage is responsible for preparing the signal, either by removing noise that arises from several sources or adjusting the signal so that the information is unveiled. Traditional pre-processing methods, such as a pipeline of linear filters, moving window average filters and statistical de-noising or re-sampling techniques, are typical procedures to prepare the signal for further tasks [**preProcessing**]. The current approach uses a symbolic representation of these techniques, in which each is represented by its corresponding *token*, which can be a symbol or a function name. In order to manage the pre-processing tasks, a string containing a set of tokens and their corresponding arguments is written by following the polish notation [**polishNotation**], in which the token precedes the corresponding argument(s), and each element is separated by a white-space character. The available pre-processing methods to be used with ?? are presented in Table **??**.

In the example of Figure **??**, the pre-processing step uses the following string: "Sm 500", which indicates that a smooth filter using a window with a size of 500 samples is applied to the signal. The resulting signal is showed under the original one. The area sectioned in the plots represents the segment of the signal that will be represented in subsection **??**.

7.1.2 Connotation - The Symbolic Time Series

In semiotics, it is described that the connotative aspect of an image or a sign corresponds to the extraction of meaning (sometimes called the *second meaning*), by the personal interpretation of its traits and characteristics [**connotation**]. The *connotation*step is exactly the same connotative aspect, but of a time series. In this, the interpreter (the analyst) has made his personal analysis of the time series and retrieved the necessary information in order to search for the desired patterns.

The symbolic *connotation* step generates a sequence of *characters* by extracting properties of the signal that are based on a conversion rule the user defines. Each of these conversion rules are designated the *connotation* methods and are responsible for converting each sample of the signal into a *character* or group of *characters* that represent the state of the sample for that conversion rule. Each of the *connotation* methods is related with specific

Table 7.1: List of common ?? pre-processing operators. As input parameters, s is the signal, fc is the cut-off frequency and win_size is the size of the window used (number of samples). The linear filters (HP, BP and LP) have a default order of 2

Name	Input Parameters	Description
HP	(fc)	Linear high-pass filter with cut-off frequency fc
BP	($fc1, fc2$)	Linear band-pass filter with cut-off frequencies $fc1$ and $fc2$
LP	(fc)	Linear low-pass filter with cut-off frequency fc
abs	none	Modulus of signal - $\|T\|$
Mag	none	Magnitude - $\|T\| = \sqrt{T_0^2 + T_1^2 + T_2^2}$
Sm	(win_size)	smoothing of T by a moving average window filtering technique of size win_size
Z_{norm}	(s)	z-normalize the time series $\overline{T} = \frac{T - \mu_{\bar{T}}}{\sigma_T}$, being: $\overline{T}$ the normalized signal, $\mu_{\bar{T}}$ the signal's mean and σ_T the standard deviation of the signal
|	N.A.	Separates the pre-processing methods applied to multiple signals or multiple processing of the same signal

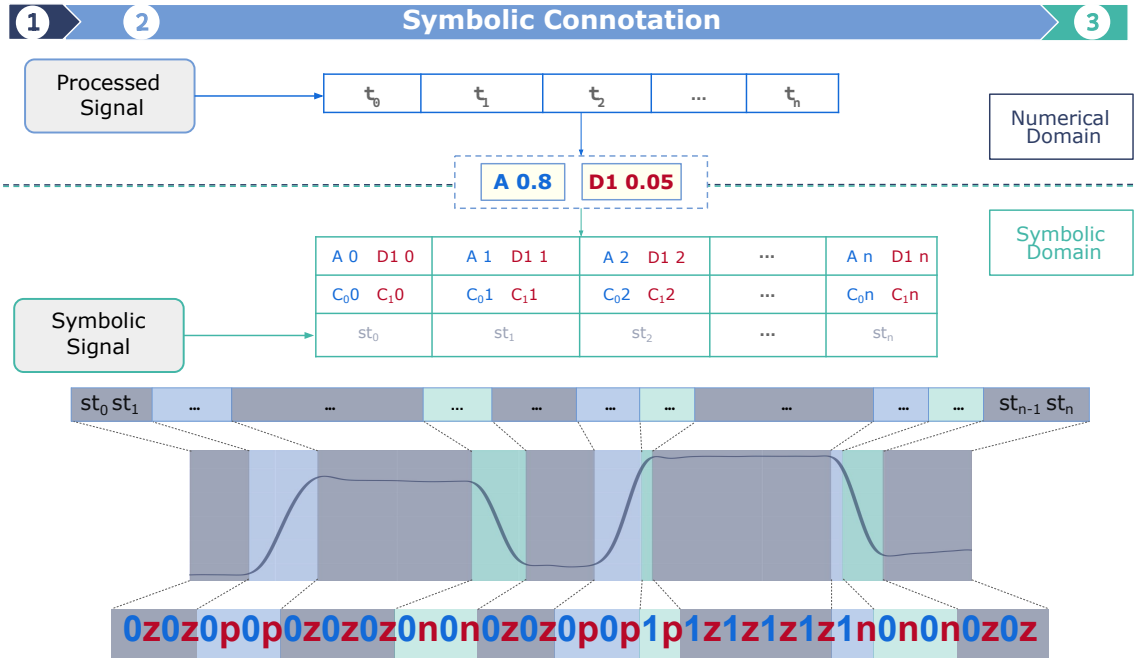


Figure 7.2: SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search.

attributes of the time series that are considered relevant for the search procedure. The string that results from this step is therefore a symbolic representation of the strictly necessary attributes of the signal that are best suited to specify a pattern match. Note that this process is decided by the analyst.

The string can be based on a single connotation method, or the combination of multiple ones. In addition, the *connotation* process also handles multidimensional signals, in which cases the string is a combination of *connotation* methods corresponding to each time series, returning a group of *characters* for each sample of the time series.

The reader may appreciate that we recall the general text definitions from Chapter **??**, and associate these with *connotation* step's output. Each sample of the time series, $T = (t_1, t_2, ..., t_n)$, is converted into a *character* from this *connotation* alphabet ($char_C$), generating a symbolic time series, $ST = (st_1, st_2, ..., st_i, ..., st_n)$, in which $st_i = char_C$. A *character* is any unit symbol from the connotation alphabet ("Group of Symbols" in Table **??**). When the analyst selects more than one *connotation* method to transform the time series T, each sample of T is converted into the corresponding *characters*, one for each alphabet of each *connotation* method applied, resulting in: $st_i = \langle char_{C1}\rangle\langle char_{C2}\rangle\langle ...\rangle\langle char_{Cc}\rangle$, *being* char_{Cc} the c^{th} *connotation* method applied. In case T is a multi-dimensional time series, the symbolic time series (MST)' samples from each time series are grouped $MST = (\langle st_{1,1}\rangle\langle ...\rangle\langle st_{1,k}\rangle, ..., \langle st_{n,1}\rangle\langle ...\rangle\langle st_{n,k}\rangle)$.

The set of connotation methods can be defined by the user and added as needed. A base list of connotation methods and the corresponding symbols used for the examples presented in the paper are listed in Table **??**.

The symbolic connotation step of the exercise of Figure **??** demonstrates the latter formalism. In Figure **??**, the processed signal is decomposed in a symbolic sequence by two connotation methods: amplitude (blue) and derivative (red). The first implies that the sample values superior to the threshold ***0.8*** will be "1", while the rest turns "0", whereas the second gives the value "p" when the signal is increasing, "n" when decreasing and "z" when stationary with a threshold of ***0.05***. Each of the samples of the signal is converted into the primitive $(st_1...st_n)$ which is the alternate association of the two connotation methods. The approximate result is illustrated by the bottom string, in which can be seen the alternation property and what it translates. The representation made by means of these two methods is expected to be the necessary to ease the search procedure.

7.1.3 Expressive Syntactic Search

The last step of the process involves searching for the desired pattern in the generated symbolic time series with a regular expression. This string is governed by the rules of regular expressions from the *alternative regular expression Python module* [**rgxPy**] and benefits from all the functionalities and meta-characters typically used with this tool, which can be recalled from Chapter 2.The search procedure returns the intervals at which positive matches occurred. It is relevant to note that it has been decided to use ??, although

Table 7.2: List of base ?? connotation operators. The input parameters are s, which represents the input signal and *thr*, which defines the threshold percentage value of the amplitude range of the signal ($max(s) - min(s)$) for a given connotation method. The operator that separates the connotation methods applied to multiple signals or multiple representations of the same signal is the vertical bar "|".

Name	Input Parameters	Group of Symbols	Description
A	(s, *thr*)	["1","0"]	Amplitude comparison, If t_i > *thr**($t_{max} - t_{min}$), t_i is "1", else t_i is "0"
D1	(s, *thr*)	["p", "n", "z"]	Derivative of signal s (s'). If t'_i > *thr*, t'_i is 'p'; elif t'_i < - *thr*, t'_i is'n'; else, t'_i is "z".
D2	(T, *thr*)	["D", "C", "z"]	Second derivative of signal T (T''). If t''_i > *thr*, t''_i is 'D'; elif t''_i < - *thr*, t''_i is 'C'; else, t''_i is "z".
AD	(s, *thr*)	["1", "0"]	Determinates if amplitude from a minimum to a maximum and vice-versa is superior to *thr*. If so, t_i is "1", else, t_i is "0".
SA	(s, *thr*)	["r", "R", "f", "F"]	Amplitude of a slope segment. The characters are case sensitive, being ***r*** for a low rise and ***R*** for a high rise. The same for falling (***F*** or ***f***).
SS	(s, *thr*)	["r", "R", "f", "F"]	Speed of the signal measured by the amplitude on the first derivative. When the sample is quicker than the threshold it is converted to "R" - quick rise or "F" - quick fall, whereas when slower it is converted to "r" - slow rise or "f" - slow fall.

any other parser, conveniently combined with the previous steps, might be used for the same purpose.

In the example of Figure **??**, the purpose is to determinate when a plateau superior to *80 %* of the amplitude range of the signal starts to fall, which based on the string representation of the second step is indicated by a "1" and a sequence of stationary ("z") to falling ("n"). The regular expression used for the search is precisely *z1n*, which indicates that the signal, at some point superior to the threshold will start to decrease.

In Figure **??** is presented this reasoning. The next chapters will provide some examples, based on real data, in which each of the steps will be thoroughly explained in order to fully understand the capabilities of the ?? tool.

In this study, we propose a tool that focuses in ease simple query search tasks in time

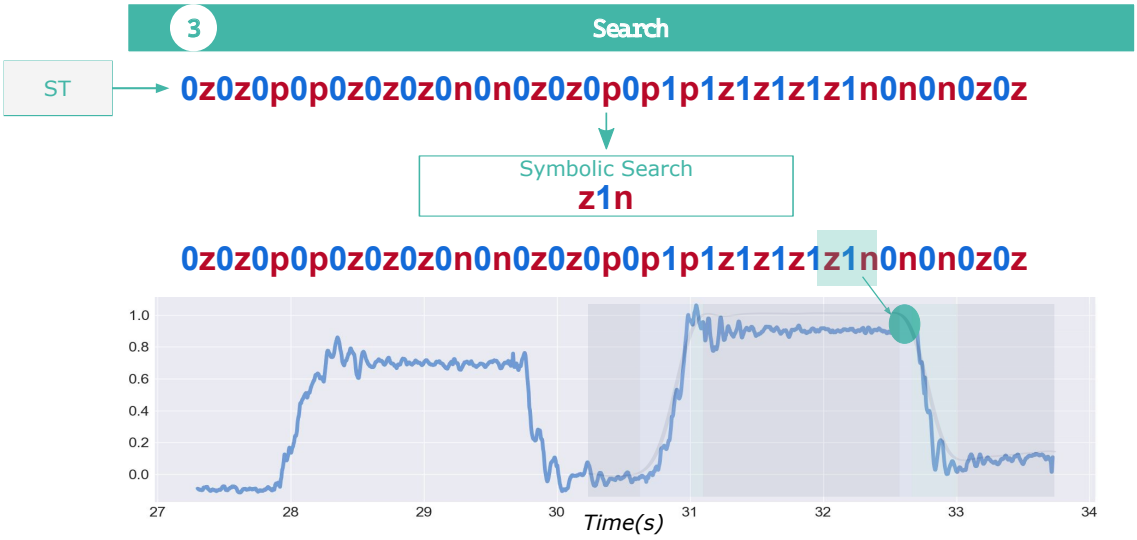


Figure 7.3: SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search.

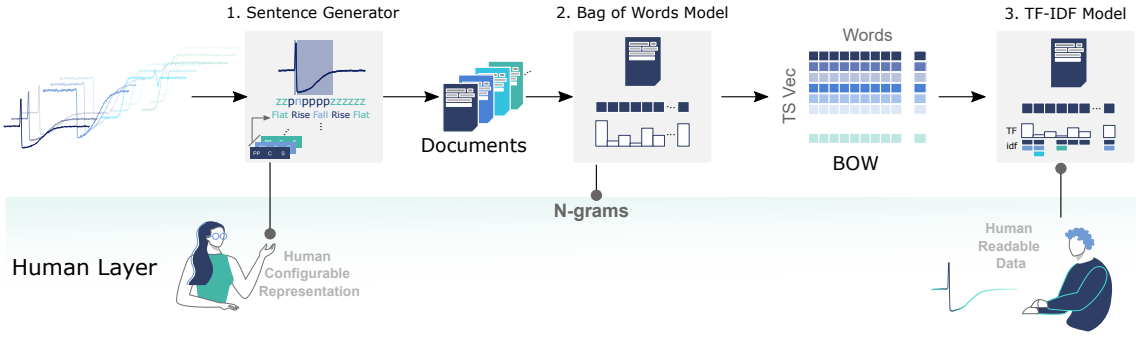


Figure 7.4: SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search.

series, which we refer as ??. This is achieved by an innovative methodology, where the user gives a syntactic nature to time series, which turns the search procedure less verbose and more related with the reasoning of the user in recognizing the desired pattern.

7.2 Towards Interpretable Time Series Classification with SSTS

The ability to transform a time series into a meaningful symbolic representation makes the usage of text mining tools available for time series analysis. This means that the analysis process can be different and complement traditional methods or even bring novel strategies by thinking in how to solve it differently. What is proposed in this section is to use this novel representation to profit from the text mining knowledge and use it into time series classification. In order to perform a class separation, differences and similarities have to be highlighted, and this is also possible to be done with text.

In text mining, the difference between text *documents* is traditionally performed with a feature extraction process on text, such as ?? or ??, returning a statistical representation of the words or ??. *Documents* with different words and sequences of words will have different weights on these models, which is used to perform a class separation on the *documents*. In this work, we take inspiration on this process and apply it to time series. Figure **??** shows the main steps to perform a transformation of the time series into a ??/?? model that can be used with a classifier.

In this section we explain each step of the proposed strategy to perform time series classification based on a textual description with ??. The fact that the time series are translated into text also provides a layer of readability and interpretability on the data, as we will explore throughout this section.

7.2.1 SSTS to Generate Time Series Documents

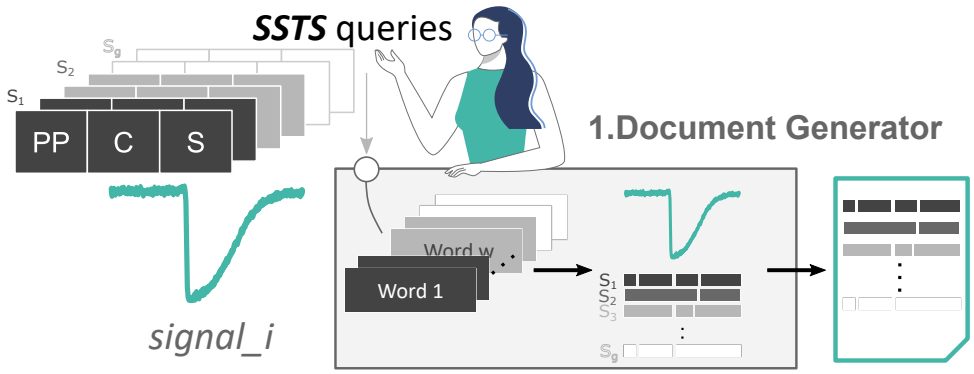


Figure 7.5: Steps for the generation of sentences from a raw time series and organization as a document. Here: PP - Pre-processing, C- connotation and S - Search are each ?? queries used to search for the patterns and attribute the matched *subsequences* the corresponding *word*.

Samples of the signal are converted to characters, patterns are searched to form words, and these are ordered by their time index. With this, the time series can be translated into sentences and be described just as a human would describe it. For instance, this signal can be very broadly translated into `Flat Rise Fall Rise Flat` or `Flat Peak Valley Flat`, while this signal would be translated into `Flat Fall Rise Flat` or `Flat Valley Flat`. This type of description is easily performed with ??, as showed on Figure **??**.

In order to perform this description, the words have to be associated with a specific *?? query*. Each query is a ?? pattern that searches the match on the time series and attributes it its corresponding *word*. For instance, the query {PP: Sm 25, C: D1 0.05, S: p+} searches for moments where the time series is increasing and attributes to these *subsequences* the word *Rise*. A specific set of ?? queries are used to build a full description of the signal dynamics in higher leveled structures, mostly with amplitude and derivative *connotations*. The selection of ?? queries is made by the analyst, either using pre-defined set of queries

or creating his/her own, more appropriate for the type of signals to analyze. This is the customizable step of the process, in which the analyst can include his/her intuition.

The ability to distinguish time series with text will be as good as the descriptive richness of the generated sentences. Depending on the differences between the time series being classified, a simple description based on the sign of the signal's derivative might be enough. However, in some cases, the overall dynamic of signals might be dissimilar in other dimensions. In order to have a rich foundation to perform a sentence description of a time series, we created the set of ?? queries presented on Table **??**, grouped by sentence ($S1, S2, ...$). The table also has an example of matching the corresponding query on a real signal. We would like to highlight that these queries are not mandatory and can be adapted by the analyst or event new connotation methods and queries can be created that might make more sense for the problem being solved. In cases where the user knows exactly what kind of patterns are relevant to perform the distinction, specific patterns can be used, targeting the relevant differences. Otherwise, the existing list can be used.

An example of performing this analysis on a signal is presented on Figure **??**. The process highlights the usage of ?? queries from two different groups of sentences. From the first group, the signal is analyzed in terms of derivative, matching parts of the signal where it *rises*, *falls* or is *flat*. A sentence is generated, describing it as `Flat Rise Fall Rise Flat`. The same process is done now with the second group of ?? queries, but this time, searching for *peaks, valleys and plateaus*. A sentence is then generated based on the matches: `Peak Valley`. The same process is applied to each class of signals, being, for each signal, generated a *document* with the corresponding sentences.

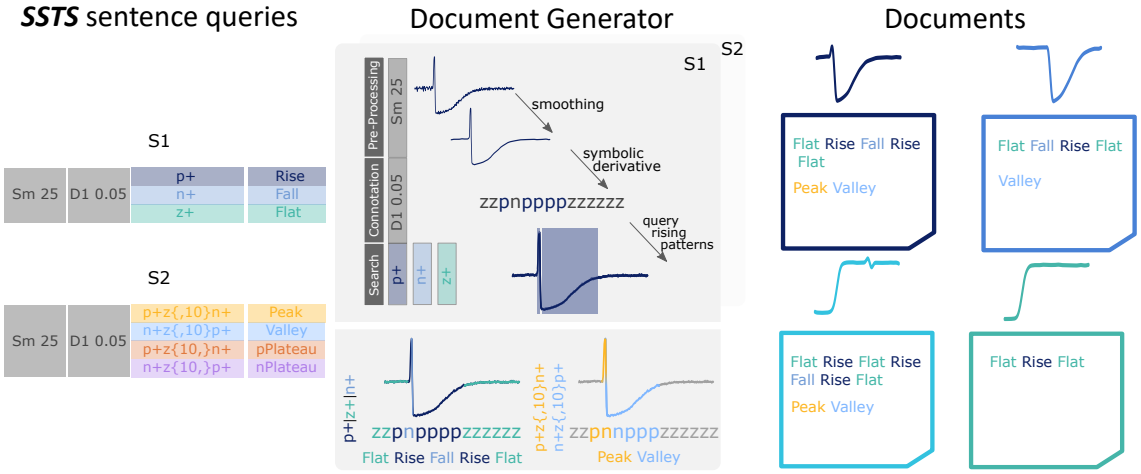
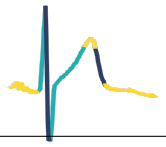
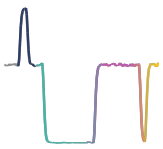
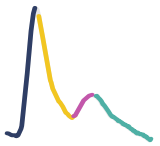
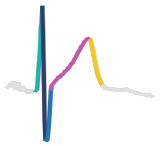
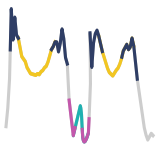


Figure 7.6: (Top) Using SSTS to detect the rising stage of a time series. Each step of the process is written described as follows: (1) pre-processing: *Sm* is the function *Smooth* with a window size of 25 samples; (2) connotation: *D1*, indicates the first derivate, from which each sample is converted to *z* - Flat, *p* - rising and *n* falling; (3) search - regular expression *p+* searches for all sequences with 1 or more *p* characters. (Bottom) Example of sentence generation. Using the other search queries (*p+*, *n+*, *z+*), we can find the derivative patterns and convert it into ordered words.

Having now a method to translate time series into text, we are able to use text mining

Table 7.3: The connotation variables, search regular expressions and corresponding words assigned to the pattern searched. The parameter m indicates the size, in samples, of the difference between a peak or a plateau.

Sentence Group	Connotation	Search	Word	Example
S1	D1 thr	p+	Rising	
		n+	Falling	
		z+	Flat	
S2	D1 thr	p+z{,m}n+	Peak	
		n+z{,m}p+	Valley	
		p+z{m,}n+	posPlateau	
		n+z{m,}p+	negPlateau	
S3	SA thr	r+	smallRise	
		R+	highRise	
		f+	smallFall	
		F+	highFall	
S4	SS thr	R+	quickRise	
		r+	slowRise	
		F+	quickFall	
		r+	slowFall	
S4	A 0.5 D1 thr	(0p)+(0z)*(0n)+	bottomPeak	
		(1p)+(1z)*(1n)+	topPeak	
		(0n)+(0z)*(0p)+	bottomValley	
		(1n)+(1z)*(1p)+	topValley	
S5	D2 thr D1 thr	(Dp)+	concaveRising	
		(Dn)+	concaveFalling	
		(Cp)+	convexRising	
		(Cn)+	convexFalling	

methods directly on the *documents*. Typically, text data is vectorized with the ?? or ?? models. This brings the reader to the second step of the process.

7.2.2 Vectorization of Time Series Documents

The document generation stage receives a time series and transforms it into a *document* with several sentences, descriptive of its shape and dynamics. As mentioned, this transformation is made with *SSTS*, which searches for specific patterns on a symbolic representation of the signal and for which a word is given.

After converting the time series into *documents*, the text is analyzed to build a matrix of *word* or *n-gram* frequencies, the ?? model. In this case, the features extracted are purely statistical, but can provide a relevant measure of differences between documents. Each row of the ?? model is a time series *document*, represented by columns that have the number of occurrences of a *word* or *n-gram*. Figure **??** shows the vectorization of a document as one row of the BoW model.

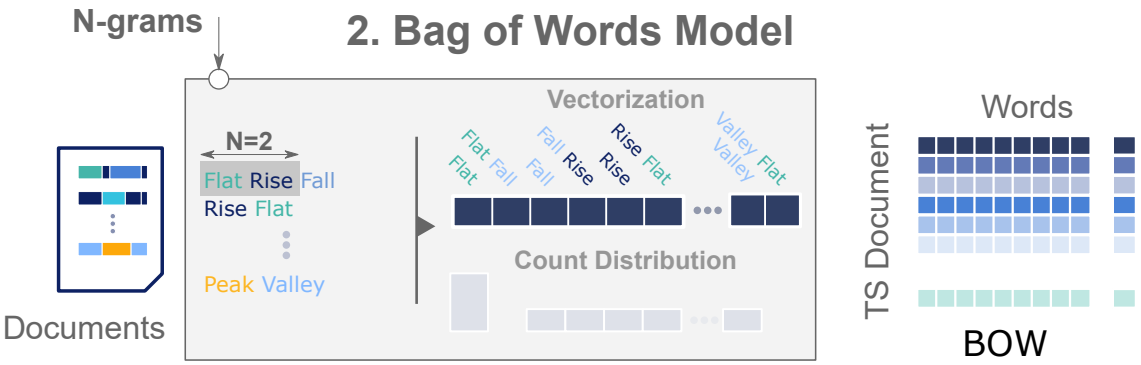


Figure 7.7: Steps for the generation of sentences from a raw time series and organization as a document. Here: PP - Pre-processing, C- connotation and S - Search are each ?? queries used to search for the patterns and attribute the matched *subsequences* the corresponding *word*.

The ?? model can be used in several ways. Following guidelines from the text mining domain, it can be used for unsupervised tasks as well as supervised ones, for topic modelling and keyword extraction. The ?? model can also be trained with Naive Bayes Classifiers, Support Vector Machine or Linear Regressors [**scikit-learn**]. In addition, the ?? model feature matrix can be converted into the ?? feature matrix, which can then be trained with the same classifiers. In this work, we explored several of these combinations to understand which can give better results.

As was pointed out in Chapter

Both the ?? and ?? models give a weight to each *word* for each *document*. This means that each time series document is vectorized into a distribution of weights for each *word*, according to its presence on the time series document and, in case of the ?? model, all the other time series documents. The difference between the distribution of each time series document can be used as a distance measure to compare them. In Figure **??** is showed the

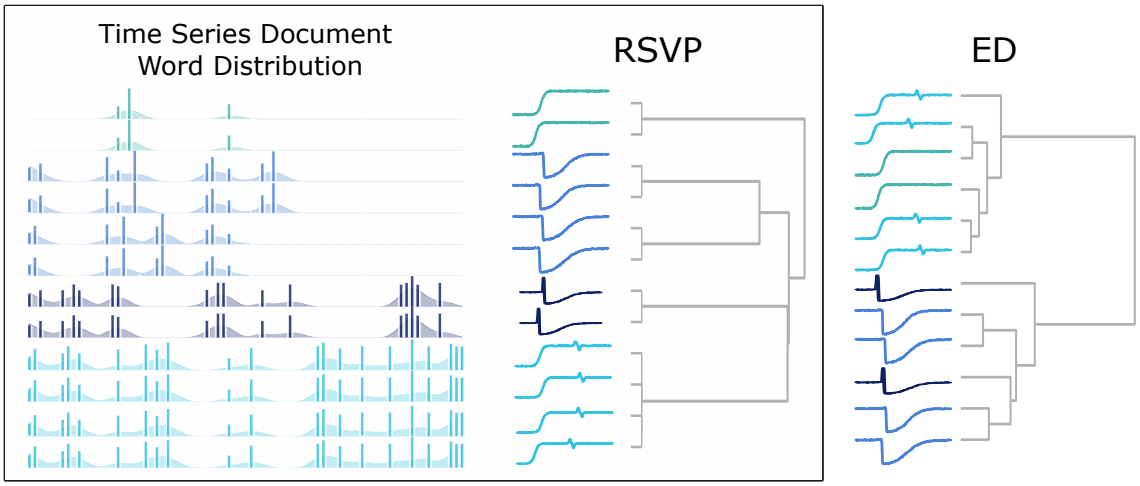


Figure 7.8

distribution of *word's* weights for four different time series documents of the *Trace* dataset, based on a ?? model. The x-axis of each distribution is the *word* or *n-gram* and the y-axis, the corresponding weight for a time series document.

The reader can notice that each distribution is different and this provides a way of distancing them. On the right of the distribution, the reader can see a dendogram of sorting examples from the *Trace* dataset. We used the cosine distance between vectors of time series documents (RSVP) and compared it with the euclidean distance (ED). Surprisingly, the ?? misses a few matches. Although the example is very simple, it highlights well how this strategy can compensate for some mismatches that can occur with the ??.

As the reader can notice, the ?? mismatched signals from class 1 with signals from class 2, and signals from class 4 with signals from class 3. The fact that a misalignment between the main structures of the signals occur (the main peaks from signals of class 1 are misaligned and the main small peak and valley shapes of class 3 are slightly misaligned), increases the distance between signals that apparently have the same shape. In the other end, the proposed strategy follows a distance measure based on the presence(absence) of specific shapes, as well as how these are ordered (if *n-grams* are used). This means that classes 1 and 2 will not be mismatched, because signals from class 1 have a peak whereas the ones from class 2 don't.

7.2.3 Towards Interpretable Results

Having demonstrated that it is possible to create a distance measure between time series with pure text, we highlight an advantage of working on the text domain, which is *interpretability* of the data. By *interpretability*, we mean that it is possible to extract meaning in what differentiate classes of signals. This advantage is taken because of the weighting factor attributed to each *word* or *n-gram* from the ?? model. As explained by Pavel Senin *et. al*, the weights can be used as a weighting vector for each word extracted, highlight the corresponding shape on the original signal and measuring its importance for the

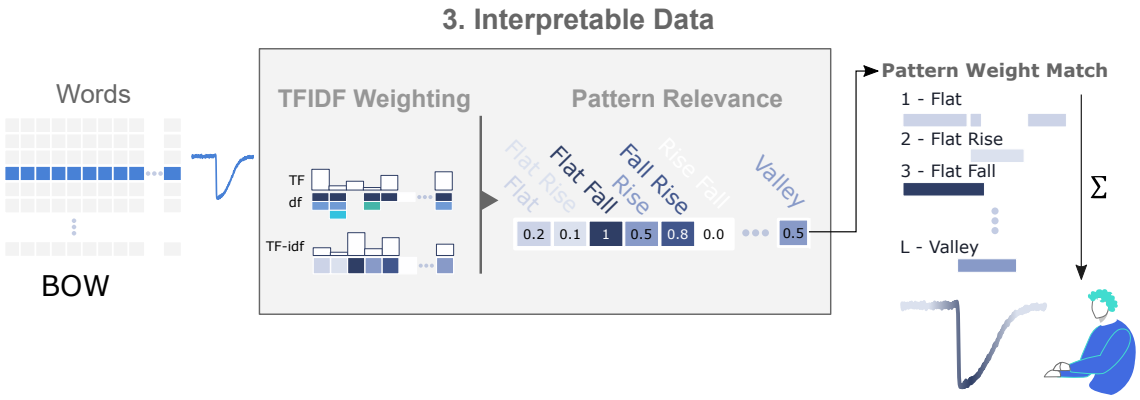


Figure 7.9

classification process [**sax_vsm**].

The process to highlight the areas of the signal that are more relevant and able to explain the difference between classes is illustrated on Figure **??**. From the ?? model, the ?? model is computed, following equations **??**, **??** and **??**. From this representation, each *time series* is represented as a vector of weights for each *word*. These weights can be used to highlight the area of the signal represented by the corresponding *word*. Therefore, the process is to search back the areas of the signal that match the *word* or *n-gram* (w_i) as an ?? query, while keeping the corresponding *weight* (h_{ij}), for word i from time series document j, on the indexes of the match, with indexes a and b:

$$(a, b) = ssts(t_j, w_i) \tag{7.1}$$

In the end, the weighted vectors are summed, returning a final vector with the weighted contributions of all *words*:

$$WTS_i(a, b) = WTS_i(a, b) + h_{ji} \tag{7.2}$$

On Figure **??**, the exemplified signal has a higher relevance on the `valley`, being the most characteristic element of that time series.

7.2.4 Experimental Evaluation on Selected Use-cases

7.3 Towards Natural Language for Pattern Search

The fact that an analyst is able to search with ?? on time series makes the process more flexible and expressive and is innovative in the sense that text patterns can be written to search for specific shapes on a time series. Nevertheless, this process requires that the analyst has some background knowledge on ??, which is fine for a computer scientist, but might be harder for an analyst outside of the computer science domain. In order to contribute for the democratization of pattern search on time series, we propose another method for pattern search, but in this case, it is based on natural language and

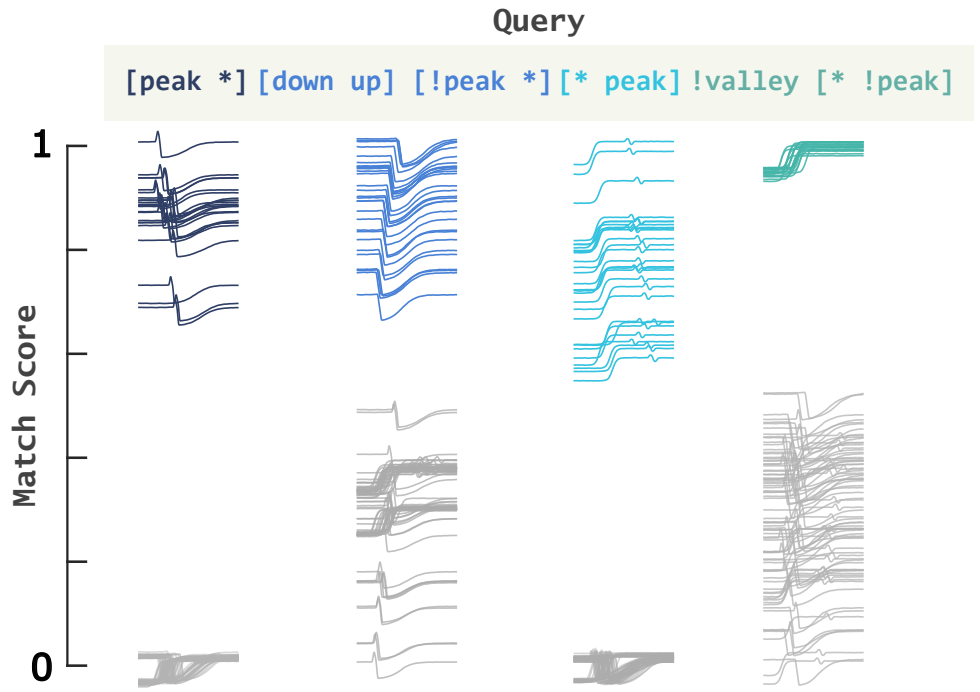


Figure 7.10: Using our proposed query language, we can create short, intuitive natural language queries to rank the 100 exemplars in Trace, separating our sought class from the remaining data. Here $\star$ is *anything*, ! is *not*, and *square brackets* are a grouping operator (more details in Section 3).

linguistic operators, from a feature-based representation of the signal instead of a symbolic representation of it. The process to find a pattern on a time series is very similar to how a user searches for *web-pages* on the *Google Search Toolbar*, using *keywords* and *operators*.

7.3.1 "Googling" Time Series Patterns

When a user opens *Google* and searches for a specific *webpage*, he/she will write a set of keywords that can match the *webpage* he/she is searching. The results of this process are a list of pages ranked based on how well it matched the *keywords* used. The reader will agree that it often matches well what the user had in mind. Of course, *Google* might not solely rely on the *keywords*, also including geolocation, browser history, etc... on the search process. However, let us take into account this simple *keyword* based search process and apply it to pattern search on time series.

As we mentioned above, patterns can be described with *words*, which is done because we associate these *words* with specific properties or the presence of specific shapes on the pattern. In that sense, we propose to create a search paradigm that relies on words represented by features extracted from the signal. Informally, if a user wishes to find examples of a particular behavior, he/she can simply *Google it*, by describing the data he/she expects to see, given that behavior. For example, we considered the four-class Trace dataset, which has been studied in more than 1,000 papers. As Figure **??** shows, it is possible to use our system to create queries that can separate each of the four classes

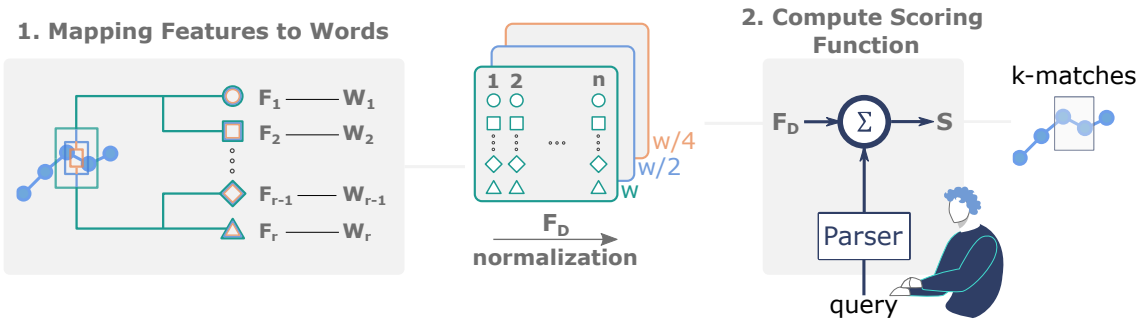


Figure 7.11

from the rest of the data. This separation is possible simply by describing with *words* the presence/absence of structures. Further we will explain in detail how the process is done, which are the operators that we designed and how queries can be written. Still, for now, we will explain the meaning of each query of this example. For instance, signals from class 1 () match well with query `[peak *]`, which translates into *has a peak on the first half and anything on the second half*. The method sorts the signals based on this query and gives a very low score for all signals that do not have a peak on the first half of the signal. The same happens for the other queries, used to sort the other classes. A similar result is achieved with the inverse query `[* peak`, which means *anything on the first half and a peak on the second half*, as well as the query `!valley [* !peak]`, which translates into *has not a valley and has no peak on the second half*.

The overall steps of the process are illustrated on Figure **??**. In order to perform the search with natural language and operators, we represent the time series into a set of features series. The features are extracted with a *moving window*, with total overlap. The process is made three times, with decreasing window sizes. Each feature is associated with a *word* (W) that should give a good intuition about the property it represents (e.g. `up` or `rising` should be clear to be indicative of the signal being rising). This feature set is the weight of the *word* for each *subsequence* of the signal. Such that when the *words* are written, the features are summed according to the operators used. Then, the search returns the *k-top subsequence* matches for the query used.

Concretely, with this method, we will demonstrate that the proposed natural language representation is expressive enough to discriminate specific behaviors on a time series and that it moves towards a democratized time series analysis process, where analysts from other scientific domains, not familiar with programming languages, can also perform high-level pattern search.

7.3.2 Mapping Features to Words

We define a *word feature vector* W as a mapping of a feature series F_i with a specific *word* W_i. With feature, we intend to describe either a property, such as the mean or standard deviation, but also a distance measure to pre-defined examples. In linguistic terms, this

word feature vector is an adjective of the *subsequence*. Every *subsequence* $t_{i,m}$ from each time series T is characterized by the selected set of *words*, being created a set of *word feature vectors* for each time series, further normalized between 0 and 1. The *word feature vector* has the size of T and the feature series values indicate how relevant is the *word* in the *subsequence*. Figure 3 shows an example of the word feature vectors for up (Fup) and down (Fdown).

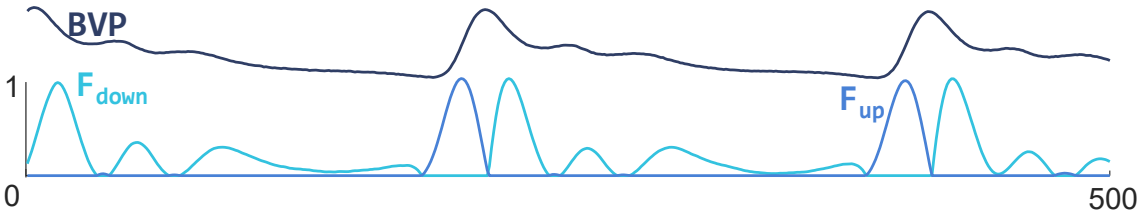


Figure 7.12

To allow interactive search, the *word feature vectors* are pre-computed in an offline indexing stage. In addition to this, we also extract three dimensions of the same word feature vector with different window lengths, based on w: $W_1 \rightarrow w$; $W_2 \rightarrow \frac{w}{2}$; $W_3 \rightarrow \frac{w}{4}$; with the intent of matching ordered sequences of words inside a subsequence with the grouped followed by operator, as will be explained further. It is important to note that the ratio of the window used for the dimension W_2 and W_3 might have to be fine-tuned for the domain, as there might not be a "one window ratio fits all". However, empirically we observed that the exact value of this ratio is not critical to the success of the search.

For each W is assigned a *word*. We use English words to make the process more intuitive, such as `noise`, `up` and `peak`. We recognize that the intuitive meaning of such words can vary from user to user depending on their domain, their experience and on the current context. Either way, words can be mapped to features that are domain specific or word feature vectors can be given a domain specific vocable, providing a more appropriate mathematical thinking behind what is its meaning. In addition, we are aware that multiple words can be given the same meaning and for this reason, we associate several synonyms to each word.

We define an initial subset of features that are mapped to words. When defining one *word feature vector* it would often come with a negation pair. A negation pair (!W) represents the exact opposite of a defined *word feature vector* (W) following the rule:

$$!W = 1 - W \tag{7.3}$$

This indicates that when one increases the other one has to decrease proportionally. Examples of such *word feature vectors* are `symmetric` and `asymmetric`, or `complex` and `simple`. Note that some words might be the opposite of each other, but do not follow this rule, or even seem to be the opposite of each other, but are not. For instance, `up` and `down` are opposite of each other, but do not follow Equation **??**. While one exists, the other

can not, but it does not mean that when one is small, the other has to be high, since the *subsequence* might just be `flat`. Another case is the word `peak`. Intuitively, we would think that `valley` is the opposite of `peak`, but the consideration of `!peak = valley` (not being peak means it is a valley) is false. In this work, we use the negation of a word feature vector for cases where there is no negation pair. This negation is realized using an operator (See Table **??**).

As previously noted, a set of features is used to extract several properties of all *subsequences* of a time series and attribute a semantic meaning to each one of them by mapping it to a specific *word*. It is our assumption that a *subsequence* can be mapped to a set of *words* that an analyst would use to describe it. Depending on the domain or *vocabulary* of the analyst, the set of words might have to be different and adjusted. Eventually, the *dictionary* can be expanded to other types of *features* and *words*. In any case, we want to demonstrate that this current set of words and operators can solve many search problems with expressive queries.

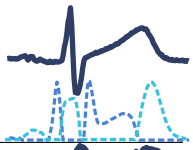
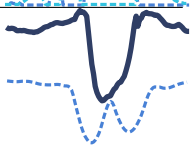
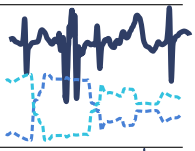
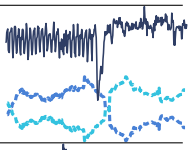
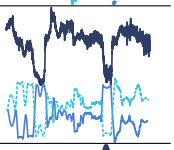
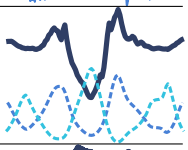
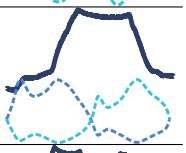
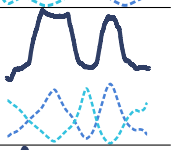
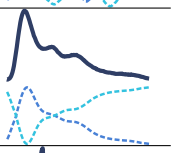
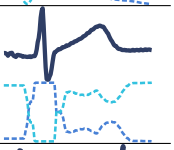
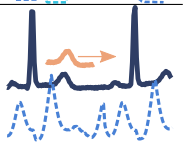
7.3.3 Linguistic Operators

The same way we use word and sentence connectors in our language to create contrast or attribute a temporal sequence, in our proposed system we use *operators*. An operator is a metacharacter or a word that can be used to diversify the way word feature vectors are handled, either in the way the information is extracted or how these are combined. It contributes to a more versatile and expressive usage of this language. Currently, we have a simple list of four operators: *negation* (`!`), *wild card* (`*`), `followed by`, and `grouped followed by` (e.g., $[W_1 W_2 \ldots W_w]$. This list can obviously be expanded and customized, but we want to demonstrate that with a minimal set of operators, most of the problems we present are solvable.

Web search engines have many operators at the user's disposal, but since a list of words is usually powerful enough to retrieve and correctly sort most of the desired results, very few (or none!) are often used. We believe that this is the case for this application as well but acknowledge that simple operators can make the query more natural and come in handy to perform conjunctions between features and multiple dimensions, such as *temporal logic* or *negation*. For instance, we often rely on *temporal logic* to express the presence of a shape in regards to another: `the peak that comes after the valley`, or even use the absence of a property: `it does not have a peak`. We also typically express the order at which structures occur on a time series: `up and then down`. These operators are especially useful to close the gap between the query and human discourse, contributing to a more expressive mechanism when using the proposed language. Currently, four operators are available. Below is a list and description of each of them, starting with the negation operator.

- **Negation Operator - `!W`** : As mentioned above, most words come as an opposite pair, but some do not follow Equation **??**. In these cases, or when the word has no

Table 7.4: List of all word feature vectors with examples. Here, $i \in [0, n]$, n is the signal's size, a is the beginning of the moving window, starting at $a = i - \frac{w}{2}$, and w is the moving window size.

Word	Description	Example
Up (**Down**)	The slope estimation of a linear adjustment ($y = ax+b$) to the *subsequence*, being `up (down)` $= a$, if a_{up}>0 (a_{down} <0) or `up (down)` $= 0$, if a_{up}<=0 (a_{down} >=0)	
Flat	The inverse of the sum of absolute differences between the *subsequence* and its average: `!flat(i)` $= \Sigma_{j=a}^{a+w} \lvert t(j) - \overline{t(a, a+w)} \rvert$.	
Noise (**Smooth**)	The residual error when modeled by a moving average (ma). `noise(i)` $= \Sigma_{j=a}^{a+w} \lvert t_j - ma_j \rvert$. (`smooth` = `!noise`).	
Complex (**Simple**)	The complexity-invariant distance measure of the *subsequence* of **??**. (`simple = !complex`)	
Symmetric (**Asymmetric**)	The normalized ?? (**??**) to the *subsequence*'s horizontally flipped self.	
Peak (**Valley**)	The logarithmic normalized ?? (**??**) to the template of a peak (valley), modulated by a gaussian function.	
StepUp (**StepDown**)	The logarithmic normalized ?? (**??**) to the template of a step-up(down) function.	
PlateauUp (**PlateauDown**)	The logarithmic normalized ?? (**??**)to the template of a plateau-up(down) function	
Top (**Bottom**)	The moving average of the time series: $ma(i) = \frac{1}{w}\Sigma_{j=a}^{a+w} t(j)$. (`bottom = !top`)	
High (**Low**)	The difference between the maximum and minimum value of a *subsequence*: `high(i)` $= max(t(a, a+w)) - min(t(a, a+w))$. (`low = !high`)	
Shape	The normalized ?? profile of the time series with a query (e.g. in orange) given by the user as an example. A word must be given as well and integrated on the query language.	

direct opposite, it can be useful to penalize the presence of the word in a *subsequence*. This operator does that by applying Equation **??** to the word feature vector, W.

When describing time series, we inevitably use temporal logic in explaining the sequence of shapes we perceive. The next operator is `followed by`.

- **`A followed by B`**: This operator rewards a *subsequence* represented by A followed by one *subsequence* that has a high score for B, within a distance of size w. A and B can be single words, multiple words or even queries for different dimensions of the time series.

  With this operator, we look ahead of a *subsequence* in the time series. However, in some cases, it might be useful to describe the sequence inside the limits of the window we defined. For these we have a special case of `followed by`, which is the `grouped followed by`, represented by square-brackets ([ ]).

- **Grouped `followed by` -** $[W_1 W_2 \ldots W_N]$: Instead of looking ahead in the time series, we look inside the *subsequence* to reward an ordered sequence of words. In this special case, the *subsequence* is segmented into N sub-windows, with size integer of $\frac{N}{m}$, and the corresponding *word* is scored within this sub-window. For this, we use the other two dimensions of the word feature vectors (W_2 and W_3). If $N <= 3$, W_2 is used, while if $N > 3$, W_3 is used.

  Often, within a *subsequence*, the differentiating property occurs on the first half, last third or another sub-window of the *subsequence*, while the remaining sub-window(s) is(are) not relevant. We therefore introduce the wildcard ($*$) operator.

- **Wildcard - *:** The sub-window where * is used is valued equally for all *subsequences*.

As with vocabulary, the reader could imagine expanding our dictionary of operators, but even with a limited set of them, we are able to successfully solve all the proposed search tasks, which cover dozens of examples. After presenting the set of elements that can be used in the proposed method to query a pattern of interest, we are ready to explain how the query is turned into a score function and finally how the selection of the k-most relevant *subsequences* is done.

7.3.4 Natural Language Query for Time Series

As illustrated on Figure **??**, the user inputs a query with all the *words* and *operators* available. The query is parsed to calculate a matching score (S) that indicates which *subsequences* are better represented by the query. The score is calculated based on the word features vectors extracted from the time series that are *called* by the query.

As mentioned, considering the possibility of using the `grouped followed by` operator, each feature is extracted three times with a different window size. For each word, three sets of word feature vectors are stored ($W_1, W_2 and W_3$) based on the original window

size (w, $\frac{w}{2} and \frac{w}{3}$). These word feature vectors are stored together in a *dataframe* and called based on the *word* written by the user. It is important to mention that all words are stored in a vocabulary file, associated with a *thesaurus* file for synonym checks.

If the signal is multidimensional, all three sets of *word feature vectors* are extracted for each dimension. Having this set of information pre-computed helps make the search run at interactive speeds, even for large data collections. When all data is pre-computed, ?? is ready to accept queries by the user.

The query field accepts any word available in the vocabulary and *thesaurus*. When any of the words are not present in our vocabulary, we alert the user which is the closest word available, based on *edit distance*. The query can accept operators and works for multidimensional querying. These are relevant elements that are used as a reference to parse the query into individual scoring elements. This parsing process is made by looking into: (1) which dimension(s) of the time series is (are) included; (2) which operators are used; and (3) the single written words. The first two define how the score is calculated, that is, how word feature vectors are combined, as well as which dimensions of the word feature vectors are used. For instance, when including multiple signals, the query parses which word(s) corresponds to which signal, to search for the correct index of word feature vectors in the pre-computed *dataframe*. To identify the signal, the following regular expression is used to find all signal names *\w+:* (*one or more characters ending with ":"*).

When the `followed by` operator is used, the query is parsed in the *elements* that come before and after it (this is applicable either if the operator is used for intra-signal or inter-signal search). For this, the following ?? is used to separate the *words* before and after the operator: *\w+ followed by \w+* ((word or words before followed by and word or words after it). What is meant by *elements* is either *words* (intra-signal) or two different signal dimensions (inter-signal). Another temporal operator is the `grouped followed by`, which is parsed by identifying square brackets in the query with the following ?? *\[.+?\]* (*anything inside square brackets*). The content of the square brackets is then combined (see Figure **??**).

When any of these elements are parsed, we end up with a single word or sequence of words, which are combined by summing their corresponding *word feature vectors* (this corresponds to an implicit OR). Figures **??** and **??** give a visual aid to understand the parsing process for each case.

The simplest case for a query would be a sequence of *words*. In this case, each *word feature vector* , associated to the corresponding *word*, is a score (S) function and are summed together to give the final score. In case a multidimensional signal is used, the user can write a multidimensional query, identifying a specific query (Q) for each dimension ($s_1, s_2, ...s_k$). For this, each query Q corresponding to a signal s are treated individually by the standard parser (P on Figure **??**), resulting in a scoring function (S) for each signal used on the query. Before summing all scoring functions, these have to be normalized (N on Figure **??**) to have equal weight. It is important to note that Q can be a query with *words* or the `grouped followed by` operator.

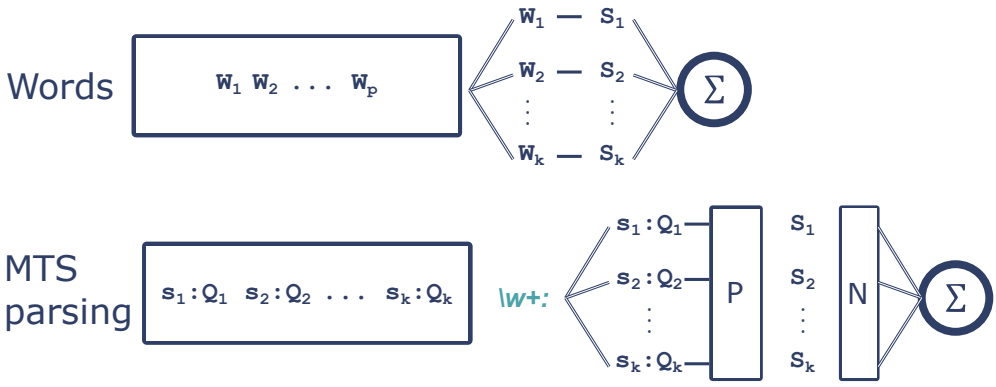


Figure 7.13: •

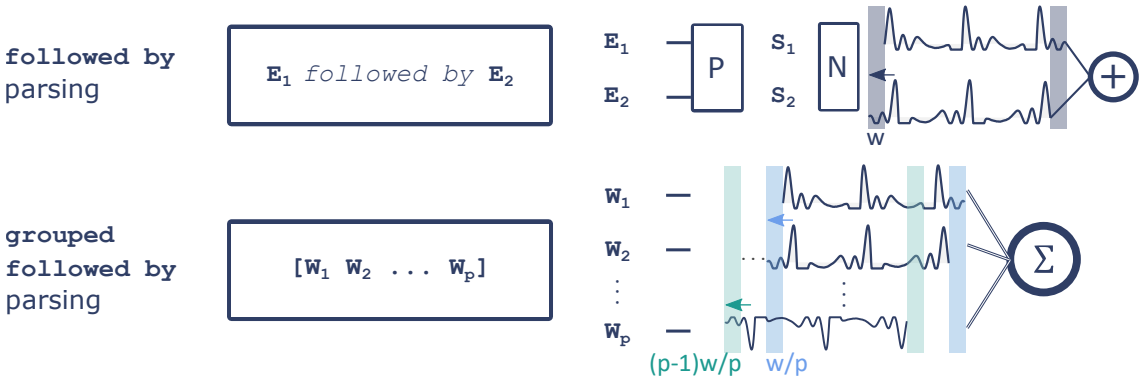


Figure 7.14: •

The temporal operators are parsed by the ??, as mentioned above. From the `grouped followed by` operator, only *words* can be used and are combined with a temporal delay that depends on the number of *words* inside the brackets. The brackets represent the limits of the *window* that is used to extract the *word feature vectors*. In that sense, when using two *words* inside the brackets, it is expected that the first half of the signal has the first word, and the second half the second word. This is extended to additional words inside the brackets. In order to combine the *word feature vectors* of the *words* used inside the brackets, we delay each *word feature vector* based on the number of *words* and corresponding position inside the brackets, such that *words* after the first one are delayed $\frac{(i-1)w}{p}$ samples, being i the position inside the brackets, p the total number of *words* inside the brackets and w the window size. The delayed vectors are added together. For example, if using the query `[up down]` and features were extracted by a window with size of 100 samples, the *word feature vector* for the word `down` is delayed 50 samples and added to the `up` *word feature vector*. When used in combination with individual *words*, the scoring function resulting from this operator is normalized between [0-1]. The reasoning is that each segment of the query should be weighted the same (e.g., if using the query `noise [up down]`, as `up` and `down` are combined, the range of this segment is [0-2], while `noise` is [0-1].

Therefore, `[up down]` is normalized between [0-1] before being added to `noise`). A similar process is performed with the `followed by` operator. The scoring function for the query previous to the operator is added to the scoring function of the query next to the operator, but delayed by one window size (w).

Finally, depending on the *words* and operators used, the resulting scoring function is used to search for the k-top *subsequences*. This is made by searching in k-iterations the maximum value of the scoring function. During this iteration process, trivial matches are not considered. A trivial match is a high valued sample of the scoring function that is a neighbor of the maximum value found. To avoid these matches, all samples of the scoring function around a k-top match on the scoring function ($[argmax(S) - \frac{w}{2}, argmax(S) + \frac{w}{2}]$) are converted to 0. The matched *subsequences* are highlighted on the signal and sorted by their match score.

8

Results and Validation

The previous chapters have been explaining the problems faced and the methods developed to solve them. A few examples were used to follow the explanation of the methods. In this chapter we go further into testing the algorithms with the Datasets from **??** to validate them and provide evidence that these are relevant and contribute to the state of the art of the time series data mining domain.

Considering the diversity of methods developed, this chapter will be divided into each single contribution, providing results for each one of them. We will start with the methods developed under the topics of **??**.

8.1 Novelty Segmentation

In this section, we present several examples in how the *nova* function, retrieved from the ??, is useful for the segmentation of time series. The reader will appreciate that we also provide a measure of the algorithm's performance considering ground truth events from public benchmarks (presented on Chapter **??**, while also comparing our proposed solution with state of the art methods for change point detection from the *Turing Change Point Detection Benchmark* **[cpd_alan]**.

This section is divided into three main categories: (1) Validation in segmentation datasets, (2) comparison with state of the art methods and (3) discussion of the results obtained.

8.1.1 Performance on Public Datasets

The datasets that were used to test and validate this method have categorized labels that were used to generate ground-truth events. These include different contexts (HAR, Hand Posture, Noise Detection, etc...) and different types of data (Inertial data, ?? and ??).

The method has been computed in the same conditions and by following the same procedure for all records of all datasets. The features used have been the same for each record, varying the time scale parameter, the overlap size of the sliding window and the

Dataset	Signals	# Ch	Task	TP	FP	FN	Prec (%)	Rec (%)	F1 (%)
Dataset 1	ACC	3	HACP	98	16	16	0.860	0.860	0.860
Dataset 2	ACC-GYR	6	HACP	157	18	22	0.897	0.877	0.887
Dataset 3	ACC-GYR	6	HACP	1378	313	263	0.815	0.840	0.827
Dataset 4	ACC-GYR	12	HACP	499	71	38	0.875	0.929	0.902
Dataset 5	EMG	8	Act/Rel	309	0	72	1.000	0.811	0.811
Dataset 6	ECG	1	Noise	132	25	10	0.841	0.930	0.883
Dataset 7	ECG	4	Noise	21	2	3	0.913	0.875	0.894
Total	N.A.	N.A.	N.A.	2629	465	386	0.850	0.872	0.861

Table 8.1: Overall results for the performance of the method on novelty segmentation. The dimension of the records is presented on the column # *Ch*, as well as the types of signals used and the task in which applied (HACP - Human Activity Change Point detection; Act/Rel - Activation/Relaxation of the EMG detection and Noise detection).

Dataset	T_s (s)	MAE/T_s	MsE/T_s
Dataset 1	5	0.53	-0.12
Dataset 2	10	0.29	-0.07
Dataset 3	1	0.34	-0.04
Dataset 4	25	0.23	-0.00
Dataset 5	1	1	-0.13
Dataset 6	10	0.12	-0.09
Dataset 7	1	0.17	-0.06
Average	N.A.	0.32	-0.07

Table 8.2: Distance error as a ratio of the time scale (T_s) for the detected TP.

kernel size. The peak detection strategy was the same for all records, which is based on a threshold mechanism. The threshold value varied for each record.

Results for publicly available datasets are presented in Tables **??** and **??**. Table **??** indicates the performance in detecting the change point events.

The results show that the proposed method has a good performance in detecting events, in real, complex, multivariate and diverse datasets. The results are consistent, supporting that the proposed method is reliable to use in multiple types of data and different contexts.

The method achieved an overall macro-averaged 85.0% of Precision, 87.2% of Recall, and a F1-measure of 86.1%. Individually, the worst performance was found on Dataset 3, although all the Datasets are accounted for similar results. A more comprehensive discussion on *FN* and *FP* will be presented in the next section. More details are also provided regarding the time discrepancies between ground truth and estimated events.

8.1.2 Comparison with SOA Methods

In order to compare the proposed method with state of the art approaches we used a benchmark provided by the Alan Turing Institute [**cpd_alan**]. The performance was evaluated by computing the proposed method to search for change point events in each of the time series available. The time scale and threshold values for peak detection were selected empirically and available at []. The results are summarized in Table **??** and Figure **??**. Both cases show the F1-score. The performance considered for other methods was selected as the best score of each method available on the benchmark [**cpd_alan**].

Dataset	Nova	AMOC	BINSEG	BOCPD	BOCPDMS	CPNP	ECP	KCPA	PELT	PROPHET	RBOCPDMS	RFPOP	SEGNEIGH	WBS	ZERO
bank	0	**1.000**	**1.000**	**1.000**	0.500	0.054	0.200	0.333	0.400	**1.000**	T	0.015	**1.000**	0.043	**1.000**
bitcoin	**0.694**	0.507	0.690	0.733	0.533	0.611	0.625	0.665	0.735	0.446	T	0.284	0.735	0.690	0.450
brent_spot	**0.861**	0.465	0.670	0.609	0.239	0.607	0.636	0.553	0.586	0.249	T	0.521	0.586	0.564	0.315
businv	**0.927**	0.588	0.588	0.588	0.455	0.386	0.370	0.294	0.490	0.275	0.370	0.261	0.588	0.289	0.588
centralia	0.984	0.909	**1.000**	**1.000**	**1.000**	**1.000**	0.909	**1.000**	**1.000**	0.763	0.846	**1.000**	**1.000**	0.556	0.763
children_per_woman	**0.879**	0.678	0.663	0.712	0.405	0.344	0.551	0.525	0.637	0.310	0.504	0.246	0.637	0.500	0.507
co2_canada	0.851	0.544	0.856	**0.924**	0.479	0.642	0.875	0.867	0.670	0.482	0.542	0.569	0.872	0.681	0.361
construction	0.933	0.696	0.709	0.709	0.410	0.602	0.709	0.634	0.709	0.324	0.340	0.185	0.709	0.523	0.696
debt_ireland	0.974	0.760	**1.000**	**1.000**	0.892	0.958	0.980	**1.000**	**1.000**	0.469	0.748	0.824	**1.000**	0.538	0.469
gdp_argentina	**0.968**	0.889	0.947	0.947	0.583	0.818	0.889	0.800	0.947	0.615	0.452	0.615	0.947	0.421	0.824
gdp_croatia	**1.000**	**1.000**	0.824	**1.000**	0.583	**1.000**	0.824	0.583	0.824	0.824	0.824	0.400	0.824	0.167	0.824
gdp_iran	0.921	0.696	0.652	0.862	0.492	0.620	0.824	0.734	0.808	0.652	0.737	0.636	0.808	0.576	0.652
gdp_japan	**1.000**	**1.000**	0.889	**1.000**	0.615	0.667	**1.000**	0.500	0.889	0.889	0.889	0.222	0.889	0.222	0.889
global_co2	0.625	0.929	0.929	0.889	0.458	0.667	**0.929**	0.667	0.929	0.463	0.547	0.293	0.929	0.250	0.846
homeruns	**0.933**	0.812	0.829	0.829	0.650	0.650	0.829	0.829	0.812	0.723	0.397	0.661	0.812	0.664	0.659
iceland_tourism	0.652	0.947	0.947	0.947	0.486	0.391	**1.000**	0.486	0.643	0.220	0.667	0.200	0.947	0.200	0.947
jfk_passengers	**0.978**	0.776	0.776	0.776	0.650	0.602	0.651	0.437	0.776	0.354	T	0.491	0.776	0.437	0.723
lga_passengers	0.885	0.561	0.620	0.704	0.563	0.606	**0.892**	0.526	0.537	0.366	T	0.592	0.537	0.674	0.535
measles	0	**0.947**	**0.947**	**0.947**	0.486	0.118	0.080	0.281	0.153	0.391	F/T	0.030	**0.947**	0.041	**0.947**
nile	**1.000**	**1.000**	**1.000**	**1.000**	0.800	**1.000**	**1.000**	0.824	**1.000**	0.824	0.667	**1.000**	**1.000**	**1.000**	0.824
ozone	0.857	0.776	0.723	0.857	0.778	0.750	**1.000**	0.667	**1.000**	0.723	0.651	0.429	**1.000**	0.286	0.723
quality_control_1	**1.000**	**1.000**	**1.000**	**1.000**	0.667	0.667	**1.000**	0.667	**1.000**	0.500	0.286	0.667	**1.000**	0.667	0.667
quality_control_2	**1.000**	**1.000**	**1.000**	**1.000**	0.667	**1.000**	**1.000**	**1.000**	**1.000**	0.750	.429	**1.000**	**1.000**	**1.000**	0.750
quality_control_3	**1.000**	**1.000**	**1.000**	**1.000**	0.766	0.571	**1.000**	**1.000**	**1.000**	0.667	T	0.800	**1.000**	**1.000**	0.667
quality_control_4	0.974	0.810	0.873	0.787	0.561	0.658	0.726	0.658	0.780	0.780	T	0.241	0.780	0.608	0.780
quality_control_5	0	**1.000**	**1.000**	**1.000**	0.500	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.500	**1.000**	**1.000**	**1.000**	**1.000**
rail_lines	**0.909**	0.846	0.846	0.966	0.889	0.966	0.966	0.800	0.846	0.537	0.730	0.615	0.889	0.205	0.537
ratner_stock	0.933	0.776	0.824	0.868	0.559	0.396	0.776	0.754	0.824	0.280	T	0.203	0.824	0.378	0.571
robocalls	0.979	0.800	0.966	0.966	0.750	0.862	0.966	0.966	0.966	0.636	0.846	0.714	0.966	0.714	0.636
scanline_126007	0.887	0.710	0.920	**0.921**	0.829	0.906	0.870	0.838	0.889	0.644	T	0.649	0.889	0.818	0.644
scanline_42049	**0.977**	0.485	0.879	0.962	0.889	0.713	0.910	0.908	0.910	0.269	T	0.460	0.910	0.650	0.276
seatbelts	0.659	0.824	**0.838**	0.683	0.583	0.735	0.683	0.621	0.683	0.452	0.383	0.563	0.735	0.583	0.621
shanghai_license	**0.979**	0.966	0.868	0.868	0.605	0.600	0.868	0.465	0.868	0.532	0.389	0.357	0.868	0.385	0.636
uk_coal_employ	F	F	F	F	0.617	F	0.513	0.513	F	**0.639**	F	F	F	F	0.513
unemployment_nl	0.820	0.742	**0.889**	0.876	0.592	0.747	0.755	0.744	0.788	0.566	F/T	0.628	0.788	0.801	0.566
us_population	0.636	**1.000**	0.889	**1.000**	0.615	0.232	0.471	0.276	0.500	0.159	T	0.889	0.889	0.113	0.889
usd_isk	0.914	0.785	0.704	0.785	0.678	0.674	0.785	0.601	0.657	0.489	0.510	0.462	0.678	0.636	0.489
well_log	0.814	0.336	0.914	0.832	0.743	0.822	**0.928**	0.776	0.873	0.149	T	0.923	0.873	0.832	0.237
apple	**0.949**			0.916	0.445		0.745	0.634			F/T				0.594
bee_waggle_6	0.657			**0.929**	0.481		0.233	0.634			0.245				**0.929**
occupancy	0.953			0.919	0.735		**0.932**	0.812			F/T				0.341
run_log	0.994			**1.000**	0.469		0.990	0.909			0.380				0.446
Average F1-measure (1D)	**0.845**	0.739	0.798	0.822	0.596	0.651	0.784	0.657	0.766	0.482	0.354	0.517	0.797	0.517	0.599
Average F1-measure (ALL)	**0.871**	n.a.	n.a.	0.855	0.604	n.a.	0.797	0.683	n.a.	n.a.	0.343	n.a.	n.a.	n.a.	0.61

Table 8.3: Comparison of performance between the proposed method (*Nova*) and other algorithms. The colors indicate if the other algorithms were better (Orange) or worse (Blue) in performance than the *Nova* method for a specific dataset. Time series from *apple*, *bee_waggle_6*, *occupancy* and *run_log* are multivariate. The average results are grouped on the last row. Averages did not considered the gray columns, since these would imply that no change point should be detected, or an error on the signal was present. *T* appears when the method timedout, *M* and *F* when the method failed in compiling

.

As presented in Figure **??**, the critical distance diagram ranks the proposed method in the top four, without a significant difference in performance (considering methods that only work in uni-dimensional datasets). The average F1-measure is 79.6% for both uni and multi-dimensional datasets. Two of the F1-measures are null due to the fact that in these two time series, no change point was supposed to be found, but the proposed method is

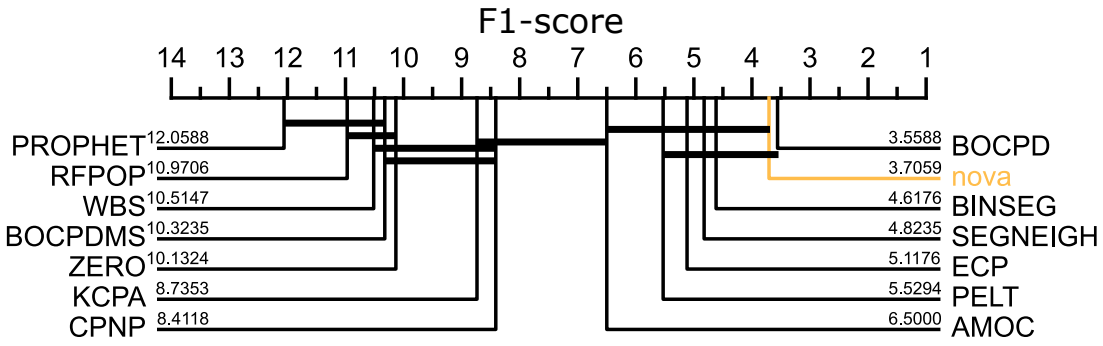


Figure 8.1: Critical distance diagram comparing the methods used in **[cpd_alan]** (except *RBOCPDMS*) and the *novelty function* (*nova* - highlighted in orange). The performance measure corresponds to the F1-score for all single-dimension datasets of the benchmark (except the ones highlighted in gray in Figure **??**).

not yet prepared to return no results.

The results obtained in this benchmark demonstrate that this method is promising, having a performance that competes with several state-of-the art methods in the problem of novelty segmentation. We highlight that the proposed method is applicable to multi-dimensional time series problems, while two of the best ranked methods in Figure **??** are not. In addition, this method is not solely focused in retrieving change points, but also provides additional layers of information, namely periodic changes and the level of similarity between segments, being an advantage over the *BOCPD*.

8.1.3 Discussion of Results

8.1.3.1 Insights on FN and FP

In this section, we present a more comprehensive discussion regarding the *FN* and *FP* counts by means of examples of what could have decreased the performance of the proposed method.

Regarding *FP* counts, these were found in several datasets to be mostly associated with changes on the signal that were not labeled. Since our proposed approach is *unsupervised*, most changes that appear significant will be detected, even if not explicitly labeled. This is visible in Figure **??**, where the subject was climbing stairs but something happened during the acquisition that makes the signal change dramatically. We found the same type of mislabeled data on Datasets 2, 3 and 4, with Dataset 3 having the worst rate of *FP*. This dataset also has a worse rate of *FN*, but mostly because the transitions labeled as sit-to-stand, are often difficult to identify as two different events. It was able to detect most of these changes, but as one event and not as two different events.

For datasets with categorized events, which are a transition between one categorized *subsequence* to another, we can evaluate from which categories the events were missed. Then, identify if specific types of events were the cause of the *FN* count rate. The Figure **??** indicates the F1-score of the *nova* function in detecting the different transition

categories of events using the color and size of circles associated with each event category. The activities before and after change point events are sorted in columns. Each row is associated with a specific event category and corresponding circle, which color indicates the detection's accuracy, while the size indicates the percentage of events associated with the corresponding event category.

As the results illustrate on Figure **??**, the F1-score is lower in cases of activities with similar characteristics on time series. In this case, the transition between $Walking \rightarrow WalkingUpstairs$ has the lowest accuracy rate. However, transitions between $Walking \rightarrow Walkingup/downstairs$, as well as transitions between $Sitting \rightarrow Standing$, are more difficult to detect. These results are supported by Figure **??**, where peaks in the *nova* function are smaller for these transitions. Although a significant portion of these events have been missed, the problem is not necessarily that the ?? has not the checkerboard pattern. In this cases, these more subtle changes were overshadowed by very significant changes, such as $Walking \rightarrow Standing$. Nevertheless, a re-normalization of the ?? segment of interest can help in better visualize the differences that were initially hidden.

An example of the enhancement process is better seen on Figure **??** from Chapter **??**. The block A is highlighted, showing clearly the three modes of the time series. Now that we only focused in that segment, we can clearly perceive these transitions. This example demonstrates that this tool has the potential of enhancing the detection of several transitions in a hierarchical process by re-normalizing segments of an initial segmentation. This would come without a significant extra computational cost, since these measures of the ?? are already computed. Nevertheless, there is

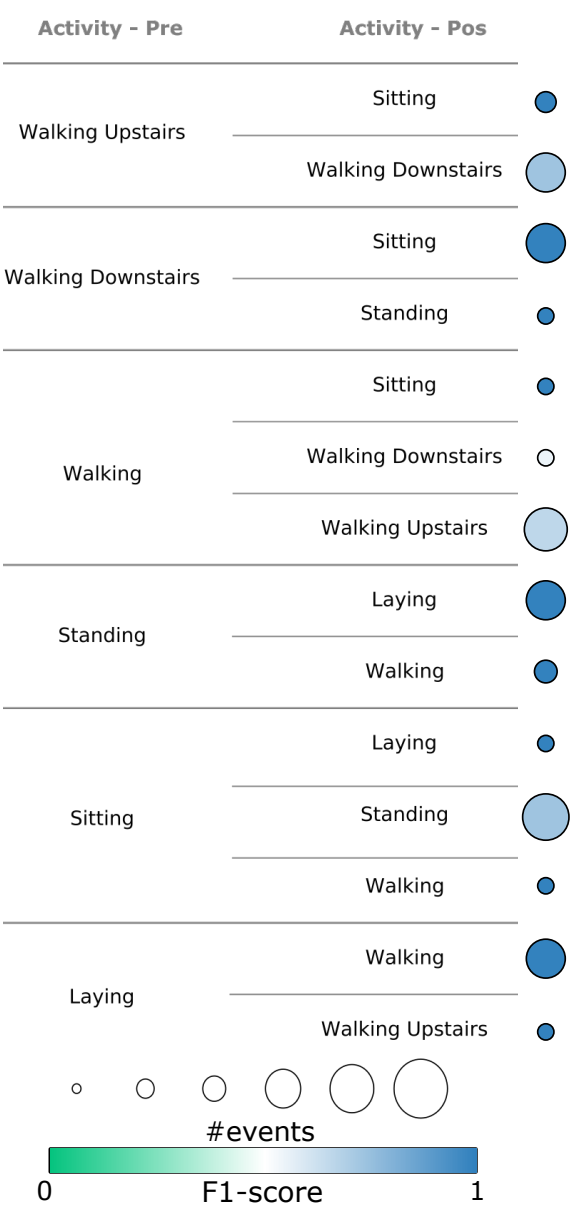


Figure 8.2: F1-score in detecting each of the categories of event transition from Dataset 2. **Pre CP** - activity performed before the change point event; **Pos CP** - activity performed after the change point. The performance in detecting each transition is showed by the circles color.

the potential of finding extra undesired events depending on the threshold level.

8.1.3.2 Comparisons to Related Work

We will now look for specific datasets from the *TCPD* benchmark that should be highlighted considering the results obtained. In Table **??**, we notice that the dataset *bank* has a significant lower accuracy than rival methods. As a default, the benchmark has no event considered for this dataset, which means that any change detected would indicate a worse performance. However, the dataset, has a description in the original paper that states "*Significant changes occur on days of large transactions*". When applying our algorithm to this dataset, we are able to detect most of these changes, as illustrated in Figure **??**. Other datasets, such as *global_co2*, *iceland_tourism*, *lga_passenger* and *us_population*, also had inferior results when compared to the best methods. In the other end, the proposed method was better than the best methods in datasets *bitcoin*, *brent_spot*, *busivn* and *jfk_passenger*. From these results, we notice that the proposed method is more fragile when events occur as a slight change in the trend of a signal, such as the case of *global_co2* or *us_population* records, while it is better for more significant changes in a pattern, amplitude or other property.

The results show that this method is highly competitive in performing event detection in time series, with high performance scores in real-world datasets. From all methods tested in the *TCPD benchmark*, only the *BOCPD* method was able to obtain similar results (Figure **??** and Table **??**), considering also multidimensional datasets, but it does so with an optimization of many parameters. The proposed method is being computed with several parameters as well, but has the potential to accept only one.

One of the other great advantages of the proposed method is that it also provides a rich visual feedback, which can be perfect to support the analyst in interpreting a time series. The content of the ?? goes beyond change point detection, providing relevant information about periodicity and similarity. Not only can we detect change points, as we can also find which segments are more (dis)similar. The other tested methods do not provide such rich information.

8.1.3.3 Effects of Window Size and Overlap

There are several parameters that affect the ability to detect the desired patterns. These are the window size, the overlap percentage and the kernel size. Although in this work we used all three parameters, these can potentially be combined into a single one. We only considered entry parameters on the method that affect the visual output and novelty function. We do not consider the parameters for the simple threshold-based peak strategy used.

The reader may appreciate that these parameters can be explained with the analogy of a camera. The window size works like the *zoom function*, defining the scale of interest in the time series. Larger windows correspond to lower *zoom values* and will compute the similarity of larger *subsequences*, while smaller windows are like a *zoom-in* function

on the time series, searching for any small detail that might be a change. The overlap percentage is the *camera sensor*, which defines the pixel resolution of the image and works as a downsampler of the time series. A full resolution of the ?? is only achieved with total overlap, and the lower the overlap percentage, the less accurate are the highlighted changes.

These two parameters are fundamental for the success of the event detection process. This is mostly evident in some of the datasets where we applied the proposed method, since the larger the time scale used, the larger the MAE in the detected events. For instance, in *Dataset 4*, the time series are very large (3 minutes for each of the 18 activities, at 20 Hz), which due to computation time and memory allocation constraints, we used a larger time window with a lower overlap percentage. The result was an increase in the ??.

In the other hand, the sliding kernel computes the sharpness of the changes detected with the novelty function. The larger it is, the smoother will be the resulting function. Potentially, even if with a small decrease in accuracy, the kernel size could be calculated as a ratio of the window size. The overlap percentage should be maximized and based on the memory available to perform the ?? computation. With this we could define the window size as the sole parameter of this algorithm.

In terms of computation time, the algorithm performs (1) a sliding window to extract features, which is *O(n)* complexity and (2) performs the dot product between matrices, which is traditionally $O(m^2n)$ (recall that m is the number of features and n is the size of the inputted signal). Finally, the correlation of a kernel on the ??'s diagonal has a complexity of $O(nM^2)$ (recall that $M = 2L + 1$, being the size of the sliding kernel).

The memory bandwidth is a drawback in this current stage, since the ?? increases exponentially with the increase in the time series size. We showed that downsampling the time series with a lower overlap percentage is a valid option, while also pursuing a hierarchical search strategy (*recall the example of walking patterns*) can help in searching for the events in multiple steps, reducing the memory limitation. Another potential solution is the computation of only the central diagonal of the ?? with the size of the kernel, which corresponds to the area of interest to detect change points. This procedure would reduce the broad applications that the ?? can provide, namely the periodic segmentation and the similarity measure between *subsequences*.

8.2 Cyclic Segmentation

The detection of cycles was validated on dataset **??**. It consists in 15 multivariate time series with the same repeated cyclic activity. Each time series was segmented based on the ??, which enhances the periodic nature of the time series when present. This method was applied to this dataset, being the results displayed on Table **??**.

The results show the ability of this function to segment the time series into each cycle. It has successfully segmented all time series in all desired segments while having a low duration error (D_e) compared to the ground truth.

Table 8.4: Detected cycles and ?? results of the detection of type 2 events, over the *??* database.

TS sample	Detected Cycles	??(%)
1	19/19	13,25
2	19/19	8,81
3	19/19	8,82
4	19/19	5,82
5	19/19	9,73
6	19/19	6,69
7	19/19	6,18
8	19/19	5,26
9	19/19	6,66
10	19/19	6,97
11	31/31	7,22
12	19/19	8,19
13	19/19	8,96
14	19/19	12,30
15	19/19	6,81

Although this method was successful in segmenting the signal, the proposed solution is not yet optimal. It does not always provide the moment where the *path* starts on the ??. This moment should be the one detected to capture the true starting point of a period.

8.3 Additional Applications

8.3.1 Search by Example

8.3.2 Time Series Summarization with TSSummarize

8.4 Pattern Search with SSTS

This section contains the solutions of the aforementioned examples (*Example 1* is omitted since it was discussed in the previous section). For all presented examples, each step will be explained individually in order to demonstrate its corresponding role, present how the transition from the float domain to the string domain is achieved, and the mechanisms for the search step. Finally, a comparison between ?? and the *classical* solution using the *Halstead* complexity measures will be presented.

8.4.1 Examples' solution

The tool focuses essentially on facilitating the solving of simple tasks, although it can be used for more complex scenarios. We will present three examples: the first and second examples consists of simple problems that require the detection of local maxima and

minima. The third example is a more challenging task, which requires the use of more sophisticated mechanisms of the regular expression module.

8.4.1.1 Example 2 - Step Detection in accelerometer signals

The example consists of detecting the subject's steps in the acceleration signal. For this particular case, only the detection of the right heel contact will be discussed, whereas the left heel contact example's solution is presented in Table **??**.

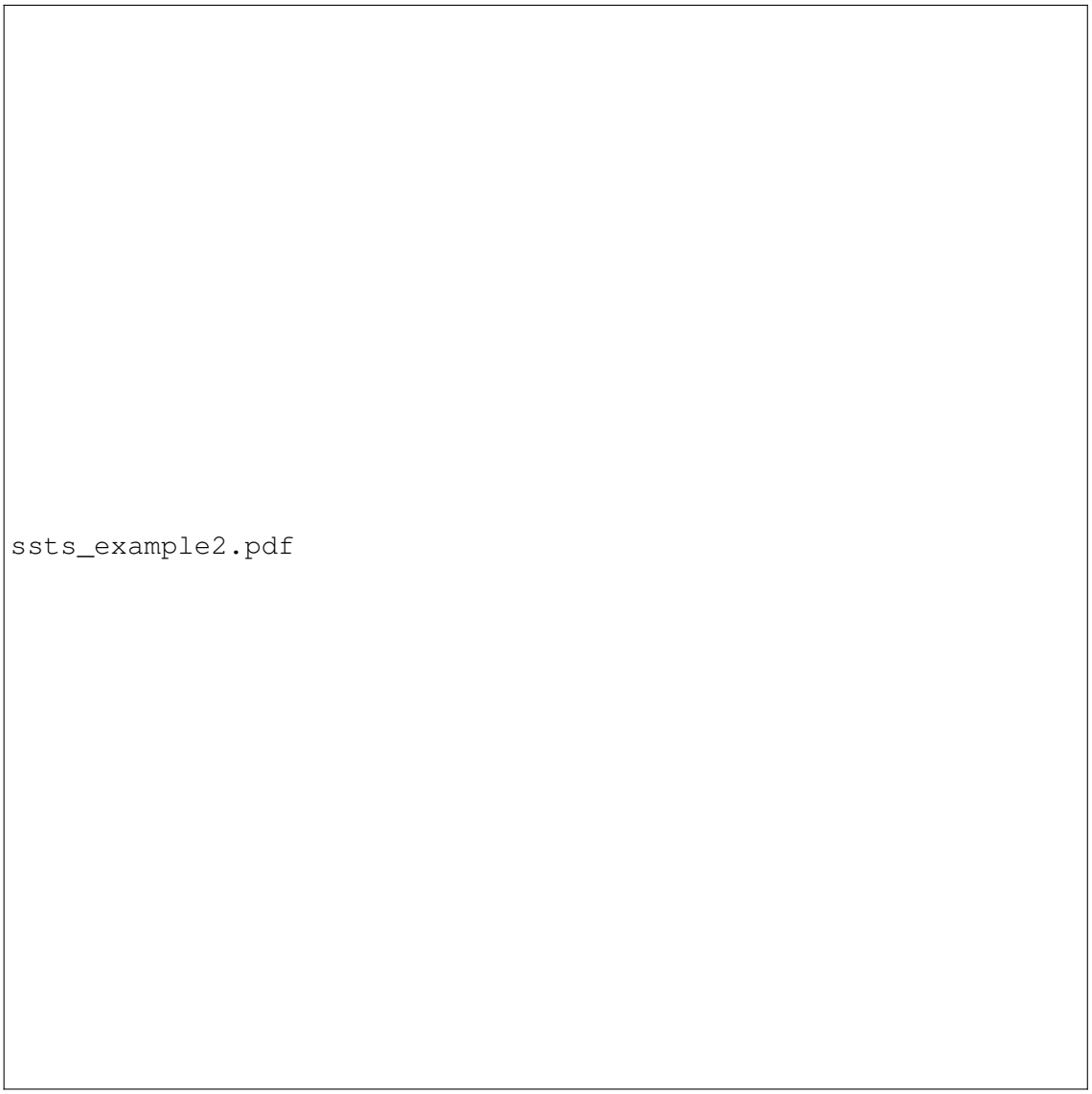


Figure 8.3: Example 2 - Solution pipeline of Step Detection example. At the bottom of the figure is summarized the operators and methods used in each step. In the symbolic connotation step, the alternation between the methods is made with colours (Blue - ampC - `, Red - diffC - `). A doted line is shown to represent the threshold level. To each colorband in the signal corresponds a specific primitive. A positive match is highlighted with green in the search step.

The signal is initially pre-processed in order to ease the identification of the subject's

steps. This is achieved using a low pass filter (` or LP). The result is presented in the second image of the first step in Fig. **??**. An highlight is also present and delineates a segment of the signal that has a minimum peak, which corresponds to a right heel contact. The string representation of this segment is depicted in the second step.

The string is a sequence of primitives composed by two connotation methods (ampC - ` in blue, diffC - ` in red). Based on these methods, the samples of the signal with amplitudes greater than *40*% of the range amplitude are transcribed into 1, while the remaining turn into 0; the slopes are converted into p, z and n, when rising, being stationary and falling, respectively.

The solution involves detecting each minimum with an amplitude inferior to the threshold level. The morphological representation of a minimum can be reduced to a negative slope followed by a positive one, which in the symbolic representation is defined by the second connotation method as the transition from n to p or z to p. Regarding the amplitude requirement, it is assigned by the first connotation method, in which any value lower to the threshold level is 0. The regular expression used to find this minimum was $0z0p$, which implies that: (1) the amplitude has to be 0; and (2) the derivative is z and then p. The detection is highlighted in green in the last plot of Fig. **??**.

8.4.1.2 Example 3 - Dicrotic notch detection in ?? signals

The ?? waves are morphologically represented by a high positive slope that corresponds to the systolic uptake and ends in the peak of the systolic pressure. After this behaviour, follows the systolic decline, which ends with the aortic valve closure, named the *dicrotic notch* in the signal representation [**abpSignal**].

These type of signals are commonly affected by low frequency noise that modulates the signal and contaminated with high frequency noise of low amplitude. The typical procedure is to bandpass the ?? signal in order to remove both types of noise. Fig. **??** depicts how a band pass filter that cuts frequencies under 1 Hz and above 20 Hz (` or BP) is applied to the signal to remove both types of noise. The modulation removal is shown with the linear regression in both original and pre-processed signals, which in the first case has a positive slope and after the pre-processing is approximately zero.

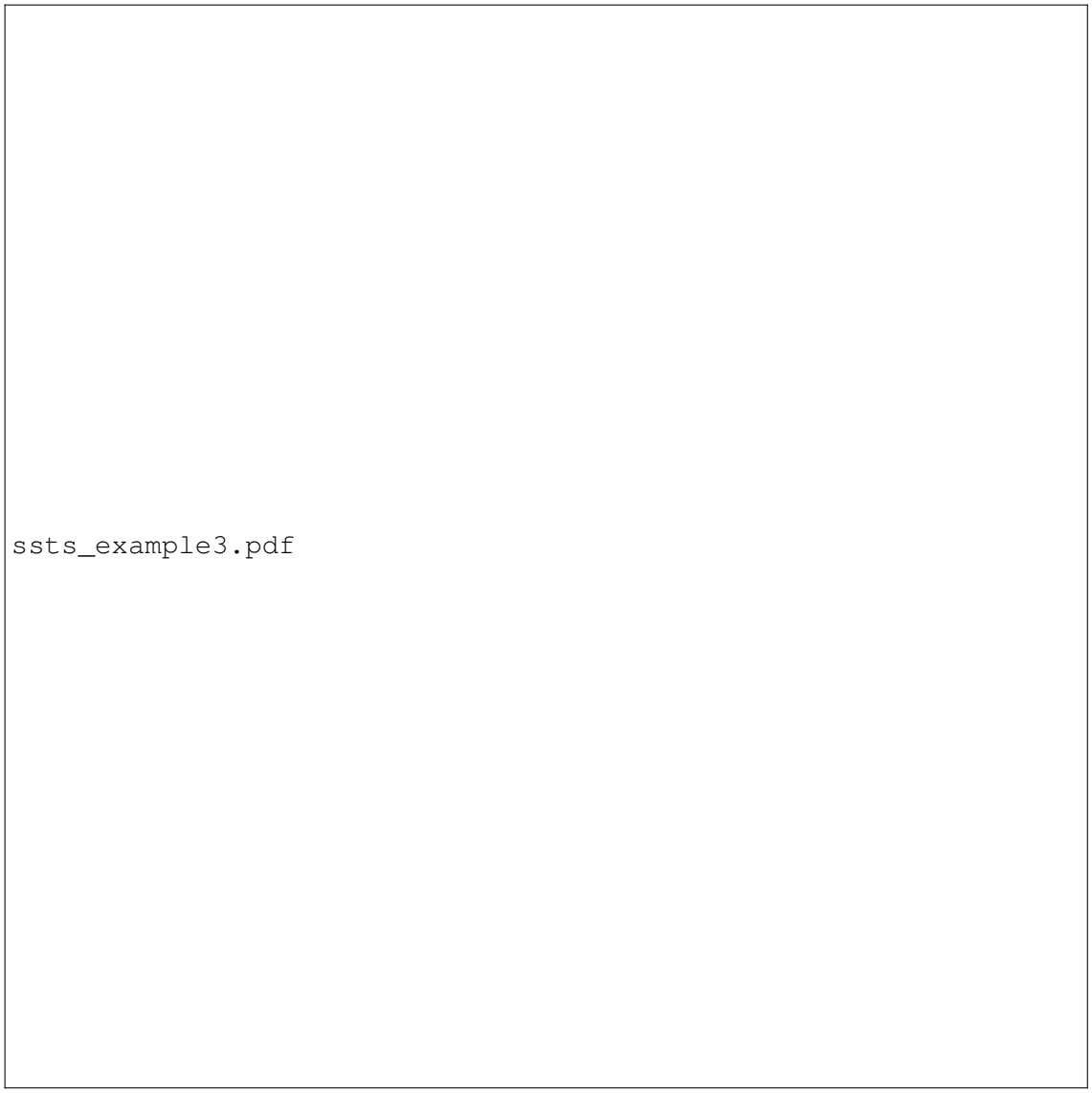


Figure 8.4: Example 2 - Solution pipeline of Dicrotic notch detection example. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between methods is represented with colors (Blue - RiseAmp - `, Red - diffC - `). For each colorband in the signal there is a corresponding primitive. The match is highlighted with green in the search step.

In the symbolic connotation step, the reasoning follows the signal morphological description and uses two methods for the symbolic representation. The first, ` (represented in blue), detects all rising and falling slopes that are higher than a specific threshold (in this case 30% of the amplitude range of the signal) and transcribes the sample values to 1, while the remaining values are converted to 0. The second method (represented in red) uses the derivative, as mentioned in the previous examples.

The first connotation method is necessary in order to distinguish between the rising slope that occurs in the beginning of the pressure wave and the one after the *dicrotic notch*. With this distinction, it is possible to find the beginning of the ?? wave as a high positive

slope ($1p$) and find the *dicrotic notch* when the lower slope starts (0.). In order to find this area of the ?? wave, the regular expression has to start with the first $1p$ primitive and end with the first 0. primitive.

The example is solved with the following regular expression: $(1p).*?(0.)$. This string means that the search will match anything (represented by ".∗?") between the first "$1p$" primitive and the first "*0.*" primitive.

8.4.1.3 Example 6 - Stable lifting detection in accelerometer signals

In the previous example, the solution was achieved by searching for one simple transition in the string generated by the sequence of connotation methods. This tool may also be used to solve more complex examples in the same manner.

The next problem involves the segmentation of a lifting step that has occurred 5 seconds after the start of a weight lifting exercise. The example can be solved in two steps: (1) find the start of the exercise, (2) search for the segment 5s after the start. This rationale can be expressed by combining two distinct symbolic representations of the same original signal, which requires the use of two pre-processing and symbolic connotation sets, in which one is used for the detection of the start and the other to find the desired segment.

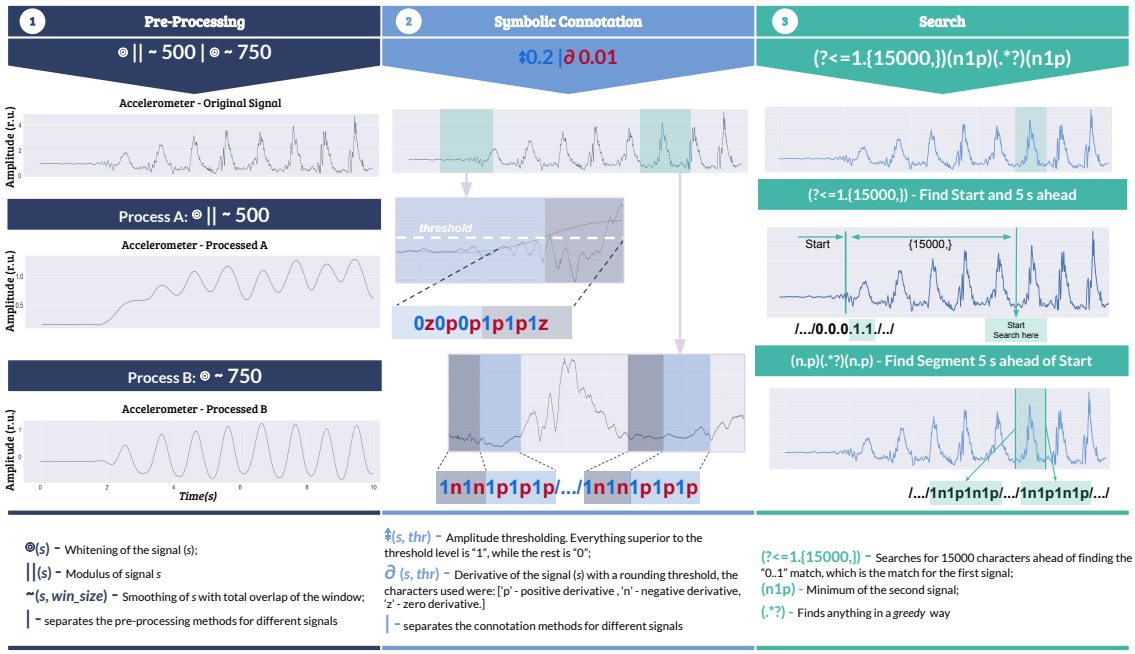


Figure 8.5: Example 6 - Solution pipeline of Stable Lifting Detection. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - RiseAmp - `, Red - diffC - `). The threshold level is identified with a white dotted line. To each colorband in the signal corresponds a specific primitive. The match is highlighted with green in the "Search" step.

Fig. **??** demonstrates how the example is solved. In both pre-processing and symbolic

connotation steps, a vertical bar | separates the methods that are applied for each representation of the same signal. The pre-processing phase uses a sequence of whitening, modulus and smoothing of the signal for "*Process A*", and a sequence of whitening and smoothing for "*Process B*". The first processing sequence turns the signal similar to a plateau, in which the beginning of the activity is easily identified; whereas, in the second sequence, the signal is smoothed such that each lifting step is well defined.

Regarding the symbolic connotation step, the first method ` (represented in blue) turns all sample values of the first signal that are higher to the threshold level into 1, while the remaining samples are converted to 0. The second set is applied to the second signal and uses the derivative of the signal ` (represented in red). Both symbolic connotations merge into a sequence of primitives, in which the first element inspects if the exercise has already started (1.) or not (0.), and the second infers the sectioning of the lifting steps, that is, if the sample of the signal is increasing (.p), stationary (.z) or decreasing (.n).

The regular expression written to solve the example is (? <= 1.{15000, })($n1p$)(.*?)($n1p$). Decomposing this expression in the two steps of the example results in: (1) (? <= 1.15000,) and (2)($n1p$)(.*?)($n1p$). In (1), a *lookbehind* operator is used, therefore, the compiler will search ahead of the first match inside the operator, i.e., the search will match the expression "1.{15000, }" and search ahead of it. This method aims to handle the temporal dependence between events in the string, in which the number 15000 is calculated by the number of characters that correspond to 5 seconds in the string. This calculation was achieved by multiplying the sampling frequency, the number of connotations in each primitive, and the desired time, in this case: 1000 Hz, 3 connotations and 5s, respectively.

8.4.2 Further Applications of SSTS

- Classification - Compressing Tool - Edition Tool - Alignment between Time Series with Longest Common Distance

8.5 Time Series Classification with HeaRTS

In this section is evaluated the ability to perform time series classification with the proposed ?? method. For this, it has been applied to the *UCR time series classification benchmark* and compared with the traditional 1-nearest neighbor based on the euclidean distance. As was mentioned on Chapter **??**, this process can be performed in multiple ways, namely with the combination of a *Baysian* or ?? classifiers with either a ?? or a ?? model. These combinations were also compared. With these evaluations, we expect to conclude (1) if this approach of using a linguistic transformation of a time series with high level words ordered in time of occurrences is possible with significant performances and (2) which is the combination that best fits this approach.

In addition to these experiments, we performed an individual analysis of several use-cases to understand which is the interpretability and readability of the data based on

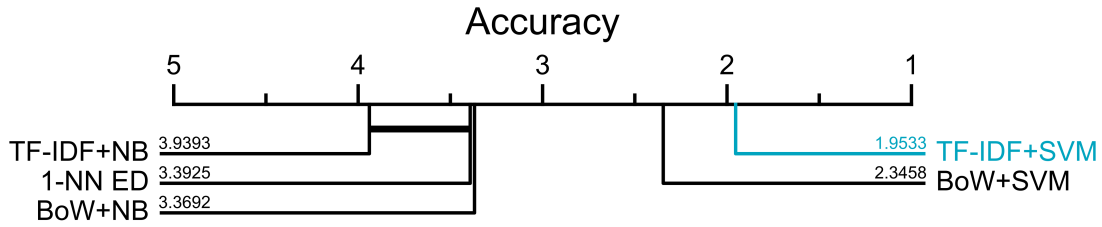


Figure 8.6: Critical distance diagram for the methods that rely in a textual representation of the time series (BoW and TF-IDF) and euclidean distance.

this linguistic translation. This study should tell us if the words extracted are valuable to indicate the *subsequences* of the time series that are relevant to identify it and distinguish it from others.

8.5.1 Classification Performance

Figure **??** shows us how vectorized documents can be compared with the cosine distance to distinguish time series. Simply by using part of the ?? queries, it is possible to generate ?? weights that are differently distributed based on the words and how these are ordered on a time series document. Of course, the *Trace* dataset used for this example is a simple dataset, with simple dynamics, but still challenging to classify well with the 1-NN ED (only 76% of accuracy). Most of the datasets on the UCR benchmark are real data and more complex. In that sense, the results presented for all the datasets are made with all the ?? queries presented on Table **??**.

As mentioned on Chapter **??**, the method translates the time series into text with ?? queries from Table **??**. These have a pre-processing, connotation and search steps. For each step, relevant parameters are necessary depending on the queries used. For this experiment, we used as pre-processing a simple smoothing, which requires a *window size* (w_s). The derivative connotation steps also required a specific threshold (thr). In addition, the ?? and ?? models can be built with different *n-gram* sizes. These three parameters were optimize with a grid search method over a range of values:

$$\begin{aligned} w_s &\in [1, 10, 25, 50, 100, 250] \\ thr &\in [0.001, 0.01, 0.05, 0.1] \\ n-gram &\in [1, 2, 3, 4, 5] \end{aligned} \tag{8.1}$$

We performed the classification with several strategies and combinations, namely: ?? with Baysian Classifier (BoW+NB), ?? with ?? (BoW+SVM), ?? with Baysian Classifier (TF-IDF+NB) and ?? with ??. The summary of the performance is presented on Figure **??**.

As can be seen, the strategies that rely on a ?? classifier are more robust. The ?? turns out to not be reliable when used with the Baysian Classifier. In the other end, it works well with ?? Classifiers, being the best method studied. We then compared it in more detail with the 1-NN ED classifier, as presented on Figure **??**

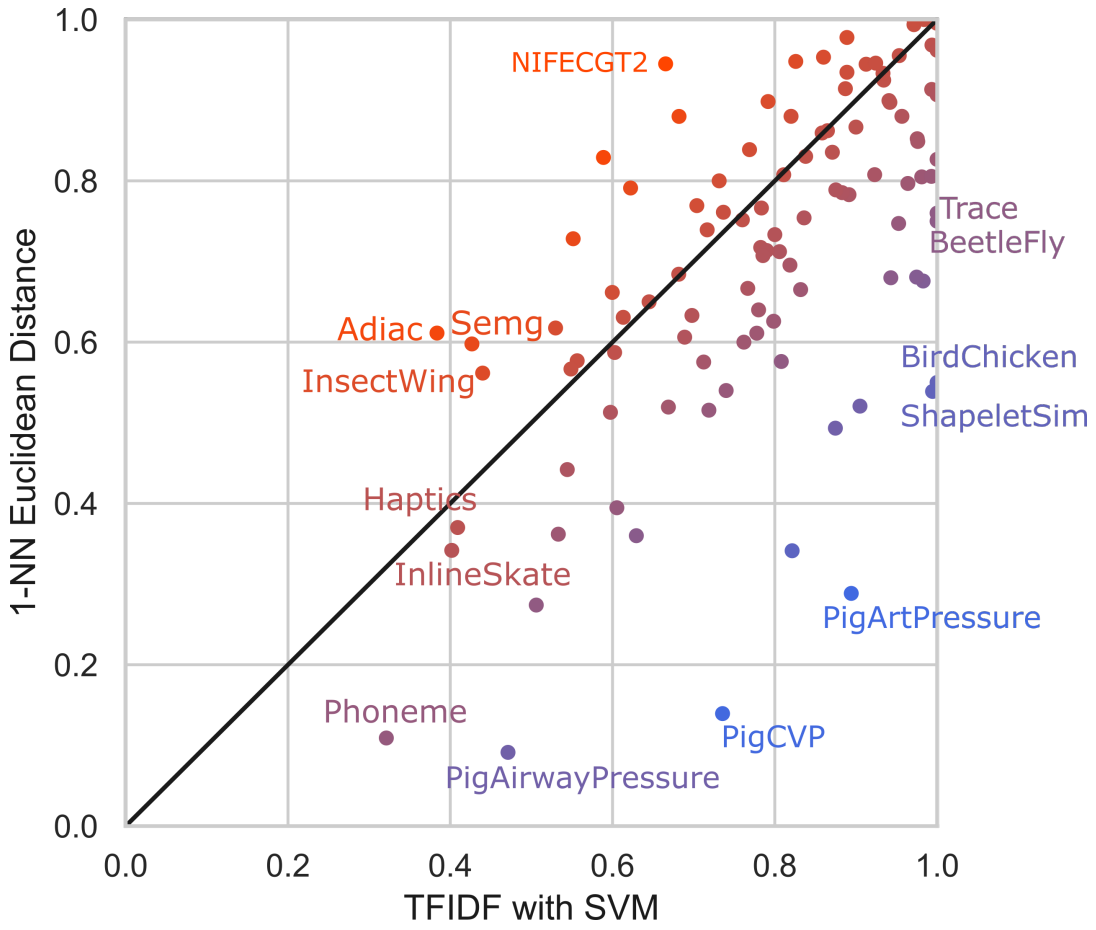


Figure 8.7: Classification accuracies for the Bag of Synthatic Patterns in comparison with the euclidean distance combined with 1-NN classifier (1-NN ED).

The average accuracy for the total 107 datasets is 79% for ?? with 72 wins, and 71% for the 1-NN ED, with 35 wins. Special cases are highlighted on Figure **??**. These indicate that the proposed method works poorly in problems with a high number of classes (*Phoneme-39*, *Adiac-37*, *PigAirwayPressure-52*, etc...), although some exceptions are found, such as in *PigArtPressure*. Both *Haptics* and *InlineSkate* are badly classified in general by state of the art methods. Binary problems that rely in shapes well described with the derivative properties, such as *ShapeletSim*, *BirdChicken*, *BeetleFly* and *Trace*, are typically better classified. With these results, we can clearly see the surprising ability of this strategy in classification tasks for most of the datasets tested against a standard 1-NN ED.

We wanted to answer two main research questions: (1) can we solve time series classification tasks based on its textual representation following traditional *NLP* methods and (2) could we use the words that describe time series as a readable explanation of the data and the differences between classes. Having evidence that the first point is achieved, we will look into the second point.

8.5.2 Interpretable Data Outputs

Besides the fact that this approach is capable of performing well in classification tasks, we expected to go further by providing some feedback from the textual description, helping to understand what differentiates the classes. The ?? model gives weights for each *word* that characterizes the time series. Searching back the *words* on the time series and summing the weights corresponding to these *subsequences*, a shape relevance is made on the time series, where areas of higher interest and that differentiate the time series from others are highlighted. The next Figures (**??**, **??** and **??**) are displayed showing the highlighted areas based on the ?? weights.

In addition to using the weights of the ?? model, it is also possible to extract relevant keywords from the textual representation. This task is called "Keyword Extraction" on the text mining domain. We believe that we can profit from this knowledge to make a more interpretable understanding what characterizes a time series. This extraction was performed with the *Text Ranking* algorithm on the documents generated for each time series.

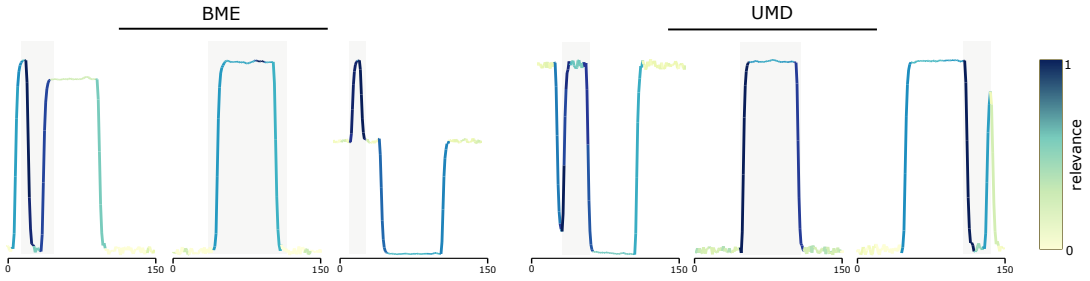


Figure 8.8: Interpretable results with highlighted shapes for the *UMD* and *BME* datasets. The *UMD* dataset was classified with with an accuracy of 99.3%, computed with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$. The *BME* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$

On Figure **??** are showed two similar datasets. The f1-score of this task was 1, meaning that it was able to perfectly characterize each shape with the textual representation. For instance, the *BME* dataset has two different sequences of shapes, since the first class has a *peak followed by a positive plateau*, while the third class has a *peak followed by a negative plateau*. The second class has simply a *plateau*. The main difference between all those classes are the peaks and what follows it, which is precisely what is highlighted. The same happens on the *UMD* dataset, in which the most relevant element is the peak and the plateau. As we can see, the transition from the peak to the plateau is highlighted.

Another dataset that is intuitive to understand is the *Trace* dataset. We have been using it frequently in this work and can very easily understand what distinguishes one class from the others. The *subsequences* of interest are clearly highlighted on each time series of each class. For instance, the first class has the peak and valley highlighted, contrasting with the valley from class 2. Class 3 and 4 have a notoriously different valued elements. While class 3 has the rising phase as the relevant shape, class 4 has a small peak followed

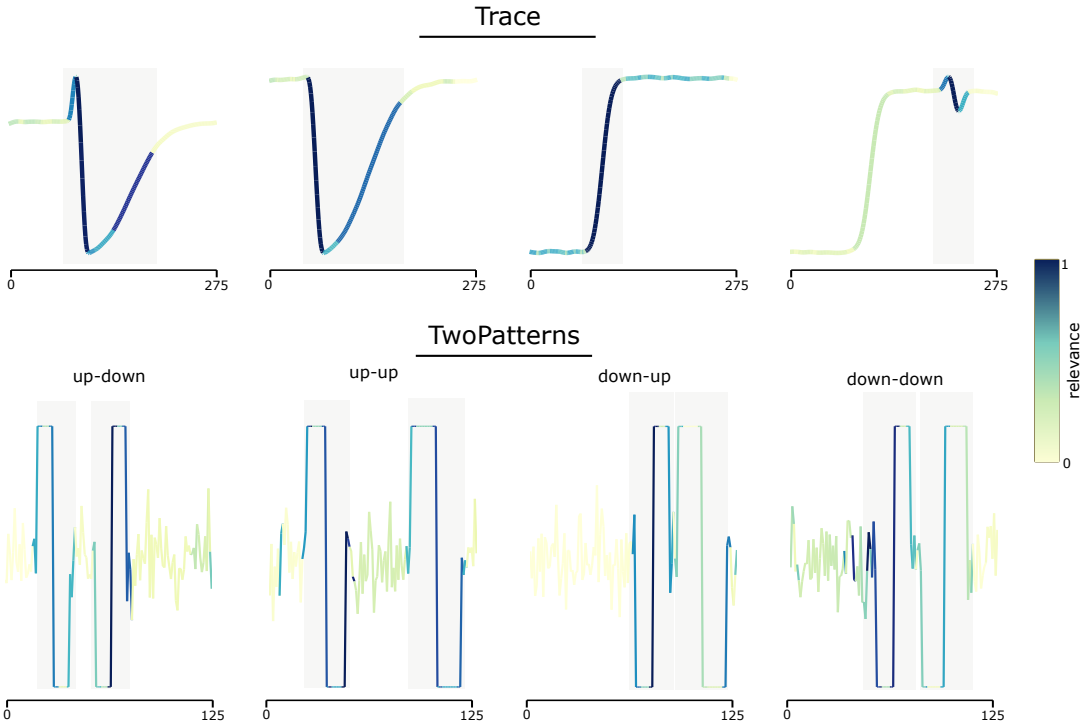


Figure 8.9: Interpretable results with highlighted shapes for the *Trace* and *TwoPatterns* datasets. The *Trace* dataset was classified with with an accuracy of 100.0%, computed with a $w_s = 25$, $thr = 0.05$ and a $1 - gram$. The *TwoPatterns* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.1$ and a $5 - gram$

by a valley on top highlighted.

The *TwoPatterns* dataset is also intuitive to understand and fits the problem for this experiment. Each class has mostly noise, filled with step functions that have different sequences. Class 1 starts by having an *up* step function and then a *down* step function. The same logic is applied to the other classes. The proposed method highlights these step functions as being the relevant segments of the signal.

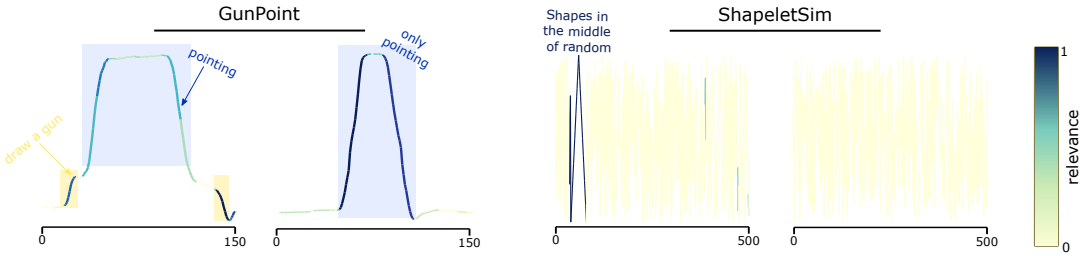


Figure 8.10: Interpretable results with highlighted shapes for the *Gunpoint* and *ShapeletSim* datasets. The GunPoint dataset was classified with with an accuracy of 99.0%, computed with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$. The *ShapeletSim* was classified with an accuracy of 99.4% with a $w_s = 1$, $thr = 0.05$ and a $1 - gram$

Finally, two other examples are also presented. One of the *GunPoint* dataset and another from the *ShapeletSim* dataset. The *GunPoint* has 2 classes: (1) the subjects draws a plastic gun and points it; (2) the subject simply points with his/her hand. The major

difference between the two classes is the drawing process, highlighted by the method as the *small rise* and *small fall*. Some of the top-10 keywords are *small fall* before a *high rise* and *small rise*.

Regarding the *ShapeletSim* dataset, the data has also two classes. In one, simple shapes, such as triangular, square or sawtooth waves were added to a random signal. The method is able to highlight the representative shape on the signal. Some of the most relevant keywords to distinguish it from the other class are: *low Peak*, *high rise* and *high fall*.

8.5.3 Discussion of Results

In this section, we wanted to answer two main research questions: (1) can we solve time series classification tasks based on its textual representation following traditional *NLP* methods and (2) could we use the words that describe the time series as a readable explanation of the data and the differences between classes. For this purpose, we applied the proposed method on the *UCR* classification benchmark and highlighted several relevant use-cases and the overall results achieved.

The results provide evidence that it is possible to compare time series based on a textual representation and using traditional strategies from the text mining domain. Not only is it possible to compare the vectorized of time series document with the cosine distance (as we have seen on Figure **??**, but ?? or ?? models can be used with a linear ?? to solve classification tasks.

Additionally, the fact that words are weighted based on their relevance to distinguish each time series document from all the others, it is possible to highlight the *subsequences* of the time series corresponding to these words and understand which are most relevant for the distinction of class, being a step forward into interpretability of time series. The limited examples were still simple and intuitive and the process turns harder to understand in more complex data. This might mean that the method has a diverse enough set of words to perform a distinction between classes, but the words used are not capable of expressing this difference. We can highlight many cases with high accuracy scores, but their interpretability falls short. A few examples can be showed. For instance, on Figure **??** is showed a binary classification problem in which the method has 100% accuracy but does not show expressively the difference between classes. Of course, it might be difficult to express this difference and maybe words are not sufficient.

The textual representation is valuable in the sense that text mining domain knowledge can be used on time series. Nevertheless, we can profit from this knowledge as much as the translation of the time series into text is rich, valuable and domain specific. With the current set of ?? queries we are able to make a limited description that works well in simple signals described mostly by their derivative dynamics. The keywords extracted for these time series make sense and we can relate to what we see. However, if more expressive queries are used to make a more robust translation, the keywords extracted can be more valuable and relatable with the time series we are looking at. Therefore, a

more diverse set of queries should be used to make possible the extraction of relevant keywords.

We also wanted to mention that the translation of time series into text brings novel ways of measuring distances. It also shows a totally different textual feature domain that can help complement the information extracted by classic features from the statistical, time and frequency domains.

8.6 Pattern Search with QuoTS

8.6.1 *QuoTS* Matches Shapes

We start by demonstrating the ability of ?? to sort how well individual shapes match a written query. For this, we used the *UWaveGestureLibrary* dataset from the *UCRArchive* as a proxy **[uWave]**, which similar to telemetry data, relies on inertial time series. We use this proxy data simply because it is much larger than any publicly available labeled transportation data. However, we note that gestural interaction with the automobile interfaces is an area of active research **[autoui1**, **autoui2]**. With this example, we show that the system can recognize different gestures with or without intuitive queries, hence, if humans were able to learn and master this tool, this recognition problem would be largely solved.

We not only demonstrate that the system can significantly distinguish gesture patterns, but it does so with very simple and intuitive queries. The ability of the written queries to match the correct class is presented both visually (Figure **??** and quantitatively (Table **??**). In Figure 5, the set of eight gestures is described row-wise, having a corresponding mean shape (**MeanWave**) and a query (**Query**) that should filter it. The visual intuition is demonstrated with the barplot (**MatchScore**), which for the sake of readability, highlights the normalized match score ([0-1]) of subsequences belonging to the row-wise specific gesture. The other classes are shown in gray. For each gesture, we show the shape for all the available axis (X, Y and Z). We attempt to create queries using only a single axis (X-axis) but used other dimensions when needed.

To quantify these results, we looked at the top-10 and top-100 matches for each class and counted how many gestures of the selected class were correctly sorted (TP/10 and TP/100). The results are presented in Table **??**. These show that the top 10 matches always correspond to the correct class. The reader might think that the first 10 matches might be too easy considering a problem that has around 400 gesture samples per class but note that having more gesture samples also means more opportunities to make mistakes. Moreover, considering the analogy of searching for a webpage with the *Google* search engine, a user will probably be interest in examining at least the top 10 results. Nevertheless, the reader will appreciate that even if we consider the top 100 matches, ?? still achieved impressive results.

Although we simply wanted to demonstrate that with a set of meaningful words we can

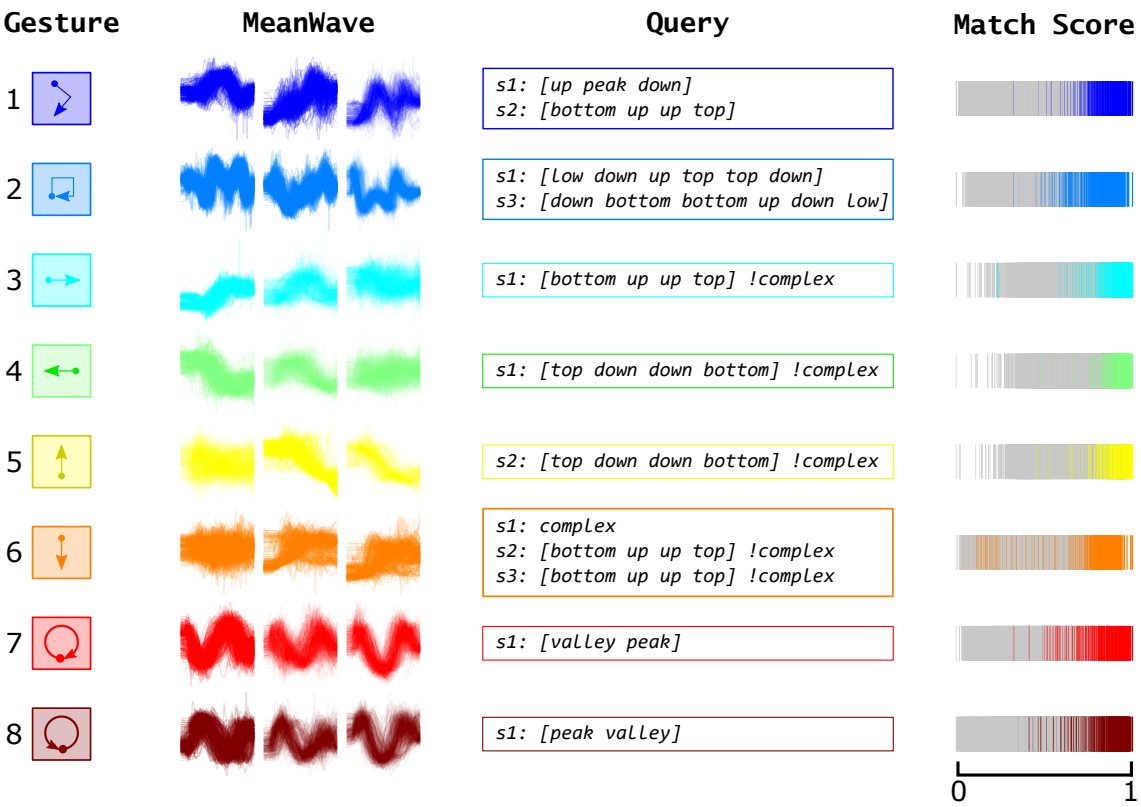


Figure 8.11: *UWaveGestureLibrary* subsequence matches with ??. Column-wise are showed the gesture class, the corresponding mean wave for all subsequences, the query used to match the subsequence class and the corresponding match score for all subsequences, highlighting with the color of the specific class.

	G1	G2	G3	G4	G5	G6	G7	G8
TP/10	10	10	10	10	10	10	10	10
TP/100	96	100	94	95	74	100	100	99

Table 8.5: Results for the top 10 and top 100 sorted gestures classes (G) when using the queries from Figure **??**.

correctly sort each of the classes of this dataset, we also want to highlight that the nature of the classes can be very well expressed by the queries. This is especially evident when we look at the query for gestures that are inverse to each other, such as gestures 7 and 8. Gesture 7 is well matched by the query `[valley peak]`, implicitly, the transposed gesture should have the exact opposite query, which it indeed does (`[peak valley]`). This also occurs for the two other sets of gestures that occur in natural pair; gestures 3 (`[bottom up up top] simple`) - 4 (`[top down down bottom] simple`) and gestures 5 (`[top down down bottom]`) - 6 (`[bottom up up top]`). We also demonstrate that for the handful cases where this did not work, there was a semantic explanation for it. (e.g. Classes 5 and 6 have not a specific pattern and seem to be specially random in their X-axis or, classes 1 and 2 are very similar, and because of that, are mislabeled). But, as our tool can perform

queries in multidimensional data, we can still discriminate these by using the Y-axis in conjunction with X-axis to sort them correctly. Note that discriminating among these eight classes is not a trivial problem. Of the more than 1,000 papers to have worked with this dataset, the current best accuracy was obtained by *COTE* algorithm which achieved 76.56% using a single axis. In addition, this dataset was acquired from eight different subjects, which indicates that our system can account for the intra-subject and inter-subject variability in motion for this dataset [**uWave**].

8.6.2 Matching *Known* Shapes with Words

We believe that ?? is intuitive enough to allow most novice users to create simple effective queries for most information retrieval tasks. However, in some cases, the user's ability to formulate queries may be a limitation. In this section we show that, perhaps uniquely, this domain allows us to generate queries by "puppeteering" a model car. To demonstrate this, we attached a smartphone to a model car (as indicated in Figure **??**) and acquired inertial data while mimicking a 3-point turn event on a table. The resulting shape is illustrated in *puppet data*. We believe this can be used in two ways: for the user to gain intuition as to how a motion/*manoeuver* is illustrated on motion data or the how the inertial data from the motion/manoeuver can be used as a MASS template on ??: • Shape Intuition: A user might not know what the shape of a 3-point turn looks like. By using a model car, he/she can mimic the motion and gain intuition over what the query should be (in this case, `[peak peak peak]`). • MASS template: As mentioned above, we have a special shape word, which corresponds to a shape template given by the user, to which a word can be assigned. We then can use the word in our language to match desired patterns with the MASS distance profile as a word feature vector.

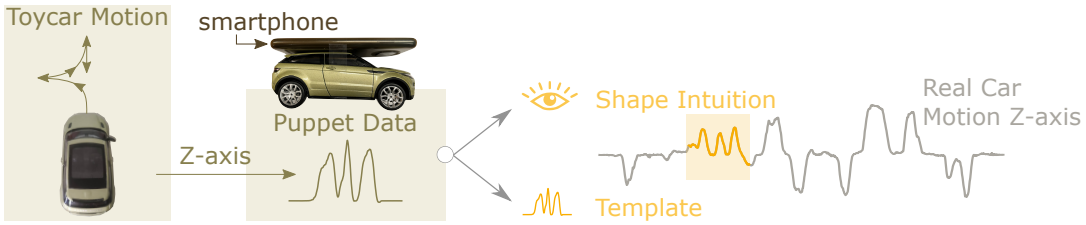


Figure 8.12: Creating query prompt data by puppeteering a car model. The puppet data for a 3-point turn (smoothed with 10 samples moving average) is presented on the left, while the real data from a real car is presented on the right.

We added the puppet data and the word `3pointturn` to our vocabulary. Searching for it in our dataset matched all the desired shapes. Note that the template has a different amplitude and time scale, but for *MASS*, the z-normalization makes the amplitude difference irrelevant, and the time scale was adjusted by resampling the template based on the selected window size for the query search. With this, the template is not only a template, it also becomes part of the language, being used in combination with all the other words available in our vocabulary.

8.7 Further Application 1: Search by Example

8.8 Further Application 2: Multidimensional Segmentation

The proposed method accepts both single and multidimensional records. The difference regards the number of features extracted. As presented on Figure **??**, the same set of features are extracted for each time series of the record and combined in the F_M.

Using a single time series of a multivariate record is optional and depends on the detection's purpose. In some cases, using a single time series from a multidimensional record can lead to missing relevant events undetected. An example of this can be seen on Figure **??** with record *"Occupancy"* from Dataset 8.

The record is a multi-dimensional time series that measures room occupancy based on temperature, humidity, light and CO2. All events can only be detected if using several time series of the record [**cpd_alan**]. On Figure **??**.left, a single time series was analyzed by the proposed method to detect relevant events, while Figure **??**.right presents the application of the method to all the time series of the record. The results are different because the information from all the time series is combined, while with the single dimensional record, the detected events resulted from the information available on the single time series.

8.9 Application to Occupational Scenario

8.9.1 Storytelling Occupational Data

1 - Example of the data we acquired in Volkswagen when we arrived and tried to compare two workstations. We were trying to find specific cyclic moments and labelled it by hand. We can use this tool to make this identification (SSM):

1.1 - show matrix of the data

2 - Describe the pattern by means of words or a regular expression

8.9.2 Pattern Search in Occupational Data

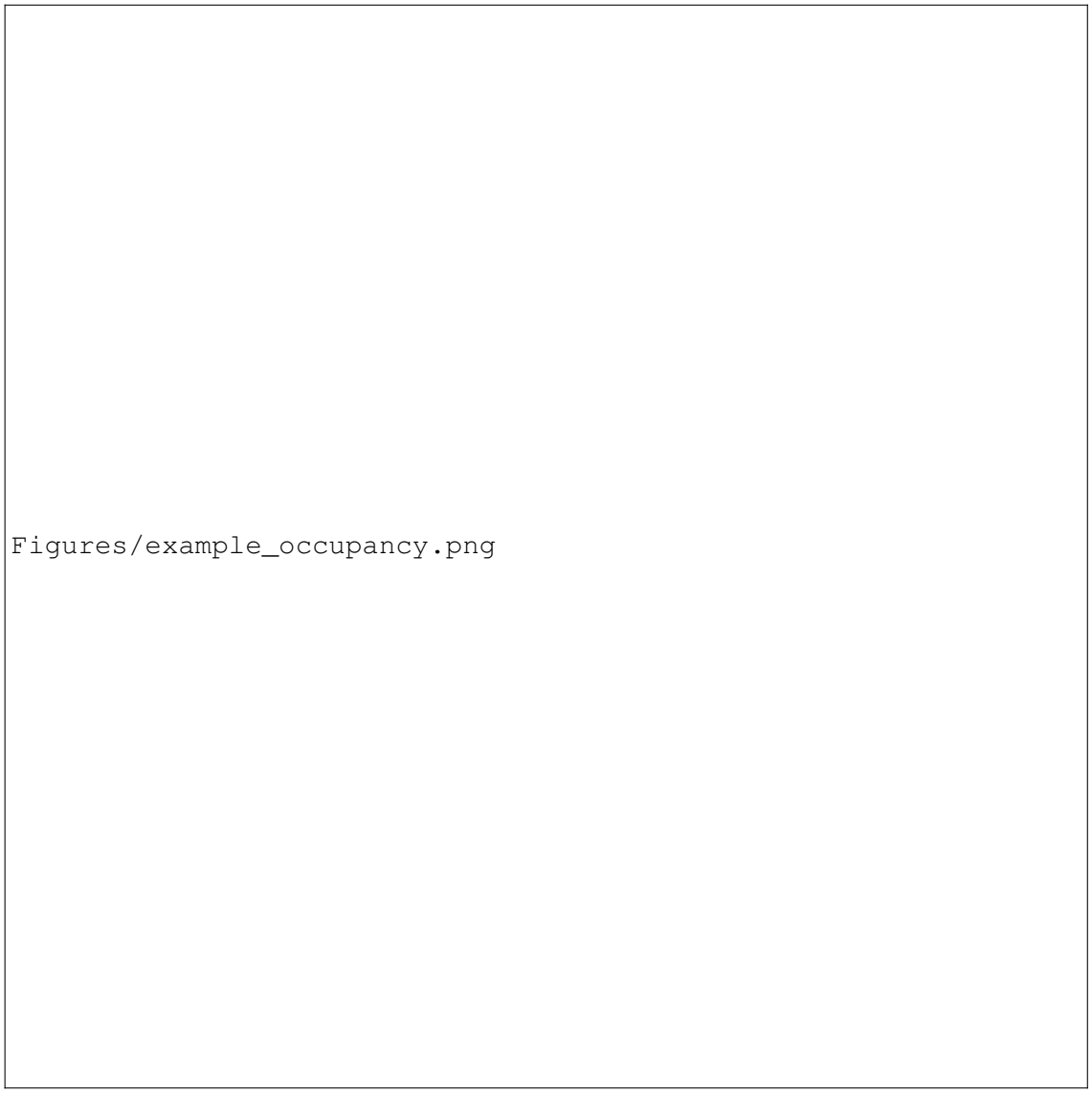


Figure 8.13: Proposed method applied on "*Occupancy*" record of Dataset 7. A single time series of the record is used to extract events.

9

Conclusion

Future Work

- Associate the Bag of Words model weights with the shapes extracted. Save the subsequences and the words. Transform it in the bag of subsequences, and you can create a distance between the words. For instance, a Peak in Signal 1 might be different than a Peak in Signal 2. Or the signal might have the same dynamic (up down flat) but the up and down have a much different shape. By comparing the shapes, we might have an additional distance measure that makes the process more relevant...keep this in mind.

- Applying the cost matrix to the SSM to see what comes out of it. Would it help get the cyclic patterns in one time?

- Explore more NLP methods that could be used on the symbolic/textual representation of time series

- BERT for Time Series

- Profile with a summarization and overall report of time series that includes a more detailed description of the content of the data, more statistical information

-

10

Future Work

A

NOVAthesis covers showcase

This Appendix shows examples of covers for some of the supported Schools. When the Schools have very similar covers (e.g., all the schools from Universidade do Minho), just one cover is shown. If the covers for MSc dissertations and PhD thesis are considerable different (e.g., for FCT-NOVA and UMinho), then both are shown.

B

Appendix 2 Lorem Ipsum

This is a test with citing something [**ecoop12-dias**] in the appendix.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea

dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

I

Annex 1 Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum

wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.