NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF PHYSICS

THE LANGUAGE OF BIOSIGNALS

THE LINGUISTIC NATURE OF TIME SERIES

JOÃO MANUEL DE ALMEIDA RODRIGUES

Doctoral/Ph.D dissertation

DOCTORATE IN NOVA INNOVATION FOR HEALTH (NI4H)

NOVA University Lisbon
September, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF PHYSICS

THE LANGUAGE OF BIOSIGNALS

THE LINGUISTIC NATURE OF TIME SERIES

JOÃO MANUEL DE ALMEIDA RODRIGUES

Doctoral/Ph.D dissertation

Adviser: Hugo Filipe Silveira Gamboa
Associate Professor, FCT NOVA University Lisbon

Co-adviser: Carlos Fujão
Ergonomist, Nova Volkswagen Autoeuropa

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 NOVA INNOVATION FOR HEALTH (NI4H)
SPECIALIZATION IN BIOMEDICAL ENGINEERING

NOVA University Lisbon
September, 2022

The Language of Biosignals

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.14) [94].

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

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.

This has probably happened because there are no entries defined in this glossary. Did you forget to use `type=symbols` when you defined your entries? If you tried to load entries into this glossary with `\loadglsentries` did you remember to use `[symbols]` as the optional argument? If you did, check that the definitions in the file you loaded all had the type set to `\glsdefaulttype`.

This message will be removed once the problem has been fixed.

1

Introduction

1.1 Biosignals and Challenges of a Data-Driven Society

It has never been easier to gather information about any aspect of our life, work, health, industry, education, and society. The continuous collection of data through the usage of mobile phones, smartwatches, hearables, wristbands, and other non-invasive wearable sensors provides a valuable volume of information. This data often comes in the form of time series, which is one of the most common data types in nature [66]. As reported in *Tankovska et al.*, the use of wearable devices has more than doubled in the interval between 2016 and 2019, reaching 722 million [141]. This leads to a large volume of time series data being gathered in all possible scenarios.

More specifically, one time series data type with special interest in our society are *biosignals*. Biosignal-based technology has been increasingly available in our daily life, being widely applied in, among others, biometrics, human activity recognition [32, 33, 7], sports sciences [84, 91, 106, 73, 64, 105, 63, 159, 43, 116], health care [155, 140, 100, 23, 109, 19], rehabilitation assistance, ergonomics [129, 128], and edutainment.

Having relevant information about a subject can be beneficial, but the overwhelming amount of data being collected brings tremendous challenges in the ability to save it, process, analyze and retrieve interpretable and meaningful information from which we can act upon[142]. Ultimately, it becomes even harder to have data that is well structured and labeled, considering that it is a sensitive and time consuming process, and complexity increases with data quantity. In the work of *Roh et al.* it 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 [126], and this is just an example of how much data can have limited use or even go unused. This is particularly problematic when developing machine learning applications, as data should be correctly pre-processed and labeled so it does not include noise, artifacts or mislabeled segments of the signal [126].

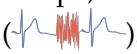
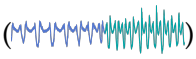
In order to do more with the data we have, tools should be available to support and help analysts to accelerate the process of information retrieval from time series. We believe that having more informative, expressive, and intuitive methods for the analysis of such

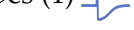
data can help researchers that are familiar with time series data mining, such as data scientists, in speeding their analysis tasks and remove some of its burden. In addition, we also believe that such methods could help democratize these tasks to researchers that are not as experienced in this domain, but could benefit from it in their research activities [83].

In this thesis, we explore two main domains of methods for information retrieval in time series, which will be referred to as (1) *unveiling the grammar of time series* and (2) *a language for time series data mining*. In the first domain, we propose a practical and manageable way to automatically segment single channel or multimodal time series. The proposed approach also brings a lucid visual support in interpreting the time series and helps *unveilling* its structure, how it is organized, and how segments are related in time. The second domain focuses

making the search for patterns and events with expressive queries, moving towards human interpretable and readable time series analysis. As the reader may notice, our topics merge the concepts of time series with text. Then, the reader may appreciate that we dwell on what we believe is the *linguistic nature* of time series.

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 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-)experienced 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 to a meaning.

This notion of structure and meaning is a good approximation of what represents the foundation of a language: grammar and meaning [34].

Grammar is generally defined as the book of rules that constitutes the structure of a language and is modeled by the morphology and syntax [34]. 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. Just 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. Our first topic is related to *unveiling* this structure with methods that can parse it.

In addition to a *grammar*, a language also has *meaning*. The *meaning* on time series depends on the context and what occurred in the physical world that is seen on the signal. Specific occurrences might be attributed a specific meaning by an analyst, as we have seen above with the physician example. In this work, we explore a language to *translate* time series into text and use this textual information as an expressive way of searching for meaningful events and patterns. This is related to our second topic, which focuses on using language in time series data mining tasks.

Until now, we have been using the term *time series*, but the thesis is entitled *A Language for Biosignals*. Having explained how time series have a *linguistic nature*, we now focus our attention to a specific domain of time series, *biosignals*, which are time series that come *from* the *human body*, such as the *heart* (ECG -), *muscles* (EMG -), *brain* (Electroencephalogram (EEG) -) or even movements (Inertial Motion Unit (IMU) -). Considering that the tools developed can be employed in any time series domain, we will use this term instead, but we will give most of our examples from occupational *biosignals*, with a special interest in showing how these can be helpful and meaningful in the context of occupational health.

1.3 Biosignals: Context and Relevance in Occupational Health

This thesis was developed in a strong partnership with the *Ergonomy* team from *Volkswagen Autoeuropa*. Therefore, the central domain of the application regarded the analysis of *biosignals* from workers to retrieve meaningful information about their occupational risk status and prevent Work Related Musculoskeletal Disorder (WMSD)s. 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 [69], accounting for the second-largest responsibility to disability worldwide [97]. These are especially prevalent as upper limb or neck disorder (with 42% of all WMSD cases reported) [57] 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 absenteeism, early retirement, and loss of productivity [144, 145].

Several strategies have been implemented to identify, regulate and prevent occupational risks in manufacturing industries, such as (1) the inclusion of job rotation schedules, which promote a variation of the exposure throughout the working day [12, 123] 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 Repetitive Worksheet (EAWS) [115, 104, 133]. Nevertheless, these strategies are not optimal because they (1) are not automated, relying on observational methods and dedicated personnel 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 [127]. An example of these incentives is the usage of *biosignals*, with wearable inertial devices for physiological, motion, and posture tracking of workers.

From IMU, 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 [26, 148]. There are some limitations to using IMU, mostly related to 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 on a fixed setup of cameras, which is unmanageable in real industrial scenarios [128]. In addition to motion sensors, the inclusion of physiological sensors, such as ECG, EMG and even functional Near InfraRed Spectroscopy (fNIRS) can give reliable evidence of other occupational health variables, namely cardiovascular load, muscular activation, cognitive effort and fatigue [139, 82, 62, 51].

The usage of biosignals in this context can play an important role in supporting the decision of ergonomists and other professionals in the industry. To develop systems that can use physiological, 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 scenarios might have interruptions or changes in the working behavior, due to abrupt production stoppage, shift breaks, or even because the worker shifted to another workspace that has a different movement 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 rotate a screw, 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 studying 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 on these problems, and how they contribute to providing relevant visual feedback for information retrieval from the occupational data and make the search for specific patterns more intuitive and expressive, even for non-experienced data analysts, such as ergonomists.

1.4 Research Paths

The previous sections introduced the topics explored in this thesis for information retrieval on time series, our main motivations to develop the proposed methods, and how these can have significant contributions in the biosignals domain, more specifically for occupational health data.

The work in this thesis contributed to all layers related to time series, from the moment data is acquired (*sensing*), processed for information retrieval (*analysis*), and how it is used to act upon (*decision making*). From these, the presented work in this document will especially address 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. **Analysis** - In this topic, three main research paths are explored with specific research topics. **A -** (1) study 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 based on their similarity. 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 and the relationship between segments can be made. **B -** explore a symbolic representation of time series and a word feature-based representation 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 ultimately be used to be more aware of why a signal is different from the others.

3. **Decision Making -** Discuss how the developed methods can contribute to more aware and informed decisions. Considering the outputs of the methods developed in research path A, how can the analyst gain intuition over the structure of the data associating it with what happened in the physical world. In what regards to research

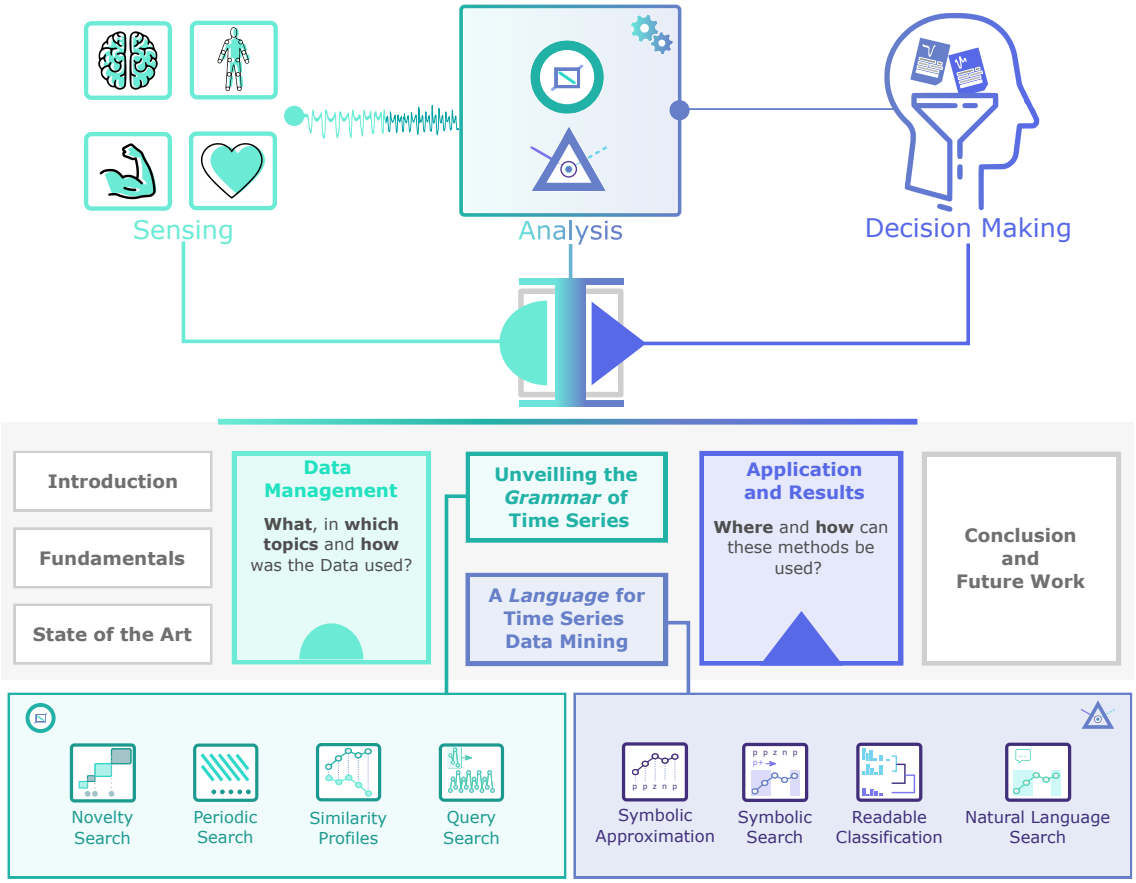


Figure 1.1: General topics contemplated on this thesis and structure of the document. It highlights the 3 layers of involvement related to time series: Sensing, Analysis, and Decision Making, focusing on the Analysis layer, which includes two major topics subdivided into 4 sections each.

path B, how expressive is the process of searching for specific events and patterns with the proposed linguistic-based search methods.

1.5 Thesis Structure

This thesis provides a detailed description and explanation of the research work developed during the Ph.D. program. It is organized into nine Chapters, each contributing to telling the story of this thesis. The reader may appreciate Figure 1.1, which illustrates a guideline of the structure of this work, with a short description of each Chapter's topics and content. **Chapter 1** introduced the main motivations, goals, and context for the development of the proposed methods. **Chapter 2** provides the reader with the fundamental definitions and knowledge needed to have a clear picture of what is developed in this work.**Chapter 3** depicts the state-of-the-art works related to what we developed, namely in the topics of segmentation, summarization, pattern/event search, and classification.**Chapter 4** describes the data we used, explaining its source for both private data and publicly available data, for which purposes it was used and how it was

used in this work.**Chapter 5** explains the algorithm developed for time series structural information retrieval (Unveiling the *Grammar* of Time Series) and provides examples of its usage for novelty segmentation, periodic segmentation, similarity profiles, and query search. *Chapter 6* covers the usage of language for time series data mining (A *Language* for time series data mining), more specifically introducing a novel symbolic approximation and how it can be used for pattern search and classification. In addition, it also explains how to use natural language with a word-feature-based representation of time series. **Chapter 7** shows the application of the previous methods to an exhaustive set of examples, namely from the occupational scenario, and presents major results. In addition, this chapter also provides a general discussion of these and how the proposed methods can be used for the benefit of the analyst. **Chapter 8** gives an overall remark on the outcomes of this thesis and a reflection on the contributions that the developed methods have in making time series preparation and data mining more expressive, quicker, and more practical for an ever-increasing number of data available. It also provides the reader with a clear idea of which are the future paths for this work in terms of novel applications and performance improvement.

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 introduce the concepts and topics covered and addressed, provide the definitions and define the used notation in this work. We start by explaining global definitions related to *time series*, which is the data of interest in this work. Then, examples of *time series*, in particular biosignals, are given. Further, standard pre-processing methods, representation forms, and distance measures are also explained.

Additionally, as this work makes a strong connection between time series and their textual nature, the association between text and time series is introduced in this chapter. Finally, we also explain the standard validation metrics used to validate the proposed methods.

3.1 Global Definitions

The information gathered by sensors is a physical quantity that varies with time. These are called *time series* and are the main topic of this work.

Time Series: A *time series* is a sequence of real values ordered in time with length $n \in \mathbb{N}$: $T = (t_1, t_2, ..., t_n)$. A *biosignal* is a category of *time series*. Several data domains rely on the *multidimensional time series* acquisition from one sensor's multiple axes, such as an accelerometer's three directions, or multiple sources, such as an IMU that fuses three different sensors.

Multidimensional Time Series: A *multidimensional time series* is a set of $k \in \mathbb{N}$ *time series* belonging to the same acquisition: $\{T_1, T_2, ..., T_k\}$. Segments of interest, called *subsequences*, are often searched inside a *time series*.

Subsequence: A *subsequence* is a segment of a (*multidimensional*) *time series* with size $w \in \mathbb{N}$, starting from a given position i and ending at position $i+w$. Therefore, two instants,

defined as *events*, delimit a *subsequence* in time.

Event: 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. Hence, *event* detection is often considered *time series* segmentation or change point detection[25]. To be clear, we will use the terms *event detection* and *segmentation* when discussing our methods, but can eventually use the term change point detection when comparing with other methods. A common strategy used in time series data mining to find relevant *subsequences* or *events* is the moving window.

Moving Window: A *moving window* is a process of sliding along a time series T to apply a specific method on each *subsequence* it hovers, a common strategy used in *time series* data mining to find relevant *subsequences* and *events*. The window has, similar to the *subsequence*, a predefined size $w \in \mathbb{N}$, which starts at a given position i and ends at position i+w. The process of *moving windows* is iterative, and windows can overlap each other. The next window will start at i+o, where $o \in [1, w]$ is the overlapping size ($o = 1$ for total overlap and $o = w$ for no overlap). On each *moving window* from each *subsequence* of the (*multidimensional*) *time series*, features can be extracted to form a *feature series*.

With a moving window, each *subsequence* of a *time series* 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 apply standard distance measures. More importantly, we start by explaining the importance of a moving window to represent a *time series* with a feature, a *feature series*.

Feature Series: A *feature series* is a feature representation of a time series with size $m = \frac{n}{w-o}$ that depends on the overlap size $o \in \mathbb{N}$ of the *moving window*. In the case of a *multidimensional time series*, the *feature series* stack a *multifeature series* with size $f_{k,m}$. Multiple features extracted from one dimension or various dimensions are grouped into a *feature matrix*.

Feature Matrix (F_M): A *feature matrix* with size $r \times (k \times m)$, represents that each of the k dimensions produces r features. This *feature matrix*, which characterizes the (*multidimensional*) *time series* in statistical, temporal or spectral domains, is used to compute the *self-similarity matrix*.

Having introduced essential concepts for the understanding of the next few sections, we further give an overview of essential techniques used in time series processing and analysis, namely filtering methods, representation methods, distance measures and its applications, and a formal explanation of the relationship between text and *time series*.

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, with the following frequency response:

$$H_{j\omega} = \frac{1}{\sqrt{1 + \epsilon^2 \left(\frac{\omega}{\omega_c}\right)^2}}, \tag{3.1}$$

where n is the order of the filter, ω the frequency ($\omega = 2\pi f$), and ϵ the maximum amplitude gain.

3.2.2 Smoothing

Another method often used to reduce the presence of noise and is a variant 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, to calculate the mean in each iteration. Other approaches also convolve the signal with a specific window (H) (e.g. *Hanning window*), which instead of giving the same weights to all the samples of the moving window (moving average), attributes a higher weight to the moving window's central samples.

$$Tm_i = \Sigma_{j=a}^{a+w} T_j H_{j-a}, \tag{3.2}$$

where Tm_i is the i^{th} smoothed sample of the time series T, segmented by a and $a + w$ (w is the size of the moving window) and H is the window function used to smooth the signal.

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 it. 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:

$$\overline{T} = \frac{T}{max(|T|)}, \tag{3.3}$$

where 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:

$$\overline{T} = \frac{T - min(T)}{max(T) - min(T)}, \tag{3.4}$$

where the time series 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:

$$\overline{T} = \frac{T - \mu_T}{\sigma_T}, \tag{3.5}$$

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 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 to this thesis will be explained.

3.4.1 Spectral Transformation

One of the first and most well-known techniques suggested for time series transformation is the DFT [2]. The idea behind this concept is that any signal, either periodic or not, is a decomposition of a finite number of sine and cosine waves, from which amplitude and phase are used to represent the signal in the frequency domain [138]. This transformation

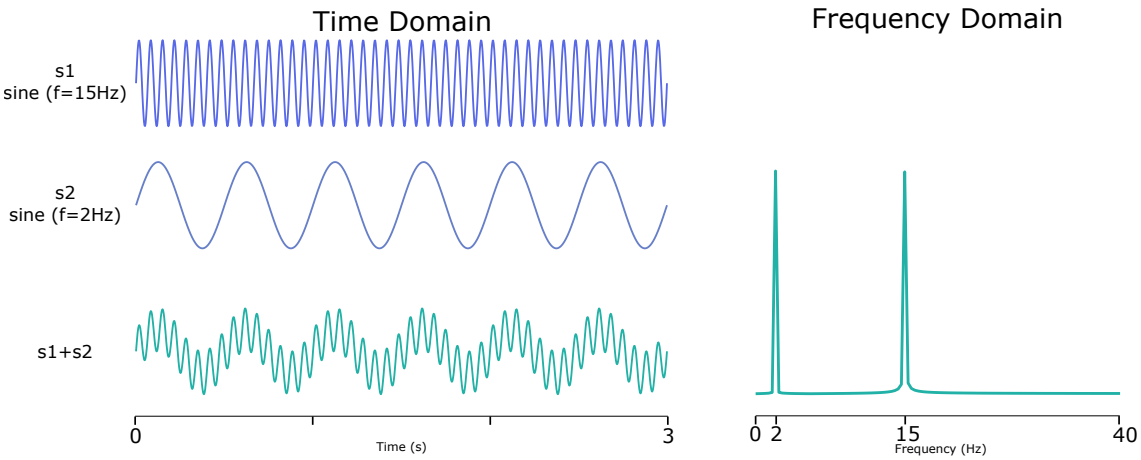


Figure 3.1: DFT of a sum of sine waves.

allows us to see the signal differently, highlighting which frequencies concentrate more or less energy. It unveils the presence of specific types of noise or artifacts, or periodic shapes. Figure 3.1 shows the transformation of a signal into the frequency domain. The signal is the sum of two different sine waves with 2 and 15 Hz respectively. The result of the Fourier transform is a frequency series with high energy at the frequencies of the sine waves.

The frequency domain opens new possibilities for the analysis of the time series, namely for the extraction of spectral features, which are also used to represent the signal in the feature domain.

3.4.2 Feature-based Representation

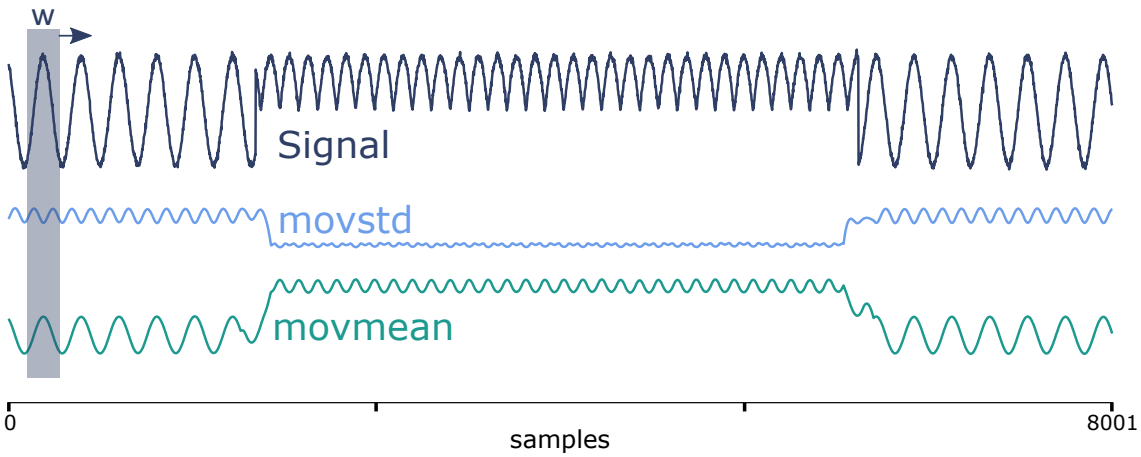


Figure 3.2: Moving window used to extract features with total overlap. The mean and standard deviation are extracted from the signal. The left shows the sine waves, while the right shows the frequency spectrum of the combination of sine waves.

The process of feature extraction is also a transformation method commonly employed and is performed by a *moving window*, resulting, for each feature, in a feature series (see

Section 3.1). On Figure 3.2 is showed a time series from which the average (moving mean) and standard deviation (moving *std*) are computed with a moving widow of size $w = 100$ and an overlap $o = 100\%$. The feature is a characterization of the signal for that specific extracted property and is commonly used in machine learning tasks. In this thesis we will present two main applications of the feature representation of time series.

3.4.3 Piecewise Aggregate Approximation

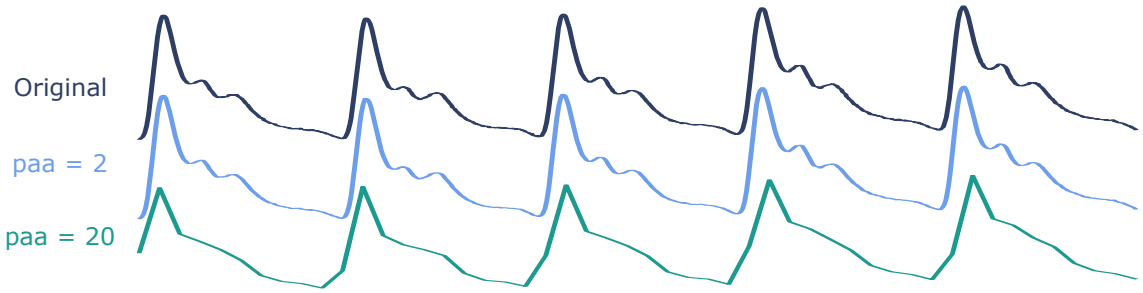


Figure 3.3: PAA representation of a ABP signal, with window sizes of 2 and 20, repsectively. The PAA representation was computed with the PYTS module [44], based on [78].

Another common used transformation method to simplify a time series and reduce its dimension is the PAA) [78]. The new representation space will have size $1 < N \leq n$, in which N is a factor of the original size n. The method 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 [78]:

$$\overline{t_i} = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} t_j. \tag{3.6}$$

An example is showed in Figure 3.3, where a ABP signal is transformed into a PAA with sizes 2 and 20, respectively.

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 SAX [86]. 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 symbolic *alphabet* chosen. Figure 3.4 shows an example of the signal 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.

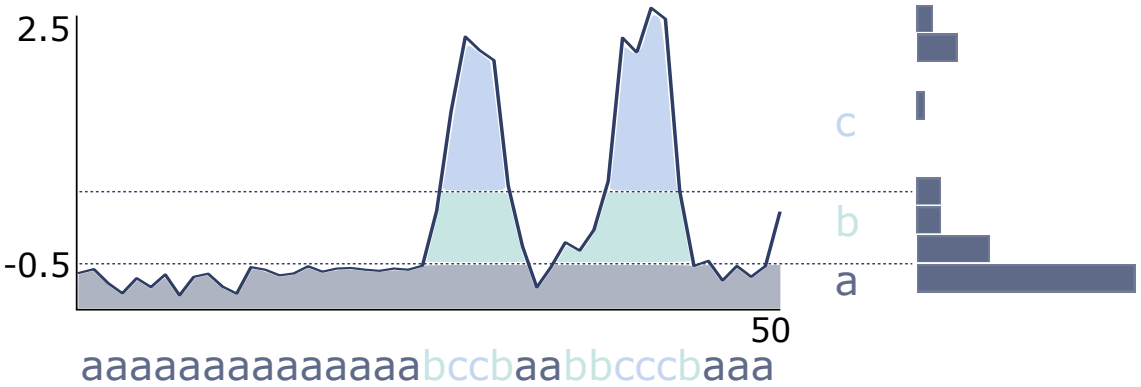


Figure 3.4: SAX representation of a power consumption signal from a Dutch Company, with window bin size of 3. The SAX representation was computed with PYTS based on [86].

This representation method is relevant to be introduced as it was a strong inspiration for the proposed novel symbolic representation technique for time series used for expressive pattern search and classification.

In order to perform pattern search or classification tasks, we have to calculate the difference/similarity between two time series or *subsequences*.

3.5 Distance Measures

There is a large number of distance measures for time series, but two of the classical and standard measures still provide state-of-the-art results in most time series data mining tasks, namely the ED and the 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

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

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

$$ED(Q, C) = \sqrt{\Sigma_{i=1}^{n}(q_i - c_i)^2}, \tag{3.7}$$

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 amplitude and offset distortion, phase

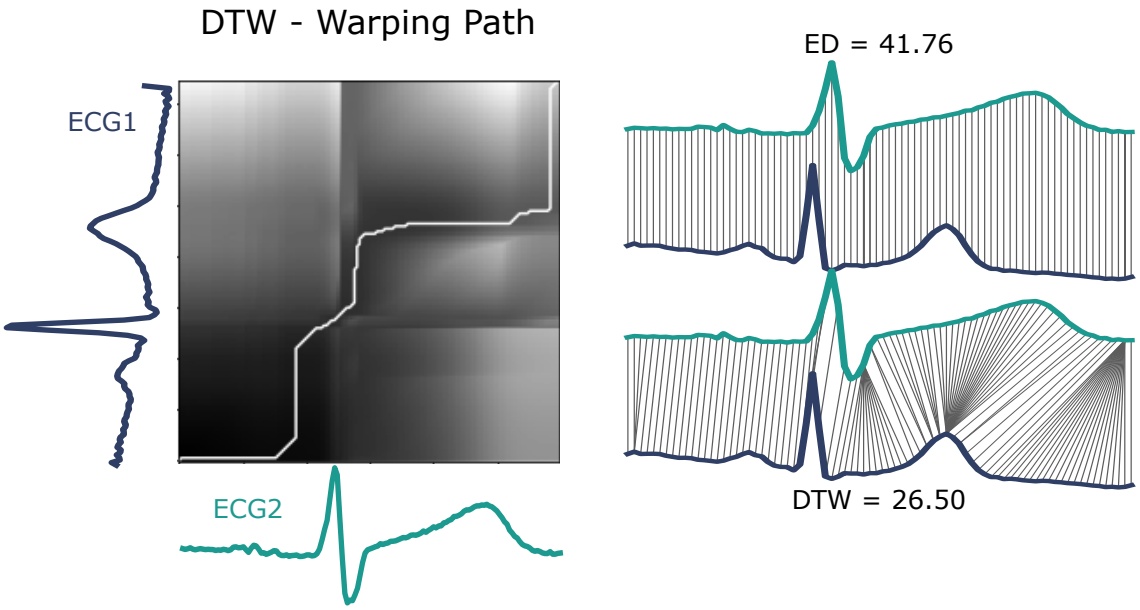


Figure 3.5: DTW and ED distances on two different ECG signals.

distortion, and local scaling ("warping") distortion. The first can be compensated by the z-normalized ED [16]:

$$z\_ED(Q,C) = \sqrt{2m(1 - \frac{\Sigma_{i=1}^{m} Q_i C_i - m\mu_Q\mu_C}{m\sigma_Q\sigma_C})}, \tag{3.8}$$

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 $d = (q_i - c_j)^2$ [76]. Figure 3.5 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 represents the set of matrix cells that minimize the warping cost, also defined as the cumulative distance of these cells [76].

$$W = w_1, w_2, ..., w_k, ..., w_K; \qquad max(m,n) \leq K < m + n + 1, \tag{3.9}$$

$$DTW(Q,C) = min\sqrt{\Sigma_{k=1}^{K} w_k}. \tag{3.10}$$

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:

$$\gamma(i,j) = d(q_i, c_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}. \tag{3.11}$$

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 3.5 shows an example of applying the ED and DTW on two different PQRS complexes from different ECGs.

3.5.3 Complexity Invariant Distance

A different type of distance measure is also used to cope with complexity invariances. This distance uses a complexity correction factor (CF) with an existing distance measure, such as ED [16]:

$$CD(Q,C) = ED(Q,C) \times CF(Q,C). \tag{3.12}$$

The CF is defined as [16]:

$$CF = \frac{max\{CE(Q), CE(C)\}}{min\{CE(Q), CE(C)\}}, \tag{3.13}$$

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[16]:

$$CE(Q) = \sqrt{\Sigma_{i=1}^{n-1}(q_i - q_{i+1})^2}. \tag{3.14}$$

These distance measures are performed on the original representation domain of the time series. As we showed above, other representation techniques can be employed, creating opportunities for other types of approaches. In this work, we propose other representation techniques to create novel ways of exploring time series. Therefore, other distance measures that can be used in different representations of time series will be explained.

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. Each *subsequence* scanned by the *moving window* is characterized by the feature value, and a 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 (to be clear, a feature vector is the set of feature values for a *subsequence* of the time series, while a feature series is a feature representation of the entire time series).

Vector-based distance measures can be used with feature vectors to compare different time series or *subsequences*. There are several vector-based distance measures, including the already mentioned ED 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 [59] (equation 3.15).

$$CS = \frac{\vec{f_A} \cdot \vec{f_B}}{||\vec{f_A}||||\vec{f_B}||} \tag{3.15}$$

being $||\vec{f_A}||$ and $||\vec{f_B}||$ the euclidean norm of each feature vector, defined as $\sqrt{\Sigma_{i=1}^{r} f_{Ai}^2}$ and $\sqrt{\Sigma_{i=1}^{r} f_{Bi}^2}$, respectively [59]. Because the cosine similarity includes a vector normalization, it does not rely on the magnitude of the vectors and the difference is solely due to the angle between vectors. This is relevant for domains where vectors have high dimensionality, which is the case for feature vectors.

3.6 Applying Distance Measures

Measuring distances between any time series give the ability to compare them. It is the fundamental instrument for most time series data mining tasks. With a distance measure, we can compare groups of time series for classification purposes or compare *subsequences* with a query template to 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 profiles.

3.6.1 Distance Profile

As mentioned in the previous subsection, measuring all the distance pairs of a time series provides the ability to retrieve relevant structural information. When computing the distance of a *subsequence* to all the other *subsequences* of the time series, a *distance profile*

is calculated. Each *subsequence* can have a *distance profile* and when computing all the distance pairs, a self-distance matrix is the result.

Recently, a strategy was proposed to compute a one-dimensional distance *profile* for a time series based on a z-normalized ED matrix. By keeping the lowest value of each *distance profile* (nearest *subsequence*), we retrieve the *matrix profile* [54]. The result gives the minimal distance pair of each *subsequence*, meaning that minimum values are *motifs* and maximum values are *discords*.

Motif: The *subsequence* pair that has the lowest distance forms a motif. That means that on the entire time series, these two *subsequences* are the closest ones. The opposite is a *discord*.

Discord: The *subsequence* pair that has the highest distance forms a *discord*. That means that on the entire time series, these two *subsequences* are the furthest apart.

3.6.2 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 [110]. Each are relevant for segmentation and summarization tasks.

Let X be a sequence with size $N \in \mathbb{N}$ 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 SDM with size $N \times N$ can be computed, such that:

$$SDM(i, j) = d(x_i, x_j), \tag{3.16}$$

being d a distance measure between elements $x_i, x_j \in X$ for $i, j \in [1 : N]$. $SDM(i, j)$ represents a cell of SDM that contains the distance value. When $i = j$ the distance should be zero, therefore the diagonal of SDM has the lower values. Besides the main diagonal, other relevant structures can be found in SDM. These include *homogeneous blocks* and *paths* [110, 111], as can be seen in Figure 3.6.

Areas with lower distance and block structure along the diagonal are highlighted as *homogeneous* segments. 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 SDM 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. In Figure 3.6, changes in the ABP signal are visible as blocks. In total, seven regimes are present on the signal, and seven blocks can be counted along the main diagonal.

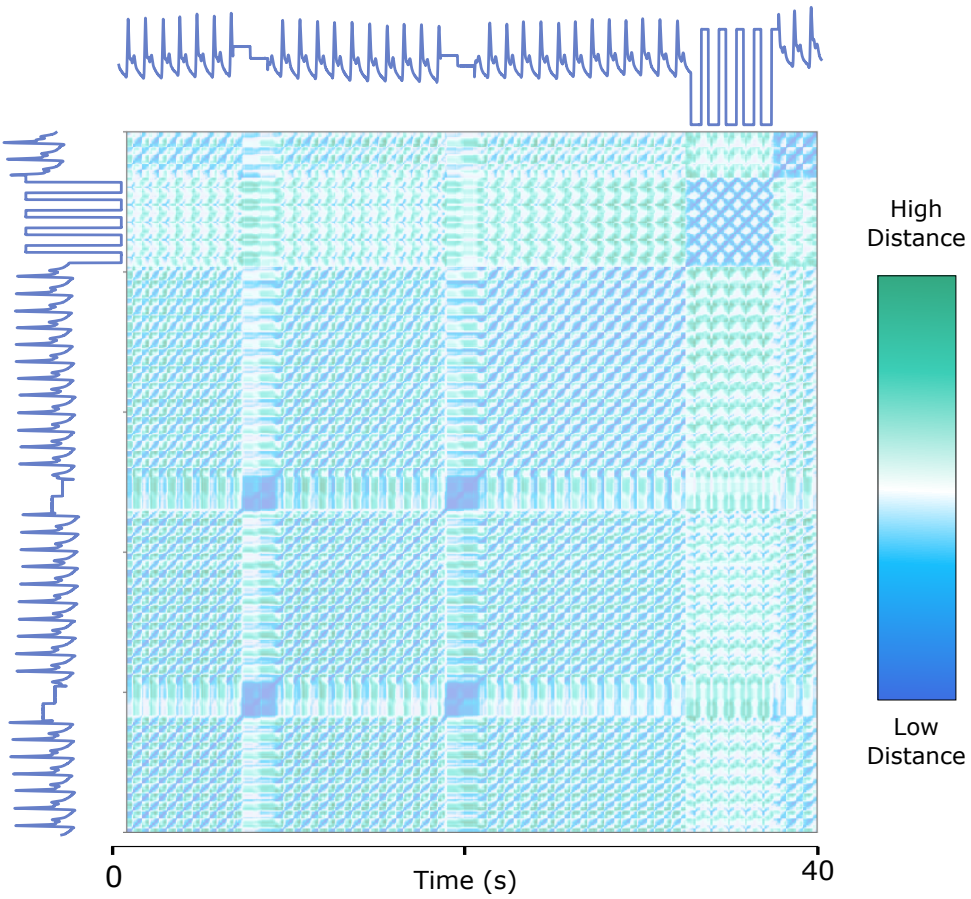


Figure 3.6: Self-Distance Matrix (SDM) representation of the ABP signal. The matrix was computed using the cosine distance between the feature representation of the signal. The main structures that can be highlighted are *blocks* along the main diagonal of the matrix, *paths* that parallel to the main diagonal, and *cross-paths* that are perpendicular to the main diagonal. The signal comes from a public dataset (see Section **??**).

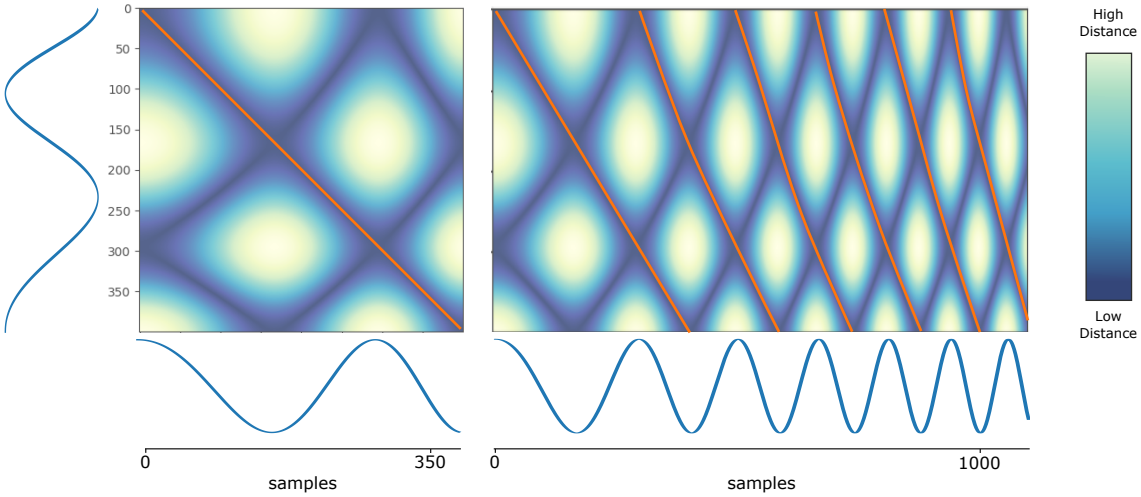


Figure 3.7: *SDM* of a chirp signal computed with the pairwise euclidean distance. The chirp signal is a sine wave with increasing frequency (in this case from f_0=1Hz and f_1=25Hz)[146]. Left) *Subsequence* of the chirp signal compared with itself. Right) The same *subsequence* is compared with the entire chirp signal.

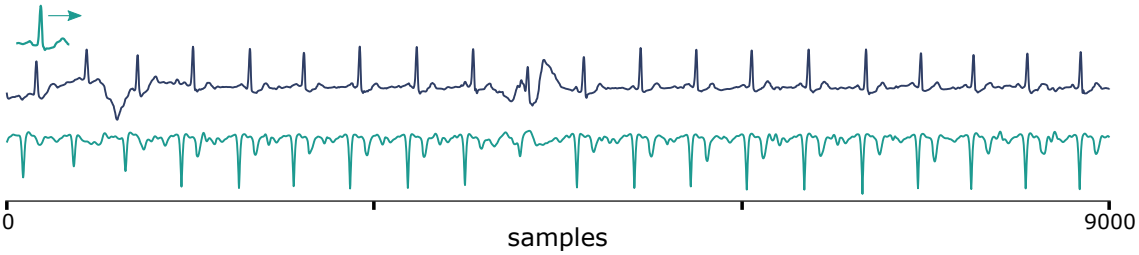


Figure 3.8: Distance profile of a query based search using the z-normalized ED. The template was a PQRS complex of an ECG.

Repeating *subsequences* in the time series are represented by *paths* on SDM. *Paths* are observed on the SDM because of matching *subsequences* during the sliding process, which computes consecutive low distance values represented in a diagonal form on the matrix. If the *subsequences* being compared are exactly the same, the diagonal is perfect, but differences might be found, which results in a distorted diagonal. In Figure 3.7 we show an example of an SDM with a chirp signal that has a continuously increasing frequency. A *subsequence* of the signal is compared with itself (Figure 3.7.left) and with the entire chirp signal (Figure 3.7.right). We highlighted the relevant *paths* in orange. When comparing the signal with itself only, the *path* is a perfect straight line, but when compared with the entire signal, the *paths* are continuously more distorted and shorter. These show where each period of the sine wave occurs, and the distortion results from the increasing frequency of the signal. In the SDM from Figure 3.6, *paths* are visible on the matrix, indicating the presence of periodicity. The *cross-paths* indicate the same periodic behaviour, but also that the *subsequence* is symmetric.

It is relevant to highlight that as distance matrices can be computed, similarity matrices can as well, following the same equation 3.16, but using a similarity measure (s) instead of d. Further, we will only mention the similarity matrix, which will be called SSM, as the proposed method uses the cosine similarity.

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.

An example of using a query-based search is illustrated in Figure 3.8, where a PQRS complex of an ECG is used as a template to search for all the other complexes. The distance computed is the z-normalized ED. The result shows a distance profile from which *minima* indicates the match with the template.

Other methods can be used to perform query-based search tasks, even using other types of templates, which can be a *drawing* [102] or text [3]. In this work, we will introduce

novel ways of performing a query-based search with regex and natural language.

3.6.4 The k-Nearest Neighbors

Having distance measures we can compare signals for several purposes, namely classification. One traditional supervised algorithm for this purpose is the k-Nearest Neighbor (NNbr). The assumption of this method is fairly straightforward: new examples will be classified based on the class of their NNbr, that is, the class of the example is the average of the class of its k-NNbr. The process has two steps: (1) finding the k-NNbr and (2) choosing the class of the example based on the neighbors [36].

To find the NNbr, the distance from the example to all the time series of the training set has to be computed. The k-NNbr are the k training examples with the lowest distance. Having the NNbr, the next stage is the determination of the example's class, which can be made with several strategies, such as majority voting, or distance-weighted voting [36].

In this work, we will propose a novel method for time series classification that will have its performance compared with a 1-NNbr based on the z-normalized ED. The proposed method is from the text-domain. As we have been mentioning in this manuscript, parts of the work developed are tightly connected with text mining methods by making a bridge between time series and text. It is then relevant to explain this relationship and how we can use it for time series data mining applications.

3.7 Text Mining on Time Series

3.7.1 Time Series Textual Abstraction

In SAX (see Section 3.4.4), 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*. As a novel symbolic representation of time series is proposed in this work, it is relevant to give the general background that helps make the association between the original time series, how it can be transformed into a symbolic time series, and the text notation that can be used for this purpose. Note that this is an introductory explanation that will be further contextualized when needed throughout this manuscript. We start with a few fundamental definitions.

Character: A *character*, c, is an unit symbolic element belonging to the vocabulary V that represents a sample or *subsequence* of a time series, such that $c \in V$. Each sample of a time series is transformed into a *character* to form a *symbolic time series*.

Symbolic Time Series: A *symbolic time series* is a sequence of *characters* ordered in time generated from the parent time series. It has length $n \in \mathbb{N}$: $ST = (st_1, st_2, ..., st_n)$, with $st_i = c \in V$. A specific sequence of *characters* of a *symbolic time series* can form a *word*.

Word: A *word* can be a single character or the concatenation of a sequence of *characters*, giving a textual representation of a *subsequence*, such that $W = (w_1, w_2, ..., w_u)$, with $u \in \mathbb{N}$, $w_i = c \in V$ and $W \in V$. Putting *words* together forms a *sentence*.

Sentence: A *sentence* S represents a group of *subsequences* of the time series, or the entire time series itself. It is formed by joining sequences of *words*, such that $S = W_1, W_2, ..., W_s)$, with $s \in \mathbb{N}$ and $W_i \in V$.

Document: The set of *sentences* in a time series are called a *document*. The *document* D can be computed by one or more sentences. In this work, a *document* represents the textual description of the entire time series.

Corpus: The *corpus* is a collection of text material (group of *documents*). It represents a higher level of textual information that is used in case we have a dataset with multiple time series. 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 an entire time series dataset.

Vocabulary: The *vocabulary* V comprehends the set of all different *characters* and *words* present in all time series of the dataset.

3.7.2 Text Features

Here are introduced traditional methods applied for feature extraction of text data, namely the BoW and TF-idf.

Bag of Words: A BoW is a feature matrix representation of a corpus, being the feature the number of occurrences of each *term* (in this case it can be a *character* or a *word*), called the tf:

$$bow(t, d) = tf_{t,d} = \frac{f_{t,d}}{\sum\limits_{t' \in d} f_{t',d}}, \tag{3.17}$$

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 a *word* or an *n-gram*.

N-gram: It is a span of followed *words* that are counted in the BoW/TF-idf. As an example, a possible *2-gram* from Figure 3.4 would be `aa`, `ca` or `ac`. This strategy resembles a *moving window* with total overlap on time series, but for text. It can make the BoW/TF-idf model more robust since it relies in more than single *word* statistics. It also has a very good fit for time series, considering that it can take into account time dependencies between *words*, which reflects the time dependency seen in time series between *subsequences* (e.g.

it might be more meaningful to say that a signal has a `peak` next to a `valley` than just saying that it has a peak and a valley).

The BoW 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*, then it is less relevant. To counteract this possible limitation, the TF-idf matrix is used.

Term Frequency Inverse Document Frequency: 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:

$$idf(t, D) = \log \frac{L}{|\{d \in D : t \in d|\}} \tag{3.18}$$

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

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{3.19}$$

Both BoW and TF-idf 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 BoW and the fact that it contains discrete features, it is suitable to use in naive Bayes classifiers. In the other end, the TF-idf is typically used with linear SVM classifiers [118].

3.7.3 Text Pattern Search

When writing or reading a document, we often use shortcuts or other commands to speed up the task of searching for specific words or expressions. In some cases, we might not be searching specifically for a character or word, but rather for a text structure or text pattern (for example, one might be searching for all the emails available in a text document). This type of search is comparable to querying in time series, being the most convenient method to perform this type of search in text a regex.

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

In order to understand the way that regex work, some of the most used characters and regex primitives are presented as follows [50]:

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* → [ssm, sam, sim]
[]	Matches anything inside the brackets	*wom[ae]n* → [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	*?<!\ d*.+* → [~~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]

Table 3.1: Main regex operators and meta-characters. Each is presented with a simple example of a good match for a possible regex. A description is also made for complementary understanding.

3.8 Performance and Validation Measures

This manuscript discusses three main time series data mining problems, namely classification, segmentation, and event/pattern detection. In order to perform a validation of the work developed, standard procedures are used. In this section are explained which procedures are typically used to evaluate the performance of algorithms in the mentioned problems.

3.8.1 Classification Problems

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 depends on the problem. These measures are used in this thesis to

evaluate the performance of classification and event detection algorithms.

In time series classification problems, a labeled dataset is divided into training and testing sets. The algorithm is trained on the training set and a final validation is made 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*, *TN*, *FP* and *FN*.

- TP_c - If the label predicted by the algorithm is equal to the target class (positive when positive);
- TN_c - If the label predicted is correctly not the target class (negative when negative).
- 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).

$$Precision(P) = \frac{TP}{TP + FP} \tag{3.20}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{3.21}$$

$$F1 - score(F1) = 2 \times \frac{P * F}{P + R} \tag{3.22}$$

$$F1 = \frac{TN + TP}{TN + TP + FP + FN} \tag{3.23}$$

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

$$macroP = \Sigma_{i=0}^{c} \frac{P_i}{c} \tag{3.24}$$

$$macroR = \Sigma_{i=0}^{c} \frac{R_i}{c} \tag{3.25}$$

$$microP = \frac{\Sigma_{i=0}^{c} TP_i}{\Sigma_{i=0}^{c} TP_i + \Sigma_{i=0}^{c} FP_i} \tag{3.26}$$

$$microR = \frac{\Sigma_{i=0}^{c} TP_i}{\Sigma_{i=0}^{c} TP_i + \Sigma_{i=0}^{c} FN_i} \tag{3.27}$$

These metrics are typically supported with a *confusion matrix*, which displays a 2D-map of the ground truth labels *versus* 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 only by searching if the ground truth sample is found, we calculate the *TP*, *FP* and *FN* based on a tolerance margin. From these measures, metrics from Equations 3.20, 3.21 and 3.22 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):

$$MAE = \sum_{i=1}^{k} \frac{|g_i - e_i|}{k} \tag{3.28}$$

$$MSE = \frac{1}{k} \sum_{i=1}^{k} (g_i - e_i)^2 \tag{3.29}$$

$$ME = \frac{1}{k} \sum_{i=1}^{k} (g_i - e_i) \tag{3.30}$$

The precision measure is relevant to indicate if the method can 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 - MsE).

3.8.3 Evaluating Query Complexity

As we are introducing novel ways of performing more expressive query-based searches with regex, we are measuring the legibility and difficulty in generating a query. In the programming field, *Halstead* measures are typically used for this purpose. These measures describe the complexity of the script directly from the source code, based on a set of metrics

calculated with the number of distinct operators (oprt) and operands (oprd), and the total number of operators (Toprt) and total number of operands (Toprd). These metrics are the [74]:

Vocabulary The number of distinct operators and operands that belong to the script:

$$Voc = oprt + oprd \tag{3.31}$$

Length

The total number of operators and operands that belong to the script:

$$Lgth = Toprt + Toprd \tag{3.32}$$

Calculated Length

It uses the entropy measure to calculate the average amount of information based on the number of distinct operators and operands:

$$CL = oprt * \log_2(oprt) + oprd * \log_2(oprd) \tag{3.33}$$

Volume

It measures the amount of information that the reader has to absorb to understand its meaning. It is proportional to the length measure ($Lgth$) and logarithmically increases with the vocabulary:

$$Vol = Lgth * \log_2(Voc) \tag{3.34}$$

Difficulty

The difficulty in writing or reading the script. It increases more having fewer operands repeated more frequently than having more operands repeated more frequently:

$$Dif = (\frac{oprt}{2} + \frac{Toprd}{oprd}) \tag{3.35}$$

Effort

Measure of the effort necessary to understand what is written and recreate the script. It is proportional to both volume and difficulty measures:

$$Efrt = Vol * Dif \tag{3.36}$$

3.8.4 Comparing Algorithms

In classification problems, a good procedure is to compare the proposed algorithm with the 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 3.9.left and 3.9.right, respectively.

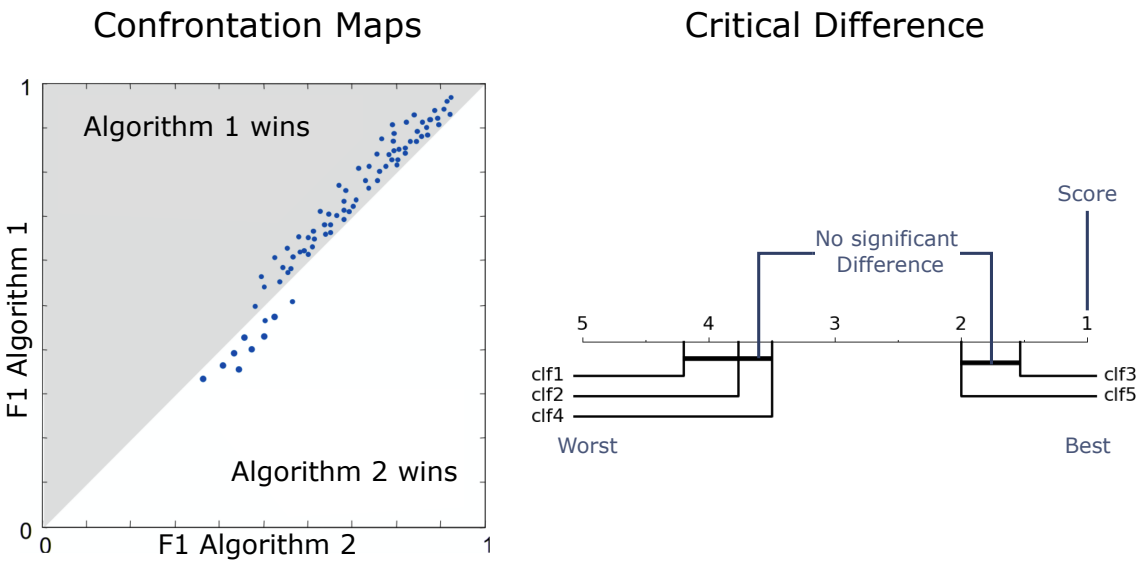


Figure 3.9: On the left are confrontation maps, used to compare classification algorithm performances. On the right is a critical difference map, which shows a statistical difference in the performance of algorithms for general purposes.

3.8.4.1 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 reading it at 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 *versus* algorithm 2. Figure 3.9.left displays an example of it taken from [75]. 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.

3.8.4.2 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 3.9.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 tick bar connects the classifier with performances that are not significantly different in the statistical test.

In this work, we will use an implemented critical difference method from [70], which uses the *Wilcoxon-Holm* test, which counteracts the problem of multiple comparisons and calculates pairwise significance between all methods evaluated [5].

4

State of the Art

In this Chapter, the reader will find an extensive delivery of the existing works regarding the main topics covered in this thesis. This will include (1) existing works for information retrieval in time series, mainly in terms of event detection and segmentation, (2) approaches for summarization of time series, (3) symbolic representation techniques and how these can be used for (5) classification tasks and (6) query-based search mechanisms.

4.1 Information Retrieval from Time Series

4.1.1 Approaches

Prior works in event detection focus on change point detection or segmentation, where the strategies are categorized as online versus offline, univariate versus multivariate, model-based versus non-parametric and unsupervised versus supervised [25, 7, 143].

4.1.1.1 Supervised Methods

Supervised methods include multi-class, binary and virtual classifiers optimized to detect change points [**review_cpd_1**], where the nature of the change can be provided as an additional advantage. Another example uses neural networks with transfer learning for segmentation [103]. Subsequence search might also be a potential method given a well-selected pattern [45]. However, supervised methods rely upon brittle training sets and class imbalance, since there are more in-state sequences than change point sequences [**review_cpd_1**]. An additional problem reported by [25] is that most algorithms' performance was validated in synthetic data, where the given nature of the application was not optimal. In response, a benchmark is available for change point detection [25], where methods can be compared on real data. This study applies this benchmark as a reference of state-of-the-art methods.

4.1.1.2 Unsupervised Methods

Existing classic non-supervised methods in change point detection, such as *Bayesian Online Change Point Detection* (BOCPD) [1], *Binary Segmentation* (BINSEG) [15] and *Segmentation Neighbourhood* (SegNeigh) [13], are witnessed to be able to perform state-of-the-art applications in various domains [25]. However, BOCPD needs hyperparameter tuning for sound performance, while BINSEG and SegNeigh have not been used in multidimensional domains. Besides, these methods have not been reported to cope with a multi-timescale change [25]. The available repository [25] collecting the implementation of some offline methods [143] above lacks a visual output that can provide users with the location of the change points. Another method applicable to real data domains, called *Fast Low-cost Online Semantic Segmentation* (FLOSS) [**floss**, 53], searches regime changes based on the nearest neighbours of subsequences, which allows the similarity comparison between segments for the segmentation and summarization of long-term time series.

4.1.1.3 Self-Similarity Matrix (SSM)

The SSM has been used for segmentation in the audio domain, based on a feature representation of the audio signal [**MuellerZ19_FMP_ISMIR**]. The advantage of the SSM is that it provides a considerable amount of information for a specific timescale. This study promotes SSM concepts and applications from the audio domain to other time series domains. The proposed method can detect events with context, associating the estimated events with patterns, (dis)similarities, periodicity and novelty, and a possible extension is the task of summarization. The search mechanism is primitively based on a specific timescale and can evolve recursively to perform multi-timescale searches.

4.1.2 Applications

We live in an "era of big data" [31]. As mentioned in the Section **??**, wearable sensors are currently available on a large scale, promoting the acquisition of massive amounts of data. Datasets of this size can no longer be handled by trivial means and call for engineers and data scientists with expertise in data mining, machine learning, and data analysis [151]. This increase in wearable usage has also been seen in industrial environments, which is motivated by the current trend of Industry 4.0 [**xu2018industry**], promoting the use of sensors to monitor in real-time their machines for damage prevention, and their workers for occupational-related disorder prevention and productivity improvements [**varandas_evaluation_2019**, **santos_explaining_2020**]. Research areas such as intelligent rehabilitation [88, 0, 0, 0, 0, 0], orthotics [162, 107, 161, 101], sports science [84, 91, 106, 73, 64, 105, 63, 159, 43, 116], activity modelling [**liu2021thesis**, 30, 29, 0], exoskeletons [150, 52], machine learning education [60], and fall detection [28, 114, 154], have also leveraged the power of biosignals from wearable sensors.

In research aspects of time series analysis, biosignals produced by various types of sensors require the data science community to develop tools to extract meaningful information for the acquisition, including reporting, pattern recognition, event detection and periodic signal segmentation and classification, among other data mining tasks [122, 21]. The availability of more reliable data and practical information is more beneficial, primarily as machine learning is increasingly applied. Numerous fields could benefit from our proposed methods, including physiological event detection for healthcare (e.g., noise, sleep problems and epilepsy), biomedical signal analysis (e.g., ECG, EEG and EMG), climate change detection, audio-based automatic speech segmentation and recognition, motion sequence segmentation, behaviour transition detection, human activity modelling, feature space study, and manufacturing industries, among others.

Ultimately, data preparation is essential for data analysis and machine learning application development. After data acquisition, the effort in data processing and preparing implies challenges, an active research subject. One of the critical issues in data preparation is the lack of labelled data. Labelling data is a sensitive and time-consuming process, of which the complexity rises with the data quantity. Nevertheless, accurately labelled data is essential for data analysis and model training: In [126], the authors stated that data scientists rely solely on a small portion of the available datasets because it is too expensive to label all the data. Such a thought reflects how paramount it is to have solutions that can improve the existing data labelling strategies to reduce labour, time costs and ground truth quality.

4.1.3 Segmentation and Event Detection

Most of the works available in event detection are focused in change point detection or segmentation. In this case, we researched the literature for both, as we treat these two problems as the detection of relevant instances in time series where it can be segmented. 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 [25, 7, 143]. Regarding supervised methods, there are multi-class, binary and virtual classifiers, optimized for the purpose of detecting change points [7]. 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 [103]. These methods, however, rely in very brittle training sets and class imbalance, since there are more in-state sequences than change point sequences [7]. Additionally, a problem reported by [25] 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 [25], where methods can be compared on real-data. The proposed work in segmentation 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) [1], the binary segmentation (BINSEG) method [15] and the segmentation neighborhoods (SEGNEIGH) method [13]. These methods have been reported successful in several domains [25]. Still, 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 [25]. An available repository provides an implementation of some of these offline methods [143], but these lack a visual output that might give the user an intuition over where a change point might be. In addition, most of these methods rely in parameter optimizations, which can make them very brittle to other datasets where these were not optimized [54].

Another method, called FLOSS [54], relies in searching change points based on the nearest neighbors of subsequences, being very successful in real data domains. It can also handle online problems and multidimensional datasets. As it searches for nearest neighbors, the similarity between segments might be compared and used for summarization.

4.1.4 Self Similarity Matrix

Similarity matrices are not novel in the analysis of time series. These matrices have several representations and denominations, being the most commonly known *recurrence plot*. These plots are a similarity matrix thresholded by a specific value that highlight only similar regions. The representation of these maps was used to analyze time series [**recurrenceplots1**, 156] and search for reoccurring shapes, anomalies or symmetric behaviors [156]. It was also used for time series classification by inputting the recurrence plot into a neural network (Convolutional Neural Networks) [160, 79] or forecasting [10]. The SSM has more information than the recurrence plot (which has been thresholded). Several usages have been explored in the audio domain, namely for segmentation, thumbnailing, periodiciy search or music alignment. For this, the audio signal is transformed on a feature-based representation [110, 111].

The advantage of using the SSM 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 data types. 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 SSM, 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 [81], hidden Markov models [163], decision trees [72], bayesian networks [72], and semi-automatic process [24], 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 [17, 18]. 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 [96, 95]. 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 [99]. 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 [147].

Other methods rely on a self-similarity approach, namely [113], 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 [158]. Additionally, autocorrelation revealed itself an useful tool, as the search over maximum

values can infer the cyclic nature of the data [17]. 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 [145].

The following work is inspired over an algorithm for the detection of musical structures on audio signals [46, 117, 20] 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 [37, 39, 38].

4.1.5 Summarization

4.1.6 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

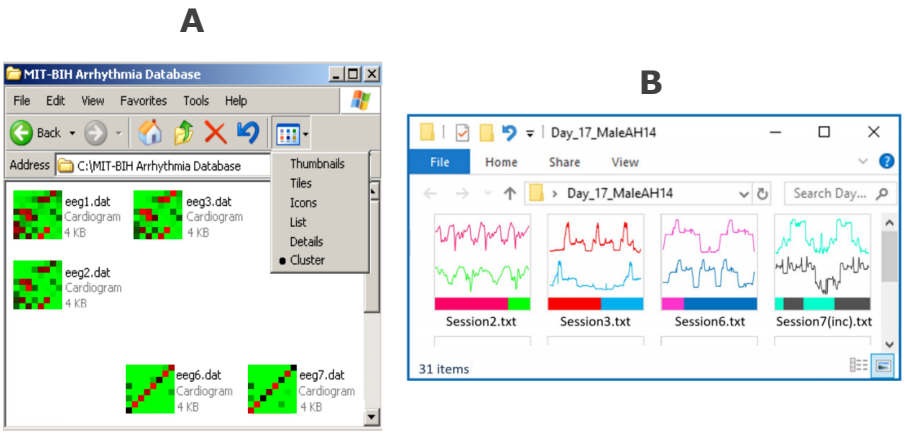


Figure 4.1: Strategies for time series summary found on the literature. These images are taken from the works from [68, 80]

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.

4.1.7 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* [68, 80]. 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 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 4.1.

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 [125, 86].

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 [80, 149]. 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.

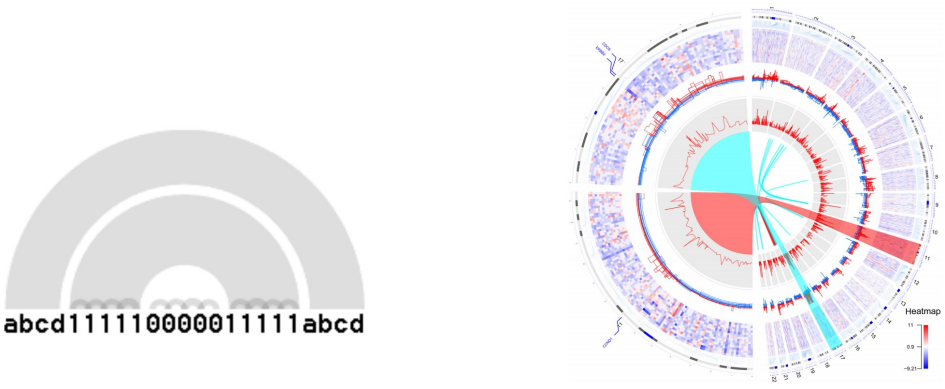
One domain that has a particular relevance in data visualization is genomics. Graphical genome maps are found to concatenate a significant amount of information in a very compact way. Genome features and sequence characteristics are assessed with this visual strategy. An example can be found on Figure 4.2b. 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 SSM. This data structure provides a significant range of information that can be used to retrieve structural information, such as block and periodic structures [110, 111, 117, 20]. This method will inspire our visualization strategy, which will be explained further.

4.1.8 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 [77].

Shape-based methods focus their attention in performing local comparisons between



(a) (b)

Figure 4.2: A - Diagram for string association. This image is taken from the works from [149]; 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.* [65].

time series. Examples of well-known methods are the Euclidean distance (ED) or Dynamic Time Warping (DTW) [85]. 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 [77, 85]. These distance measures are usually combined with a k-Nearest Neighbour (k-NN) classifier to 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 [130].

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 [130]. 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) [86] or the *Symbolic Fourier Approximation* (SFA) [131]. The first approach proposed for TSC with symbolic representations was the work of Jessica Lin *et. al* with the *Bag of Patterns* (*BoP*)[85]. 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 [130, 132].

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 [67]. 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* [157]. 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 [157].

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 [136].

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.

4.1.9 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 Symbolic Representation

4.3 Dictionnary-based Classification

4.4 Search by Query

5

Data Description and Management

The data used in this work focused mainly in the usage of biosignals. Some of the data used was searched with the intent of being related with data that can be acquired in occupational scenarios. This is to provide strong evidence that the methods developed, although applicable to any kind of time series, can be used for occupational health data as well. From public databases we used ECG, ABP, EMG, ACC and GYR. It is important to mention that for the sake of performance evaluation and comparison, several benchmark databases were used with a broad type of time series.

5.1 Dataset 1 - UCR Benchmark

Description

The University of 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[42]. 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 (audio, devices, ECG, Electrocularogram (EOG), EEG, Human Activity Recognition (HAR), etc...), which also are from many different domains, such as medical, financial, motion, entomology, etc...[42]. The list of datasets can be found on the following link [14].

Purpose

This dataset is the benchmark used in the context of time series classification to validate the proposed symbolic approach for this task.

5.2 Dataset 2 - Human Activity Recognition

Description

The dataset was found on Kaggle and comprises data from 15 subjects that were performing 7 activities while wearing a sole wearable ACC mounted on the chest. The data was

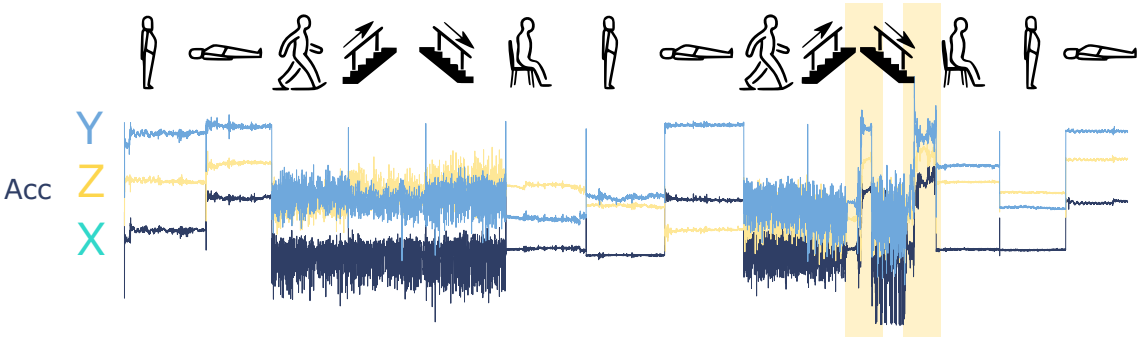


Figure 5.1: Example of the signal for this dataset. It shows the 3 axis of the accelerometer signal and the corresponding labels in each section [9, 8]. Yellow areas indicate moments where there was a change on the signal not related with the labeled activity.

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 [27].

Purpose

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

5.3 Dataset 3 - Smartphone Dataset for Human Activity Recognition in Ambient Assisted Living

Description

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 recorded the internal ACC and GYR data at a constant rate of 50 Hz. Each activity has been categorized and labeled on the acquired data [9, 8].

Purpose

This dataset was used in the context of novelty segmentation [9, 8].

5.4 Dataset 4 - Smartphone-Based Recognition of Human Activities and Postural Transitions

Description

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

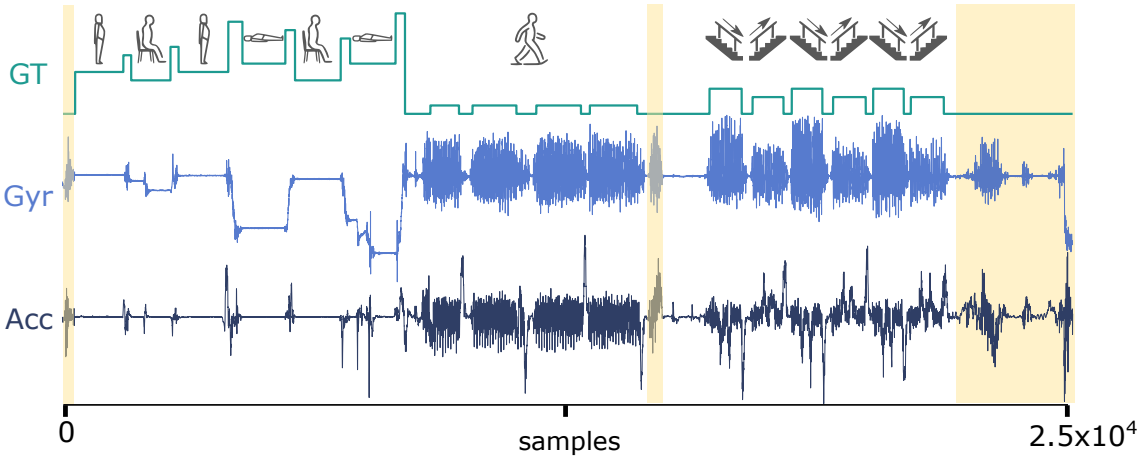


Figure 5.2: Example of the signal for this dataset. It shows one axis for the ACC and GYR signals and the corresponding labels in each section [119]. In this datasets, labels also highlight posture transitions, such as *Standing to Sitting*. Yellow areas indicate moments where there is activity but the signal was not labeled.

(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 [119].

Purpose

This dataset was used in the context of novelty segmentation [119].

5.5 Dataset 5 - Wireless Sensor Data Mining (WISDM) Smarphone and Smartwatch Activity Biometrics Dataset

Description

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 [152].

Purpose

This dataset was used in the context of novelty segmentation in complex real-life scenarios.

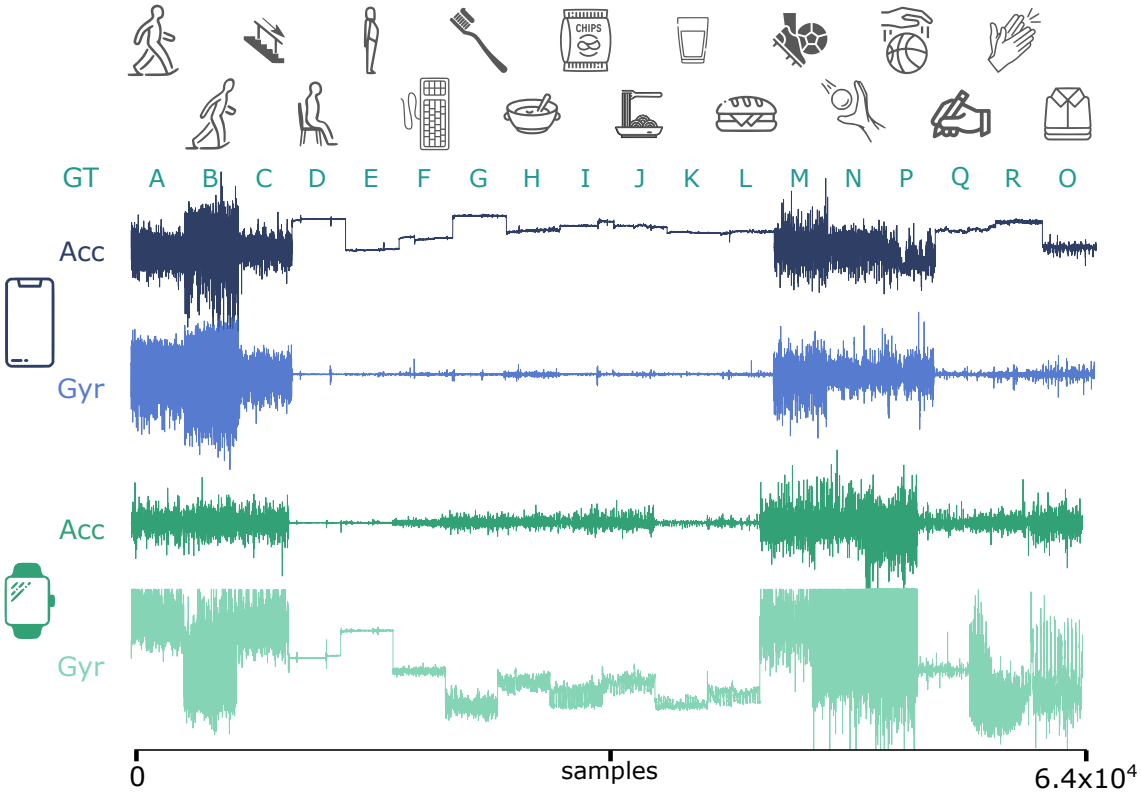


Figure 5.3: Example of the signal for this dataset. It shows one axis of each type of data acquired, for both smartphone and smartwatch. The activities are highlighted as well and labeled as: A - walking, B - running, C - walking on stairs, D - sitting, E - standing, F - typing, G - brushing, H - eating soup, I - eating chips, J - eating pasta, K - drinking, L - eating a sandwich, M - kicking, N - catching a ball, P - dribbling, Q - writing, R - clapping, O - iron [152].

5.6 Dataset 6 - EMG Data for Gestures

Description

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 [93].

Purpose

This dataset was used in the context of novelty segmentation, to test the method in estimating transitions between the activation (onset) and relaxation (offset) of the muscular activity.

As indicated in Figure 5.4, the ground truth would have considered as events the

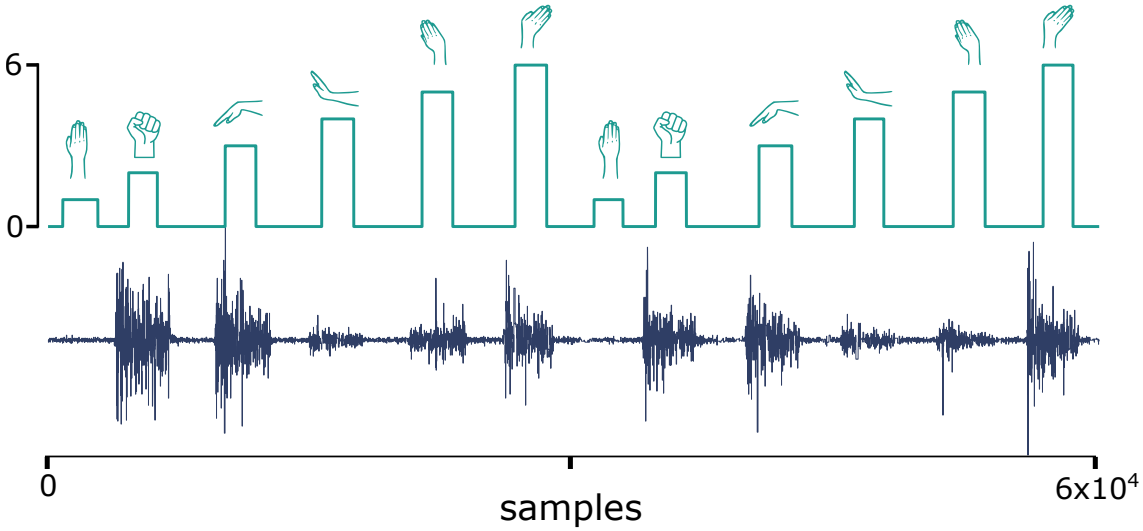


Figure 5.4: Example of the EMG data and the corresponding hand postures.

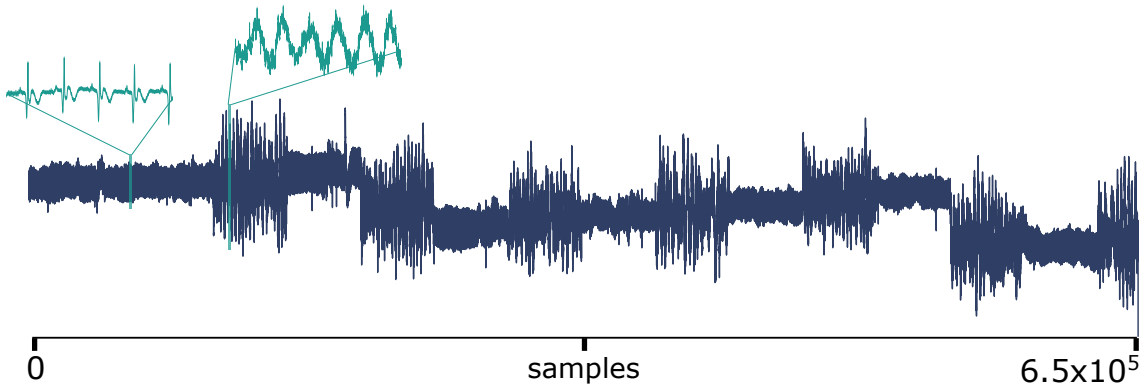


Figure 5.5: Example of an ECG record contaminated with noise. Two sections are highlighted showing the clean and contaminated area of the signal. [19]

absence of activity (which the original dataset uses because it was designed for a classification problem). In this case, we did not considered these events in our ground truth for this dataset for the problem of segmentation.

5.7 Dataset 7 - MIT-BIH Noise Stress Test Database

Description

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) [109, 55].

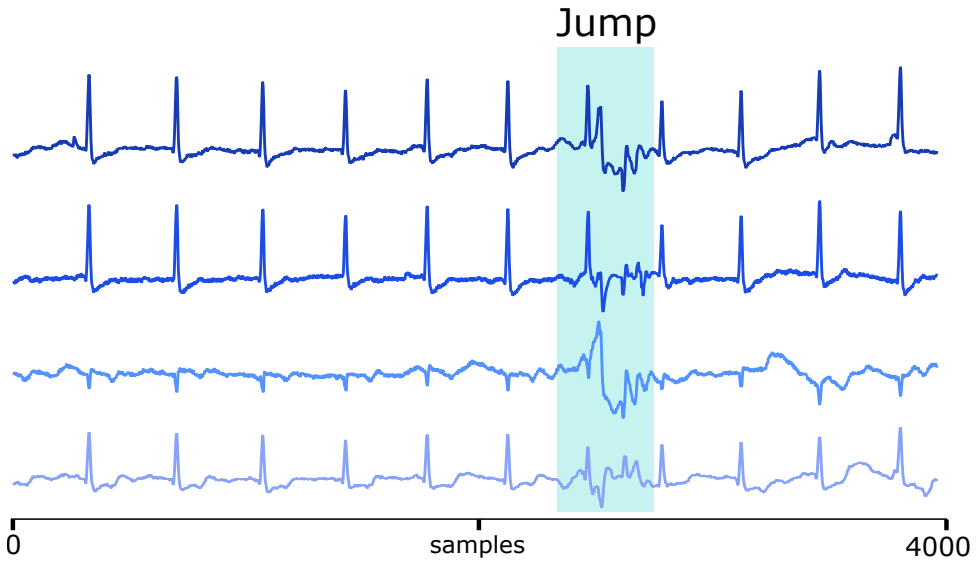


Figure 5.6: Example of an ECG signal contaminated with motion artifacts. The artifact is visible due to a jump from the subject.

Purpose

This dataset was used in the context of novelty segmentation, to test the method in estimating transitions to and from noise sections of the signal.

5.8 Dataset 8 - Motion Artifacted Contaminated ECG

Description

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 [19, 55].

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.

5.9 Dataset 9 - St Petersburg INCART 12-lead Arrhythmia Database

Description

This dataset has ECG signals from patients undergoing tests for coronary artery disease. 75 annotated recordings were extracted. Each record is 30 minutes long and has 12 leads, each sampled at 257 Hz. The annotations include beats as well as relevant artifacts/occurrences present on the signal, such as arrhythmia [55]. Records *I02, I04 and I09* were used.

Purpose
This dataset was used in the context of pattern search with QuoTS for the detection of different types of arrythmia.

5.10 Dataset 10 - Alan Turing CPD Benchmark

Description
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. The dataset comprises several time series from real-world data. It was built from a project of the Alan Turing Institute 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 [25].
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.

5.11 Dataset 11 - HCILab Behavioural Driving Dataset

Description
It is a public dataset that studied ways of assessing the driver's workload combining auto telemetry data with physiological sensors. The authors conducted a real world driving experiment with 10 participants measuring a variety of physiological data as well as a post-hoc video rating session [135]. It includes a variety of physiological signals, such as ECG and corresponding heart rate, Skin Conductance Response (SCR) and body temperature. In addition, it collected driving speed and GPS location. The map of the driving site can be seen at [135]
Purpose
This dataset was used for QuoTS in pattern search. For the example presented we used the data of participant number 10 [135].

5.12 Dataset 12 - CSL-Share Dataset

Description
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[11]. More specifically, the

recording was made for the work of [89], 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 ACC sensors, 4 sets of EMG sensors and an electrogoniometer. 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. This setup was used to perform 18 different activities, namely *Sit*, *Sit-to-Stand*, *Stand*, *Stand-to-Sit*, *Stair-Up*, *Stair-Down*, *Walk*, *Curve-Left-Step*, *Curve-Left-Spin*, *Curve-Right-Step*, *Curve-Right-Spin*, *Run*, *V-Cut-Left*, *V-Cut-Right*, *Lateral-Shuffle-Left*, *Lateral-Shuffle-Right*, *Jump-One-Leg*, *Jump-Two-Leg*. As a disclaimer, **lateral-shuffle-left/right** is 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 90°at jogging speed [87].

Purpose

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

5.13 Dataset 13 - EN3BLES

Description

The ENABL3S dataset contains bilateral EMG and motion data from inertial wearable sensors positioned on joints and limbs. Ten subjects participated in the recording, which involved freely transitioning between sitting, standing, and five walking-related activities. In total, 23 sensors were used (14 EMG, 4 Goniometer (GON), 5 IMU) and 52 channels (14 EMG, 8 GON, 30 IMU - 5 triaxial ACC + 5 triaxial GYR) [**enables**].

Purpose

The dataset was used for the segmentation of activities, made by finding the break-points on the multimodal time series.

Ground Truth

The labels of each activity were available for each dataset. The segmentation points used as ground truth are the transitions between labels.

5.14 Dataset 14 - Industrial Job Dataset

Description

During the period of this thesis, a private dataset was acquired at Volkswagen Autoeuropa in the context of this thesis and a master thesis from [**sara2019**]. The purpose was to test a wearable system capable of calculating the ergonomic risk through direct measures from inertial sensors. 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"[49]. For this work, the technology setting consisted on 4 9-Domain of Freedom (DoF) IMU sensors (composed internally by a triaxial ACC, triaxial GYR and triaxial Magnetometer (MAG)) and an Android wireless communication system. The last one was made through an application called Recorder, also developed by Fraunhofer AICOS.[**sara2019**].

The acquisition was made in a *Volkswagen Industrial* assembly line where 12 manufacturing workers performed their work tasks while having attached 4 IMU sensors in their upper body segments. 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 [**sara2019**]. The acquisitions made involved a total of 6 operators with each one performing at least 2 different workstations. The various acquisitions were simultaneously filmed, and to synchronize the ground manual annotations of the data, at 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). There were also registered some details regarding their anthropometric characteristics.

The mentioned study was centered on the ergonomic assessment of the dominant arm. For this reason, the IMUs 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. Additionally, a Smartphone was attached to the trunk of the workers as well, working as an additional IMU, from which the position of the arm in regards to the body posture was calculated. Each one of these 4 IMU devices had incorporated within them 3 triaxial sensors (ACC, GYR, MAG).

Purpose

This database was used to study the application of the algorithm to detect (1) Working Periods with the novelty function, (2) Periodic Working Cycles with the similarity function and (3) Search by example. The data was annotated based on video inspection.

Some important notes regarding exceptions found in sensors:

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

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

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

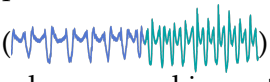
More details regarding the dataset and the acquisition process can be found at [**sara2019**, 128].

5.14.1 Notes for Ground Truth Annotations on Novelty Segmentation

Regarding the proposed method for novelty segmentation, the datasets used were mostly designed for classification tasks. Therefore, these were labeled with categorical values for each sample of the time series. As the problem of novelty segmentation requires the detection of specific samples of the signal, we adapted the labels of the datasets to fit the purpose of segmentation. This was made by only keeping the transitions between categories of labels, for example, in Figure 5.2, the ground-truth (GT) is categorical. For evaluation purposes, the ground-truth was converted to a binary signal, where categorical transitions were valued to **1**, to keep only the position of the change, and not the category. This is valid for segmentation purposes only.

6

Unveiling the *Grammar* of Time Series

Researchers are interested in understanding the structure of the recorded signals, the meaning behind them, and the influences of the context. In Chapter 1 (see Section 1.2) we gave several examples of how signals in our everyday life can be structured and partitioned, such as a signal representing a walking regime shifting to a jogging regime (). The examples we mentioned in Chapter 1 and above manifest the relevance and importance of the following topics:

- **Novelty segmentation**: to identify significant changes in the signal's behaviour.
- **Periodic segmentation**: to detect the presence of repeating cyclic patterns.
- **Labelling**: to measure how similar the segments are between each other with the help of similarity profiles.
- **Query search**: to find repeating patterns in the time series based on a *subsequence* given as an example.
- **Summarization**: to summarize visually the time series by partitioning it into relevant and uniform/periodic segments and joining segments based on their similarity.

In this Chapter, we present the proposed solutions to these five problems mentioned above, inspired by a method used for audio signal analysis and thumbnail generation

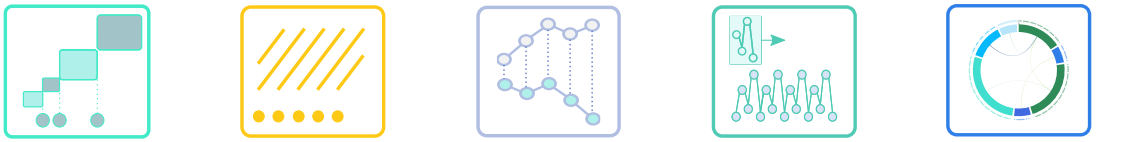


Figure 6.1: Information retrieval topics explained in this Chapter. Each of these topics is a result of analyzing the SSM.

[110, 117, 20, 22]. Such a method has not yet been extended to other types of *time series* domains that could greatly benefit from it [4]. The method uses a feature-based SSM of (multidimensional) time series, from which visual and analytical information is rendered to perform the segmentation process and associate *subsequences* of the time series with each other, which forwards automatic labelling and ultimately can be used to summarize the signal. We extended its application to *biosignals* with the usage of general features and introduced new methods to analyze the SSM for periodic segmentation, labelling, query search, and summarization.

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 and how are *subsequences* related, especially for tasks related to data labelling. The structure of a time series is built of *segments* delimited by *events*. The problem explored in this section is the search for such *events* and how to find the ones that are significant and useful to then perform a similarity search and summarization of the time series.

From the definition of *event* (see Chapter 2 Section 3.1), 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 Ranks

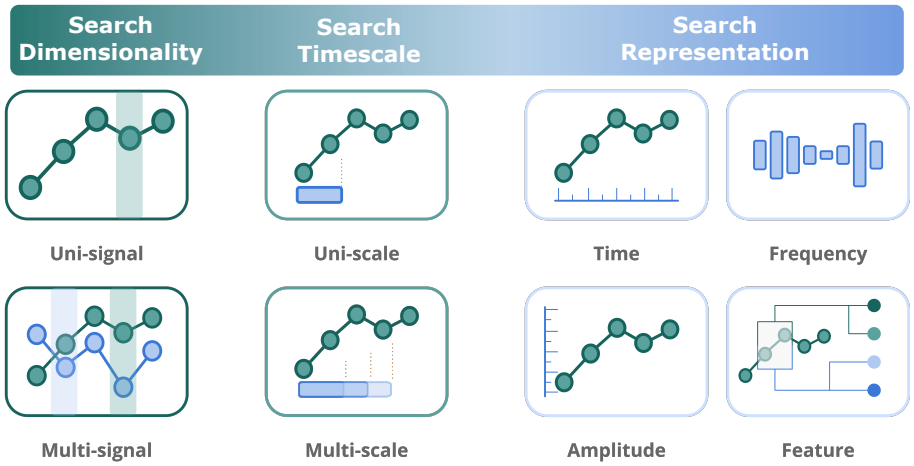
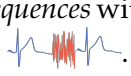


Figure 6.2: Event search in different ranks of dimensionality, timescales, and representation.

Figure 6.2 illustrates the search ranks of the problem formed by three layers:

- **Dimensionality**: The search can be applied to one or multiple time series. In multidimensional space, some events can coincide in several time series, while others are specified on a particular dimension. For example, some gestures produce noticeable signals on only one dimension of the three-axis accelerometer.
- **Timescale**: *Events*' occurrence can vary from different timescales. For example, when the signal being analyzed is zoomed in from hour to minute scales, some events may disappear while new events may be detected.
- **Representation**: The searchable objects can be straightforwardly the temporal nature of time series or other representation, such as frequency or other extracted features.

Besides the ranks mentioned above, the search procedure can be customized by context or target, which is highly related to the relevance given to an *event* or a *subsequence*. Types of events that are considered significant include:

- **Property change**: The change of a property, such as a change in mean or frequency, or a set of properties is greater than a threshold, e.g., low to high frequency -  .
- **Peak/Valley**: Peaks and valleys can typically be associated with significant physical changes, e.g., ECG peaks like  .
- **Periodicity**: The starting points of each period in a periodic signal are considered relevant, e.g., ABP periods like  .
- **Recurrent pattern**: Re-occurrences of similar *subsequences* with specific patterns should be of interest. Unlike *periodicity*, *recurrent* patterns do not have a temporal regularity.
- **Anomaly**: Highly dissimilar *subsequences* with particular patterns are of the reference value, e.g., noise in a clean signal  .

6.1.2 Proposal

In order to fill as many research gaps as possible, this study started by defining the search space for the aforementioned topics, considering that if the time series is transformed in the feature space, we can have a multidimensional similarity analysis based on the features' behaviour. The notion of *difference* in time series can be associated with *distance/similarity*, enabling finding segmentation points, recurrent patterns, anomalies, and periodic shapes, which is ideal to perform all the approaches listed above. This feature-based multidimensional representation is already successfully used for time series' classification purposes [**tsfel**] and is potentially applicable to all the mentioned topics. For instance, any feature's change would be relevant for novelty segmentation, such that changes in the mean,

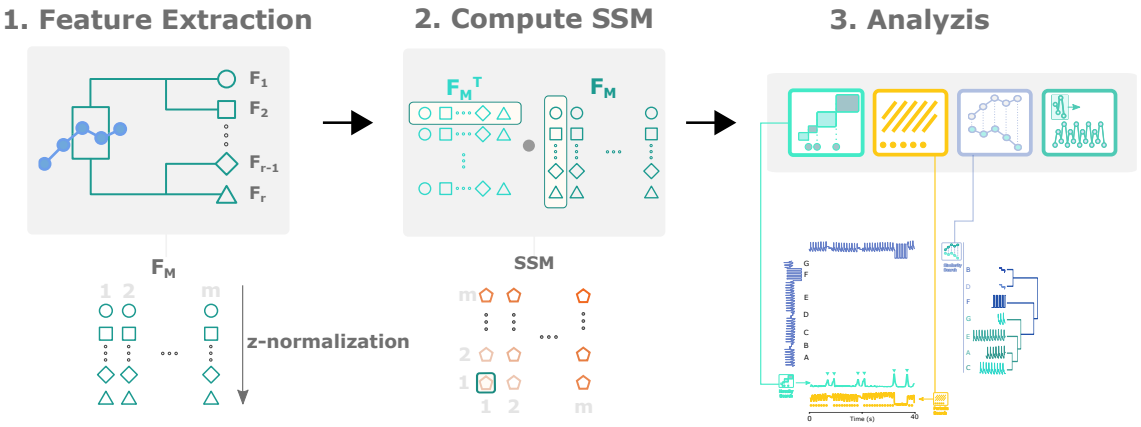


Figure 6.3: A step-by-step flowchart for calculating and analyzing the SSM. The signal-based calculation requires input parameters of the window size w and the overlapping percentage o to fulfill the first-stage feature extraction. Features are extracted on each subsequence $(sT_1, sT_2..., sT_N)$, where N is the total number of windows. K features are extracted from window i (sT_i: $f_{i_1}, f_{i_2}, ..., f_{i_K}$). Different features are associated with different shapes (◯, □, ◇, and △) in the figures. The features can be extracted on an M-variable record and each feature is positioned as a row on the F_M for the SSM computation.

standard deviation, frequency, or other properties are all options worth searching for. By characterizing the signal in the feature space, we can explore changes in all feature representations.

We propose an unsupervised methodology that searches for events (1) in uni and multidimensional space, (2) with a fixed timescale and potential multi-timescale application opportunities, and (3) on an SSM computed by a feature space representation of the time series. The events to be searched are any changes in the SSM related to a segmentation point and/or a periodic event.

The proposed method's reliability for event detection will be evidenced by considerable experiments in various type-agnostic databanks of multiple time series domains and comparisons to state-of-the-art methods. It should be highlighted that events in different datasets are extracted from the same information source, i.e., SSM, which meanwhile provides insights into unsupervised automatic labelling.

6.2 Building the Self-Similarity Matrix

In this section, we explain the steps of the proposed method. The extraction of relevant events from time series starts by computing the SSM. As explained in Section 3.6.2, this matrix has relevant structural information to retrieve *events*, namely *blocks*, *paths* and *similarity profiles*. Figure 6.3 summarizes the steps involved in calculating the SSM.

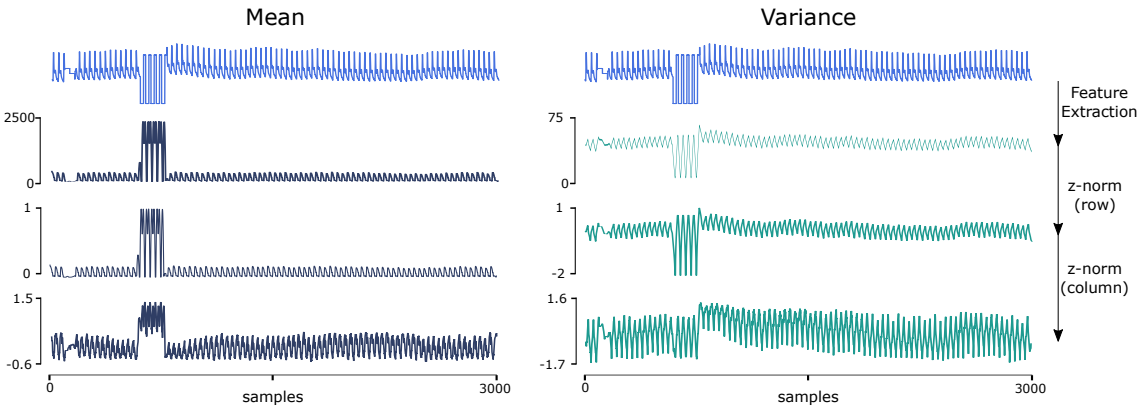


Figure 6.4: Example of feature vectors before and after normalization. Mean and Variance features are presented for the ABP signal from Dataset **??**.

6.2.1 Feature Extraction

The structural information on the SSM reflects how informative the feature set can translate the signal's changes and disruptions. Behavioural changes may be related to a variate set of features. As a feature can be sensitive to a particular type of change, the set of features should be diverse to identify a multivariate set of events and be agnostic to various signal types. We turned to the available features from the *Time Series Feature Extraction Library* (TSFEL) [**tsfel**] for Python, which has been proven effective and efficient in our previous work on HAR and multimodal biosignal processing [**liu22pipeline**, **liu2021thesis**]. We selected over 50% of all TSFEL features in the statistical, temporal, and frequency domains with relatively lower computational costs, as listed in Table A.1 in Appendix A, regarding our proposed method's high calculation resource consumption.

The features are extracted with a moving window with size w, specified by the user, with an overlap percentage o. The selection of the two sizes significantly influences the results: w defines the timescale at which features are extracted so that the wider the window, the more *zoomed-out* the search will be. The second parameter defines the pixel resolution of the resulting feature series, increasing the amount of information with a larger overlap.

The extracted features are grouped into a feature matrix (F_M), where the rows represent a feature series and the columns correspond to all subsequences. In the multidimensional case, r features extracted from each of the k dimensions are ordered in the F_M as rows, forming $r \times k$ elements in each row, as illustrated in Figure 6.3.

Each feature series (rows of the F_M) is z-normalized for a more balanced contribution to characterizing the signal. A further normalization is applied to the feature vector (columns of the F_M), which optimizes the cosine similarity computation between feature vectors by simply adding the dot product to calculate the SSM. As an example of two simple features (mean and variance) and their normalized versions, we show Figure 6.4.

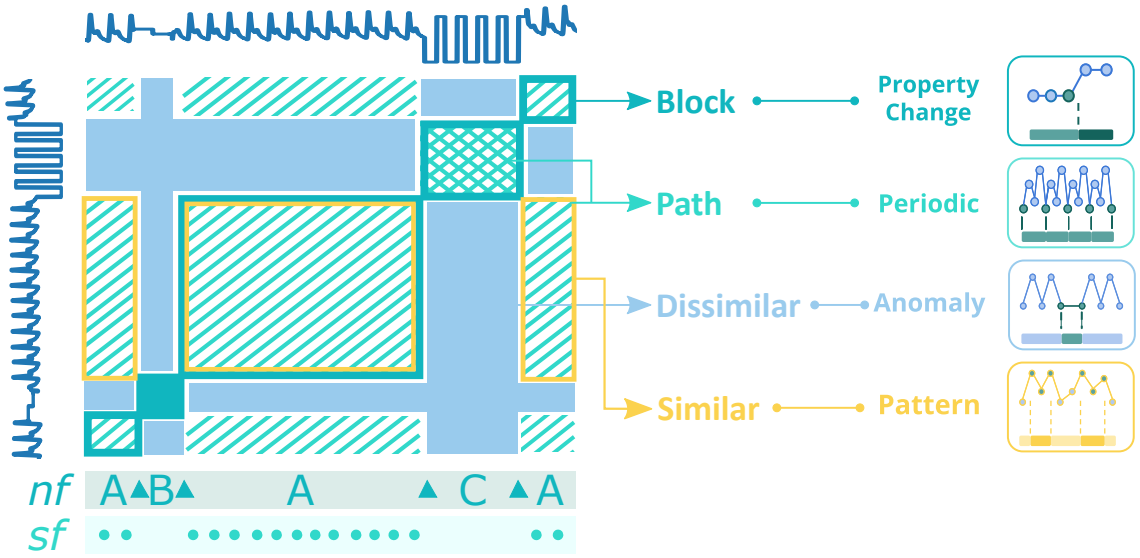


Figure 6.5: The informative structures of an ABP signal's SSM. The three main structures are highlighted in the simplified illustration: A - the homogeneous segments corresponding to periods in the ABP signal; B - the homogeneous segment representing missing data; C - the homogeneous segment cueing sensor detachment. The "blocks" in the figure accentuate homogeneous behaviour while the paths in the figure depicts periodicity in the segment. Segment C has a cross pattern, which symbolizes periodicity and symmetry. nf: *novelty function*; sf: *similarity function*; $\triangle$: change points separating blocks A, B and C.

6.2.2 Feature-based SSM

After grouping all the features extracted, the next stage is to apply a similarity measure to the feature space and compute the SSM. This process consists in comparing each *subsequence* with all the other *subsequences*. Since each column of the F_M is each subsequence's feature characterization in the entire feature set, the SSM, i.e., the comparison between segments, is obtained by calculating the dot product between the z-normalized transposed F_M and itself:

$$\text{SSM} = F_M^T \cdot F_M. \tag{6.1}$$

The dot product scores the similarity based on the subsequence's feature values. Cells of the SSM with higher similarity scores indicate that the corresponding *subsequences* have similar feature values [117, 20]. As a result, the SSM provides rich visual information, highlighting structures that describe the signal's morphological behaviour over time and structure, such as blocks and paths.

In Figure 6.5, the main structures are illustrated and highlighted in an example of an *SSM* [117] computed from an ABP signal, where the main structures are *blocks* and *paths*. Our proposed method utilizes the resulting main structures to extract the desired information. *Paths* show recurrence of patterns, which indicates the morphological matching between corresponding *subsequences*. Circles in the *sf* layer exhibit when the

paths start. The *cross-pattern* in *block* C means that the *subsequences* are periodic and symmetric.

Differently, *blocks* are square-shaped structures of homogeneous areas in the SSM, translated as constant behaviour in the time series. The change between block structures along the main diagonal displays a relevant change in morphology and behaviour in the time series. In Figure 6.5, the SSM is segmented into several blocks on layer *nf*, for which the $\triangle$s mark the change points that separate blocks A, B and C. Besides *paths* and *blocks*, the SSM provides similarity measures between *subsequences*, which can be used to spotlight (dis)similar segments, such as anomalies, motifs or cycles. Several strategies were applied to the SSM to extract the mentioned information.

6.3 Information Retrieval

The SSM is a powerful visual tool *per se*, exposing relevant information that a raw observation could miss. Automatic discovery of information of interest will increase the SSM's practicability and versatility, for which three approaches for information retrieval on the SSM are put forward: (1) novelty search of *block* transitions, (2) periodic pattern search of *paths*, (3) similarity profiles of *subsequences*, and (4) how to use a query from the SSM to search for specific *subsequences*.

6.3.1 Novelty Search

The search for *novelty* is inspired by a method used in musical structure analysis [46], which is computed with the help of the *libfmp Python* package [**libfmp**]. The process involves searching for transitions between *blocks* using a moving chequerboard square matrix, resulting in a one-dimensional function: the *novelty function*.

As shown in Figure 6.6, *block* transitions along the diagonal are represented by a chequerboard pattern. Such patterns can be detected by correlating a standard chequerboard matrix with the diagonal of the SSM, for which a sliding squared matrix, designated *kernel*, is used. The kernel incorporates a Gaussian function with a smoothing factor. The kernel K_N combines two different square matrices: K_H and K_C. K_H is responsible for identifying the homogeneity of the SSM on each side of the centre – the more homogeneous the pattern is, the higher the corresponding values will be. K_C measures the cross-similarity level. Therefore, when sliding the kernel K_N along the diagonal, a higher correlation value will be returned when it reaches a segment of the SSM with a similar chequerboard pattern. The result is the mentioned novelty function [41, 110, 111].

In position **A** of Figure 6.6(right), due to the high homogeneity, the kernel returns a value approaching 0 when summing the product between it and the section of the SSM it overlaps. In contrast, the kernel in position **B** reaches a segment with low cross-similarity and high diagonal similarity, which results in high correlation values with a chequerboard

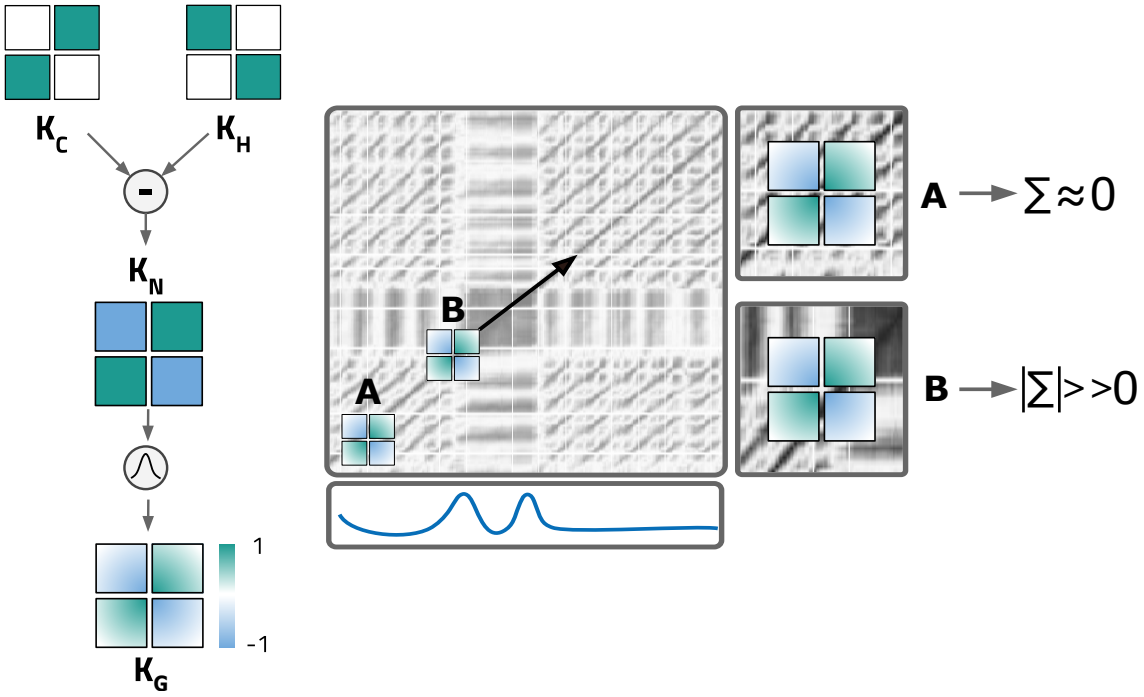


Figure 6.6: Left: description of the matrix (kernel) used to compute the *novelty function*, based on the works of *Mueller et. al* [110, 111]. The chequerboard pattern of the kernel K_N is achieved by combining the kernel K_H (homogeneity measure) and K_C (cross-similarity measure). Combined with a Gaussian function, the K_G is obtained; right: the process to compute the novelty function based on the works of [41, 110, 111]. Kernel K_G slides along the diagonal of the SSM to compute the *novelty function* presented as the bottom sub-plot. Positions A and B point to the effect of block transitions on the *novelty function*.

pattern. Therefore, high *novelty function* values are witnessed in these transition segments [41, 110, 111].

Each section of the kernel has the same size, $L \in \mathbb{N}$, and $D = 2 \times L + 1$ configures the total kernel size. The kernel has an odd size to adapt zero values in centred points, and a total size of $D \times D$. K_N is defined by [110, 111]:

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

where $a, b \in [-L : L]$ and $sign$ represents the sign function (1, 0 or -1). A radially symmetric Gaussian function is used to smooth the Kernel [110, 111]:

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

where σ is 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 kernel K_G is computed by point-wise multiplication with the Gaussian function:

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

The *novelty function* n_f is calculated by correlating the kernel with the diagonal of the matrix:

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

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 the *novelty function*, 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 SSM. The *path*'s start point punches where the period of the pattern begins. In order to find the periodicity, we compute the similarity function s_f by summing the values of the symmetric SSM column-wise or, equally, row-wise. Each element of the sf is calculated by

$$sf(x) = \sum_{i=0}^{m} \mathrm{SSM}_{ix}, \tag{6.6}$$

where i is the column position for the sum, sf_j is the sample of the function at position j and m is the feature-series size. As segments with similar morphology will be closely described by the extracted features, the columns will have an approximate representation, resulting in similar values on the *sf*. The similarity function will enhance such behaviour when facing periodic series. The identification of events related to the periodicity of a time series is then feasible by searching for local minima (valleys) on the similarity function.

Although not validated in this work, an additional application of the similarity function should be outlooked. Considering that each sample of the sf is an average similarity of a subsequence to all other subsequences, it is possible to find *anomalies*. Regarding an *anomaly* as a subsequence highly unique and different from all the rest of the time series, its average similarity to all the other subsequences should have a low value.

6.3.3 Similarity Profiles

The principal elements, *blocks* and *paths*, are the information basis for segmenting the time series. Besides, SSM also provides pairwise similarity values between all *subsequences* of the time series, an important measure that can be used for clustering and *motifs*/*discords* discovery. The similarity profiles exploit the similarity values of the SSM to facilitate *subsequences* comparison. A similarity profile charts the similarity values of a *subsequence* (one column/row of the SSM) to all the other *subsequences*. The higher the values, the more similar the *subsequences*. In addition to the *subsequences* comparison, the similarity profile can also compare between entire segments of the signal. Consider, for instance, all

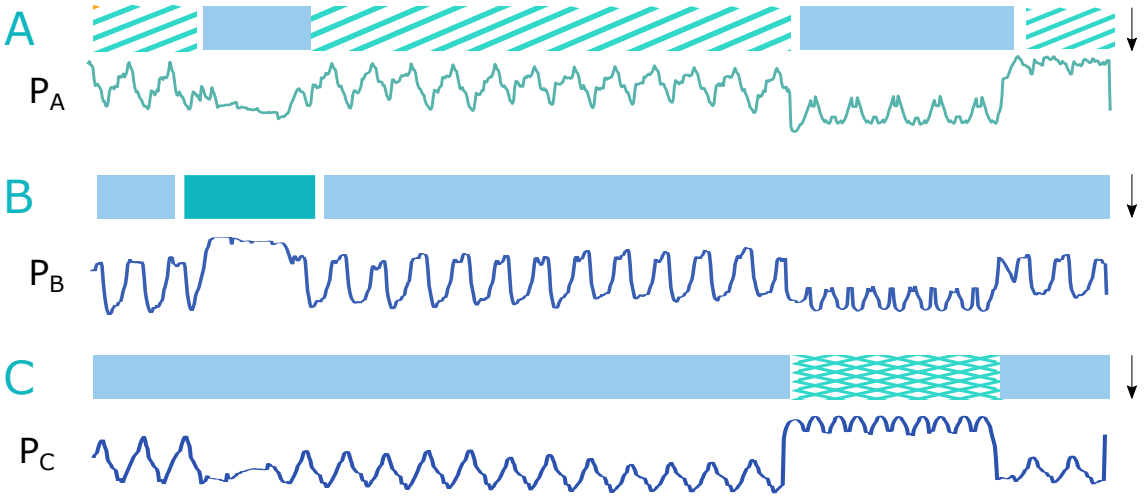


Figure 6.7: Profiles computed for each segment of the example signal used in Figure 6.5.

three *A*-segments highlighted in Figure 6.5, whose profiles are highly similar despite the different sizes.

Although the segment comparison could be directly based on the region of the SSM delimited by two *subsequences*, we propose a more effective measure of two segments' similarity/difference according to their similarities/differences to all the other *subsequences*. A *similarity profile* $P_s(c)$ of a segment is computed as the column(row)-wise average similarity values of the region delimited by the segment being profiled (size l), and all the other *subsequences* of the time series (size m):

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

The *similarity profile* is computed column(row)-wise. Each column(row) $c(r)$ is the average similarity value between the reference segment and the segment corresponding to c. The reasoning is that similar segments should have closer *similarity profiles*. Since the profiles have the same size, they can be compared by certain distance measures, such as the ED, to form clusters, an automatic clustering solution based on the segments generated by the *novelty* and *similarity* functions.

A general example of applying this process to the segmented time series based on the *novelty function* is shown in Figure **??**. Each segment category (A, B and C) extracted from the SSM of Figure 6.5 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

The mentioned similarity profiles are also useful to search for specific repetitions of a query from the time series. The process follows the traditional methods of template-based

search methods explained in Chapter 3.6.3, but instead of doing the process directly on the time series, it is performed on the SSM. The search procedure works by sliding the smaller column window (the example selected on the time series) along the SSM, one sample at a time. The distance, D, between the example and the segment it slides over is calculated through the sum of absolute differences:

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

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

6.4 Illustrative Evaluation in Various Application Scenarios

Experiments from multiple domains were carried on to validate the practicability and universality of the process to represent the time series into a feature-based SSM and the methods of retrieving information from the SSM.

6.4.1 Acceleration Signals in Human Activity Domain

Figure 8.17(top) exemplifies the SSM's usage on a record of an HAR dataset (see Section **??**), where the data of all three accelerometer axes is applied. The SSM was computed using a 250-sample window size, and a 95% overlap. Along the diagonal, the novelty function generates block-wise references for estimating activity transition using a 45-sample kernel.

We can identify in Figure 8.17(top) that the detected segmentation points match the activity transitions. Although all transitions are visible on the novelty function, the transitions between similar activity patterns in the walking category (straightforward, upstairs, and downstairs) are more challenging to differentiate, as block *A* suggests, which is plausible since the properties of these segments are morphologically similar.

The proposed unsupervised method automatically and sensitively detects any significant change in properties. As can be found in the yellow-marked part in Figure 8.17(top), the period in which the subject was performing the *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.

When *zooming* the SSM into segment *A* in Figure 8.17(top), the three activities in the walking category can be effortlessly segmented based on the change points revealed by chequerboard patterns, as the two most prominent peaks in he corresponding novelty function pinpoint (see Figure 8.17(bottom left)). In addition, it is noticeable that the matrix segments related to *Upstairs/Downstairs* are also segmentable into smaller *blocks*. As the information is not available in the dataset description, we believe they are a flight of stairs.

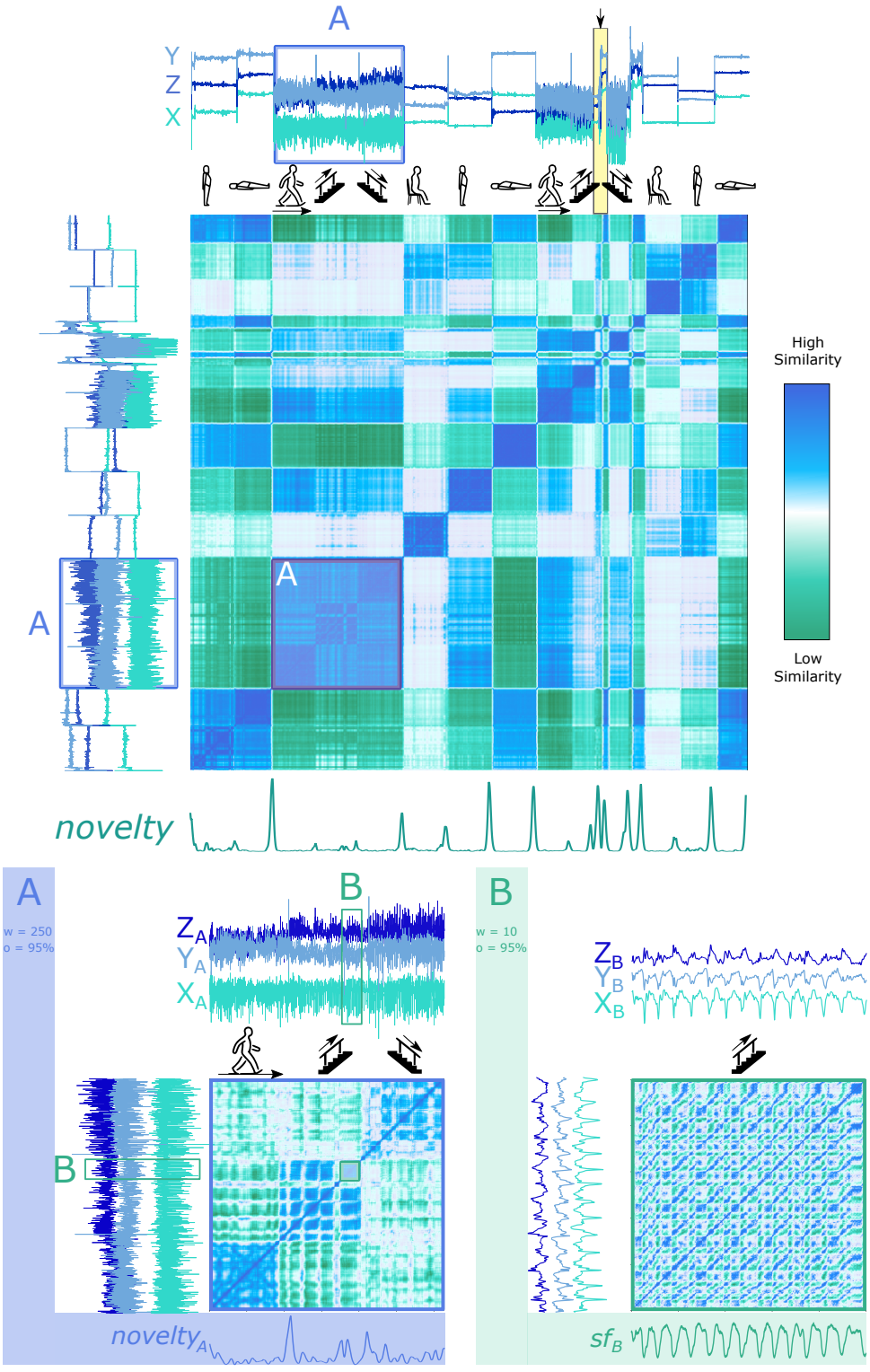


Figure 6.8: An SSM-based novelty search strategy to detect segmentation events on a signal piece from Dataset 1 (see Section **??**). Top: $window_{size}$=250 samples, $kernel_{size}$=45 samples, and overlap=95% on the activity sequence Standing → Laying → Walking → Upstairs → Downstairs → Sitting → Standing → Laying → Walking → Upstairs → Downstairs → Sitting → Standing → Laying; bottom left: signal change point detection on segment A with a size of 5000 samples, an overlap of 75%, and a kernel size of 25 samples; bottom right: further zooming in with a window size of 10 samples and an overlap of 95%, to reveal more periodic details of segment B.

Questions may arise at this point. Why is the signal periodicity of the three walking activities not evident in Figure 8.17(bottom left)? The reason is that the window size used to compute the SSM is relatively large. If features are extracted with a smaller window size closer to the walking period, the *paths* delineating the pattern recurrence will be visible. Figure 8.17(bottom right) shows the SSM built from segment B of the original time series, with a window size of 10 samples and an overlap of 95%. The *paths* in the matrix enable the periodicity detection with the similarity function sf_B.

6.4.2 Arterial Blood Pressure (ABP) Signals in Posture Recognition domain

Many biomedical signals, such as ECG, ABP, and Respiratory Inductance Pletismography (RESP), contain retrievable structural information like periodicities. Meanwhile, unexpected changes may occur during the acquisition due to physiological responses, medical disorders, or sensing problems like noise, interferences, artefacts, and electrode detachment. We visualize two examples of physiological changes in different types of periodic signals.

The ABP signal can vary due to postural changes, as an available experiment at *Physionet* confirms [61, 56]. Figure 8.18(top) shows the process of segmenting the ABP signal based on postural changes, where the trapezoidal and the square wave tag the ground truth of slow and fast postural transitions. The proposed strategy well perceives the change points. Observably, the shape of the raw ABP signal in each regime is undistinguishable through the naked eye. Therefore, it is constructive to rely solely on the signal itself to implement postural change detection. It is important to point out that the periodicity of the signal is not visible on the matrix because the features were extracted with a window size of 5,000 samples, which is much larger than the period size. A smaller window size of 250 samples in the current scenario allows periodic segmentation, as Figure 8.18(bottom) illustrates, where the SSM is computed on the first 10,000 samples of Dataset 5 (see Section **??**). The resulting *similarity* function gives prominence to the periodic nature of the ABP signal.

The SSM of Figure 8.18(top) also shows which segments are similar to each other. The blue-coloured parts in the matrix indicate high similarity, presenting that segments from the same posture are more similar than between different postures. For further illustrative purposes, we computed the similarity profiles of each segment as if segmented by the *novelty* function, which evidences that the corresponding sections could be well clustered based on the similarity profiles ($P_A = P_C$ and $P_B = P_D$). In the same way, the similarity profiles in Figure 8.18(bottom) examine the similarity between segmented *subsequences*. Profiles with a similar shape can be grouped ($P_A = P_C = P_E = P_G$ and $P_B = P_D$), which can be applied to automatic clustering, as will be exhibited further in the next Section.

6.4.3 Electrocardiography (ECG) Signals in Biomedical Domain

Another widely used biomedical signal, ECG, also testifies to the feasibility of our proposed method. The ECG signal in Figure 6.10(left) displays the presence of the condition called

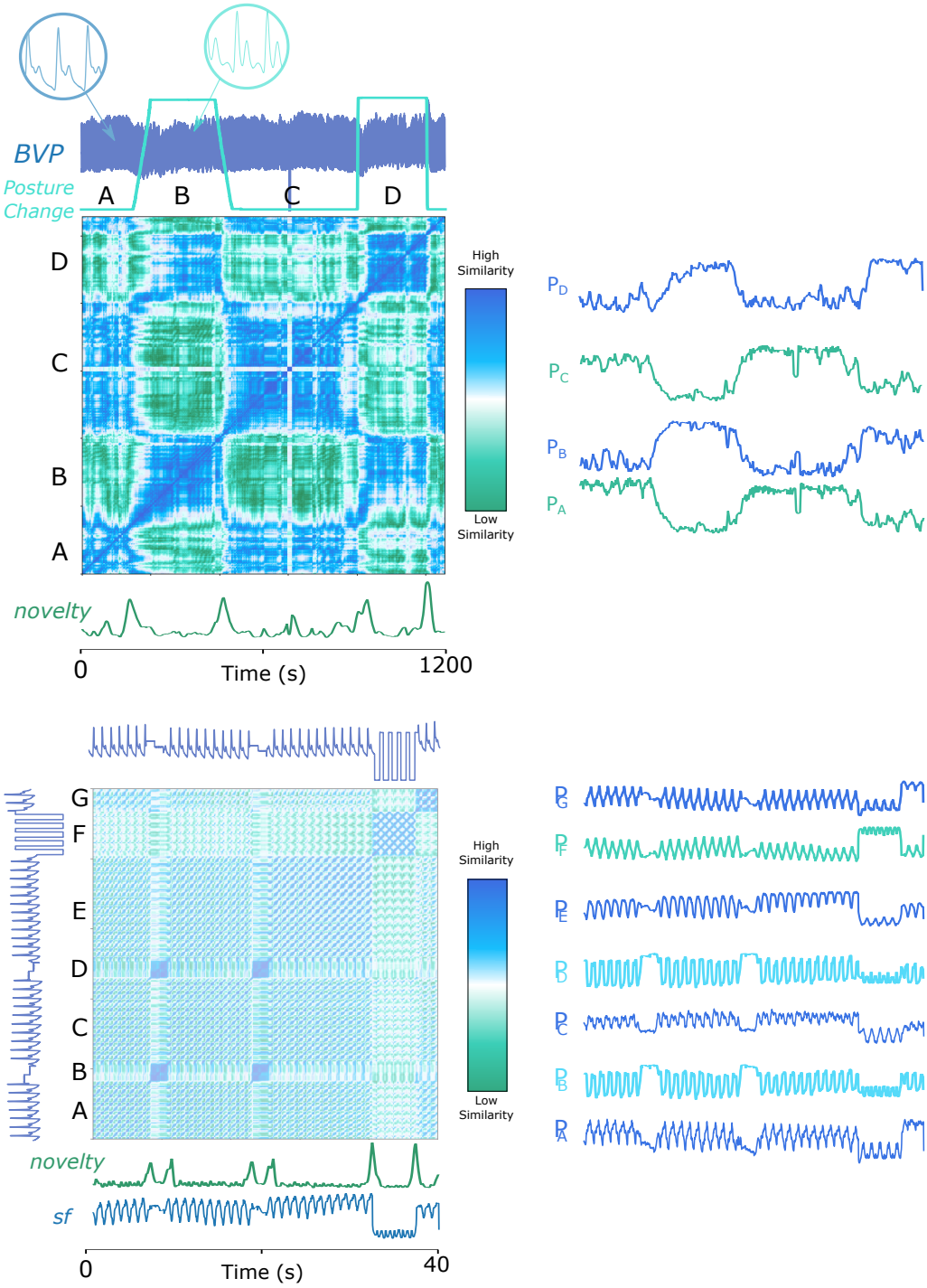


Figure 6.9: Novelty search on an ABP signal from Dataset 5 (see Section **??**). Top: a window size of 5,000 samples, an overlap of 95%, and a kernel size of 200 samples. The trapezoidal and the square wave mark the ground truth of slow and fast postural transitions. Bottom: the first 10,000 samples, with a window size of 250 samples, an overlap of 95%, and a kernel size of 200 samples. The right parts of the top and bottom subfigures plot the corresponding similarity profiles for each subsequence segmented by the novelty function.

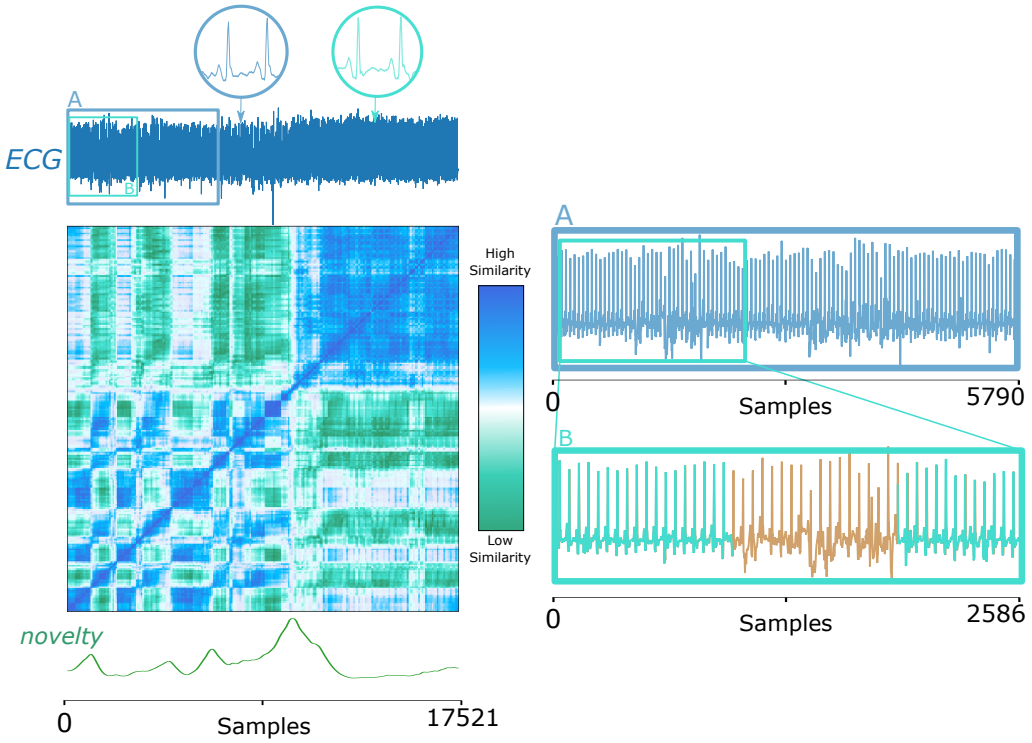


Figure 6.10: An ECG signal with a *pulsus paradoxus* condition starting at the $10,000^{th}$ sample from Dataset 6 (see Section **??**). Left: the SSM diagnoses two modes in the signal, whose patterns are zoomed in the circle thumbnails, respectively; right: zooming parts of the original signal can verify SSM's ability of automatic ECG pattern change detection and contribution to segmentation.

pulsus paradoxus, an exaggerated fall (>10 mmHg) in the subject's blood pressure during inspiration [108], which can also occur when the patient changes sleeping posture after heart surgery[53], as the following example relates. Similar to the ABP signal elucidated in Section **??**, the human eye hardly perceives the change points in ECG signals. Once again, our proposed strategy shows strength.

In addition to the novelty detection, segment A previous to the pulsus paradoxus occurrence can be partially detailed again to reveal minor changes due to additional noise, verifying SSM's sensitivity to structural changes, as Figure 6.10 (right) imparts.

6.4.4 Single Channel versus Multidimensionality Application in Multi-Sensor Scenarios

The proposed method accepts both single- and multidimensional records. The difference regards the number of features extracted. As compared in Figure 6.11, the same set of features is extracted from each time series to build the F_M. Using a single or several time series of a multidimensional record is an option, depending on the purpose. In some cases, the use of non-complete dimensions may miss relevant events, as Figure 6.11 instances the record "Occupancy" from Dataset 7 (see Section **??**).

The record is a multidimensional time series that measures room occupancy based

on temperature, humidity, light, and CO_2. By comparing the signals and formations in the left and the right parts of Figure 6.11, it can be understood that some events can be detected using the CO_2 series exclusively, but some are missed.

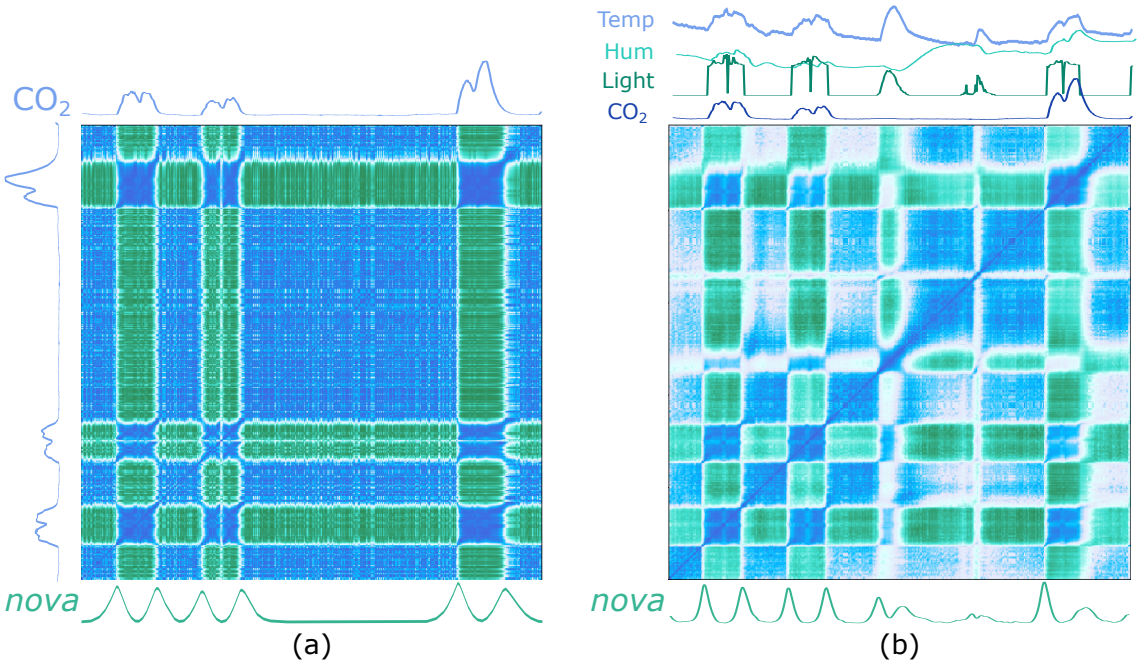


Figure 6.11: The proposed method applied on the *Occupancy* record of Dataset 7 (see Section **??**). Left: calculations on the separate CO_2 time series only; right: calculations performed by extracting features on the complete four time series.

6.5 Time Series Summarization

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 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 can 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 by analyzing their similarity. In addition, a characterization of the segments is possible with 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.

6.5.2 Compact Design

Strategies that are typically used to present information compactly are found in several domains. In text analysis, for instance, the relationship between repeating sequences is illustrated with arc diagrams [80, 149]. 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. For instance, 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 in Figure 4.2b (see Section **??**). This type of visualization inspired the summarization approach proposed for time series.

The proposed visualization strategy has several elements that can be used to transform the SSM into a compact form of filtered information. The elements are illustrated in Figure 6.12 and listed as follows:

- Novelty N and Subsegment Layers P - The novelty layer N has the segments resulting from the novelty segmentation step, color-coded based on how similar each *subsequence* is with the first segment. The subsegment layer P partitions each novelty segment into single periods, when the *subsequence* is periodic;
- *Chords* C - Each novelty segment is connected to the most similar subsegment with a colored connector;
- Signal Layer - The original signal is located on top of the partitioned segments.

6.5.3 A step-by-step example

The steps are indicated in Figure 6.12 for the ABP signal example. After computing the SSM, 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 6.13, 6.14 and 6.15 show the step-by-step process to summarize the time series by analyzing the SSM.

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

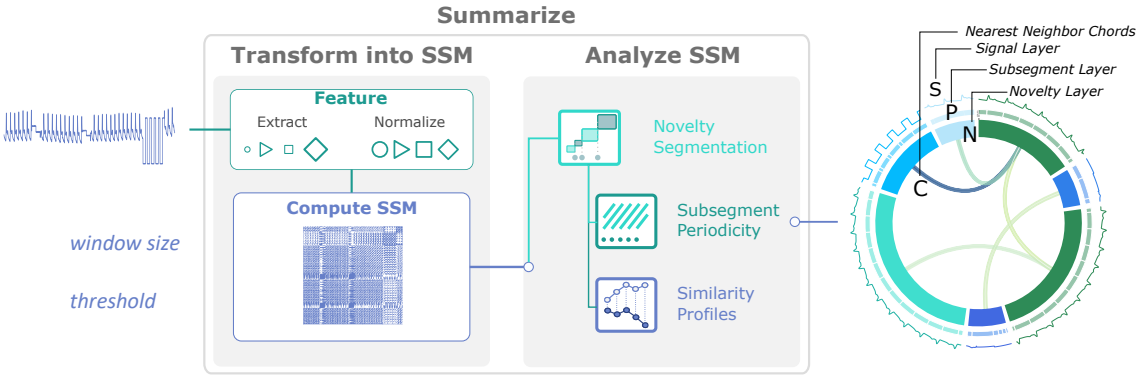


Figure 6.12: Steps to use the aforementioned methods on the SSM to summarize the time series into a circular plot with C - chords that connect nearest neighbors, S - the signal layer, P - the subseqment layer and N - the color coded novelty layer. Each of these elements of the summarized plots are built based on novelty segmentation, subsegment periodicity check and similarity profiles.

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 6.13.

Further, the segments are compared with the *similarity profiles* of each segment. Figure 6.14 illustrates as example the rows of the SSM 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 dendrogram of Figure 6.14. The dendrogram shows that

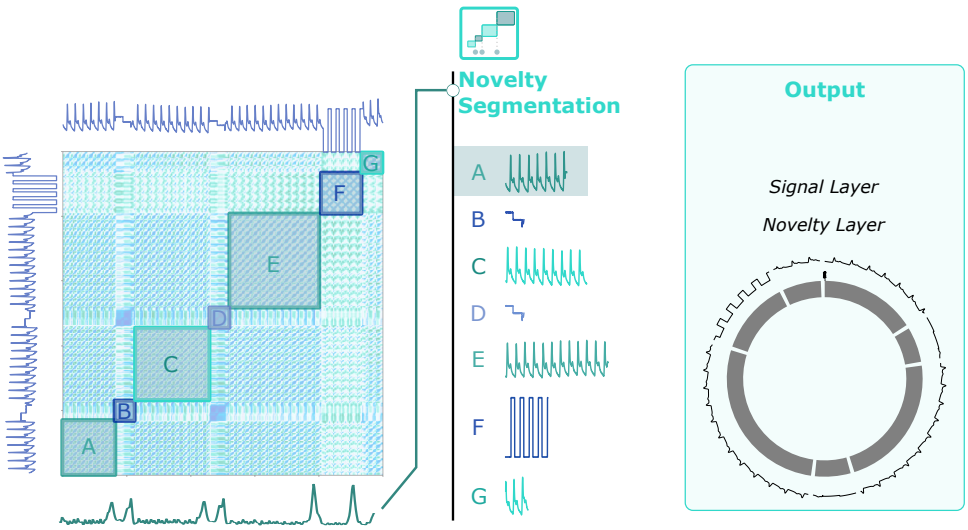


Figure 6.13: How to reach a standard format. This step applies the novelty search method to find the segmentation points. The signal is broken into A to G segments.

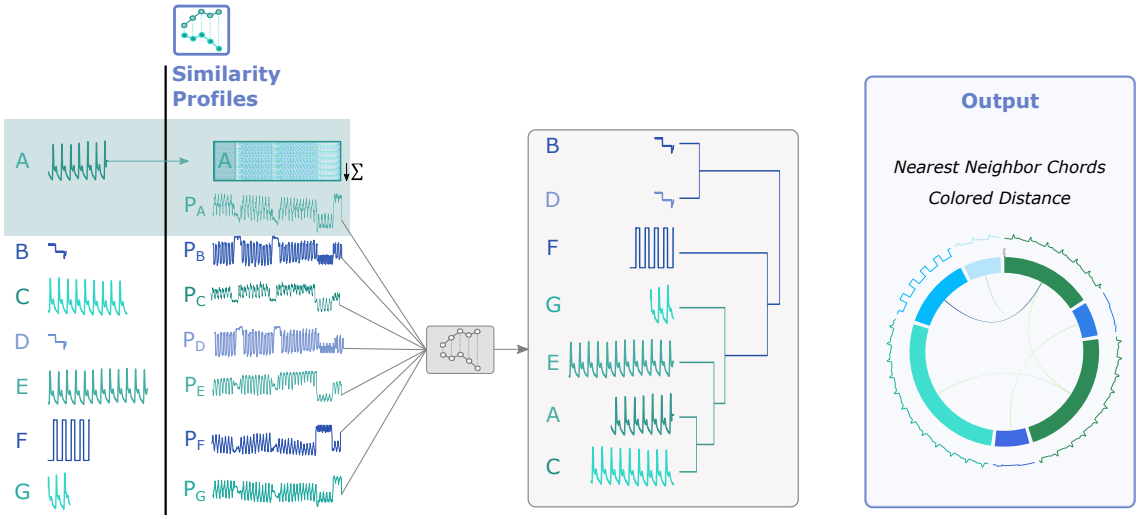


Figure 6.14: From each segment, a similarity profile can be computed. The pairwise euclidean distance is applied to these profiles, from which a dendogram is created. From this association, layer C can be drawn and the colors of the novelty layer can be added. Segment A is highlighted for the next step.

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 ED or DTW, would not be able to directly sort these segments correctly. Using *similarity profiles* is more robust and invariant to the size and time distortions. The illustrated dendogram also has distance measures between the novelty segments. We use the distance from each segment to the first segment (A) and select the corresponding color-code to be represented on the summarized layout.

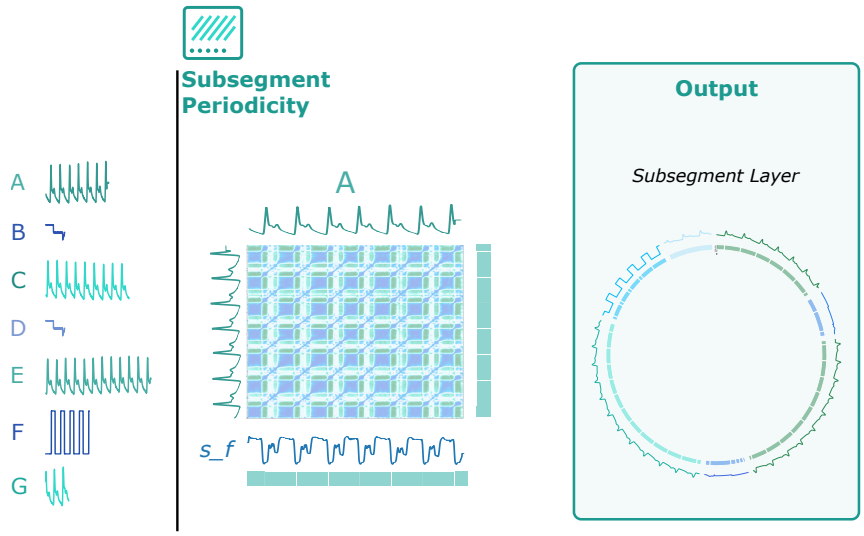


Figure 6.15: After finding the first layer of the segmentation points, the periodicity of each segment can be checked and added as a sublayer of segments, adding layer P. In this case, segment A is the example.

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 6.15 illustrates the *periodic search*, where periods are segmented by the minima of the s_f.

This summarization process is mostly graphical and follows the results of each process for information retrieval from the SSM. Therefore, this method will not present overall results or be validated in Chapter 8.

7

Language for Time Series Data Mining

Human language is foremost a means of communication, in which the information is represented by sentences, composed of words that can then be broken into sequences of symbols. Time series are, in their turn, carriers of information about a physical measure, represented as a sequence of ordered real domain numerical data observed during a given temporal interval. Analysts can have a visual perception of the morphological behaviour of time series and can describe it in such a way that another person understands which segment of the time series he/she is referring to. As we mentioned in Chapter 1, 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, one 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 available well known approaches to search for specific words or patterns with text queries in traditional text editing softwares. regex are an example of such querying mechanisms. Other existing mechanisms rely in keyword-based queries, such as the *Google* search toolbar, which sorts web-pages according to how these match the set of keywords written. These are examples of the flexibility we find in tools available for query-based text search mechanisms. We imagined such tools could also be beneficial for time series query-based pattern search tasks.

In this chapter, we provide evidence that it is possible to use such reasoning with time series and propose solutions that promote higher flexibility, cognitive speed, and expressiveness in searching for patterns in time series as we would typically do with text. As presented in Figure **??**, two main strategies are proposed: (1) Synthatic Search on Time Series (SSTS) and (2) *Quo -where-* On Time Series (QuoTS). The first explains how to perform query-based search on a symbolic representation of the signal. A novel symbolic representation is introduced and the usage of regex is proposed to search for patterns directly in this symbolic representation. We also show how this representation can be used in time series classification tasks with Human Readable Time Series (HeaRTS). The second method that is proposed uses a feature-based representation of the signal, being each feature assigned to one word, which can be used with additional *operators* to search

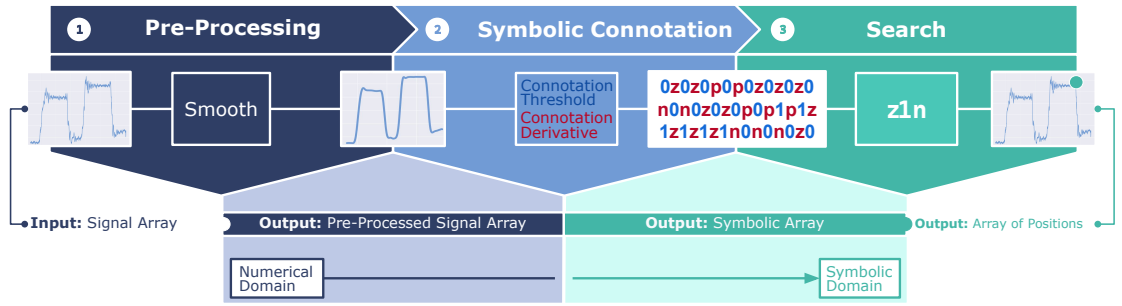


Figure 7.1: SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation, and search. This Figure also provides an example of the sequence of changes that occur in each step.

for the desired patterns with natural language.

The chapter will start by detailing how SSTS works, more specifically the symbolic conversion and the search process. In addition, Human Readable Time Series (HeaRTS) will be explained, namely how to use SSTS and its symbolic representation to find differences between time series and ultimately be used towards interpretable classification tasks. Finally, this chapter ends with QuoTS and how the analyst can use feature-words (keywords) with operators to search for patterns on time series more expressively and with words.

7.1 Syntactic Search on Time Series

In Chapter 3 was presented the well-known SAX, which is a symbolic representation of the time series based on its amplitude distributions. With SAX, query with text was not considered, but SSTS takes inspiration from this symbolic transformation by including additional attributes besides amplitude, such as the derivative, speed of the derivative, etc... (as will be listed further). Each property characterizes each sample of the signal. The combination of these attributes into a sequence of primitives is a symbolic characterization of the signal that enables the use of text-based querying mechanisms, such as regex, to search the desired patterns. Overall, the proposed tool benefits from the visual interpretation made by the analyst and provides SSTS to express it as a regex.

The proposed tool, SSTS, is structured in three stages:

- **pre-processing** - prepare the signal to highlight the desired patterns and ease the search process;
- **connotation** - a 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;
- **search** - a parser (regex) 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 SSTS process and will be accompanied by an example to elucidate how it works. The represented signal is the z axis of an ACC placed on the wrist of a subject while performing a rotating task at 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 7.1 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 SSTS.

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 [98]. 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 [58], 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 SSTS are presented in Table 7.1.

In the example of Figure 7.1, the *pre-processing* step uses the following string: Sm `500`, which indicates that a smooth filter using a sliding window with a size of *500* samples is applied to the signal. The resulting signal is showed under its original form. The selected area in the Figure 7.1 represents the segment of the signal that will be represented in subsection 7.1.2.

7.1.2 Connotation - The Symbolic Time Series

In semiotics, it is stated 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 [153]. The *connotation* step is the same connotative aspect but in this case, instead of a regular image, it is a time series. We follow the reasoning that the interpreter (the analyst) has made his analysis of the time series and retrieved the necessary information 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 that the user defines. Each of these conversion rules is designated a *connotation* method. These are responsible for converting

Table 7.1: List of common SSTS 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 $\overline{T} = \frac{T - \bar{\mu_T}}{\sigma_T}$, being: $\overline{T}$ the normalized signal, $\bar{\mu_T}$ 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

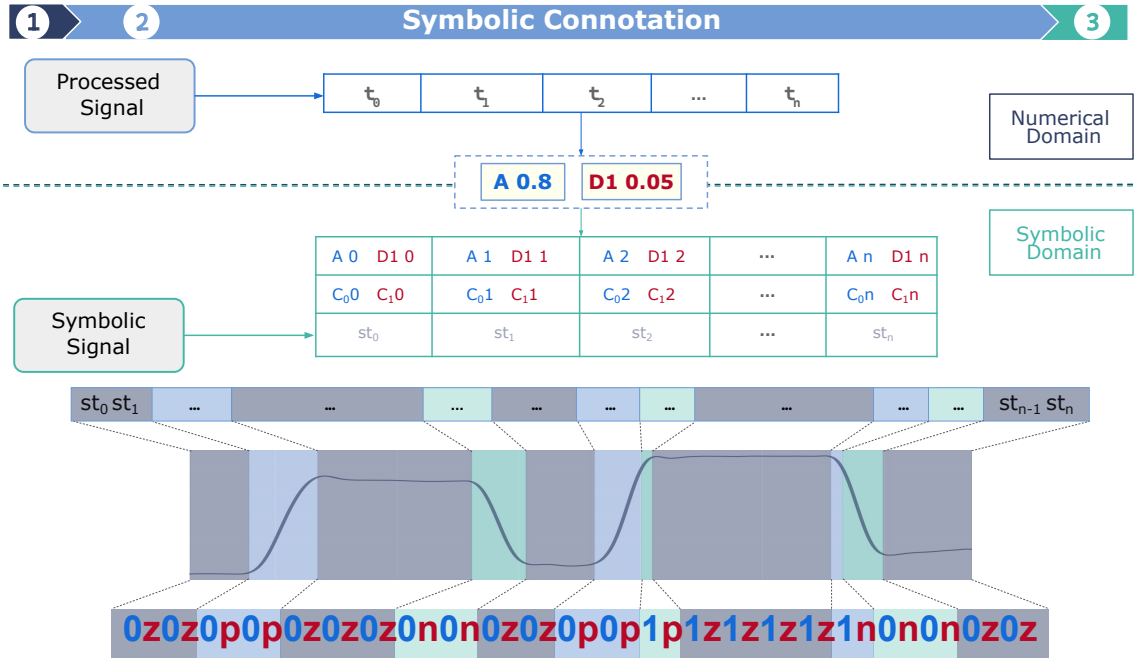


Figure 7.2: Here are highlighted the second stage of the process. It is the moment the signal is transformed from the numerical domain to the symbolic domain. A represents the amplitude method (blue) while $D1$ is the first derivative (red). The exemplified signal plus the symbolic translation are presented.

each sample of the signal into a *character* or group of *characters* that represent the state of the sample for that conversion rule (e.g. if the time series sample has a negative derivative, it will have the character `negative`). Each of the *connotation* methods is related to specific 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 a combination of multiple ones. In addition, the *connotation* process also handles multidimensional time series, in which cases the string is a combination of *connotation* methods corresponding to each time series, returning a group of *characters* for each of its samples.

Following the previous description, we recall the general text definitions from Section 3.7.1 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 7.2). 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: $st_i = < char_{C1} >< char_{C2} >< ... >< char_{Cc} >$, being $char_{Cc}$ the c^{th} *connotation* method applied. In case T is a multidimensional time series, the symbolic time series (MST)' samples from each time series are grouped $MST = (< st_{1,1} >< ... >< st_{1,k} >, ..., < st_{n,1} >< ... >< st_{n,k} >)$.

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/tokens used for the examples presented are listed in Table 7.2.

The symbolic connotation step of the exercise of Figure 7.1 demonstrates the latter formalism. In Figure 7.2, 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 using these two methods is expected to be 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 regex. This string is governed by the rules of regex from

Table 7.2: List of base SSTS 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 isn't; 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.

the *alternative regex Python module* [48] and benefits from all the functionalities and metacharacters typically used with this tool, which can be recalled from Chapter 3 (see Section 3.7.3). The search procedure returns the intervals at which positive matches occurred. It is relevant to note that it has been decided to use regex, although any other parser, conveniently combined with the previous steps, might be used for the same purposes.

In the example of Figure 7.1, the purpose is to determine when a plateau superior to *80 %* of the amplitude range of the signal starts to fall. The symbolic time series matches this moment when a the sequence of characters starts with a `1` (*time series is higher than the 80 % threshold*) and is followed by the sequence `zn` (*a sequence of stationary to falling*). The regex used for the search is precisely `z1n`, which indicates that the signal, at some point superior to the threshold will start to decrease, as presented in Figure **??**.

Figure 7.3: Step 3 of SSTS. Specific *subsequences* of the time series can be searched with a regex. In this case, the search is to find the moment a `plateau` starts to fall, which is found with the regex `z1n`. Blue are the symbols for amplitude and red for the first derivative.

In the next Chapter we will show several applications of this method for pattern search tasks. Next, we present how to use it for classification purposes.

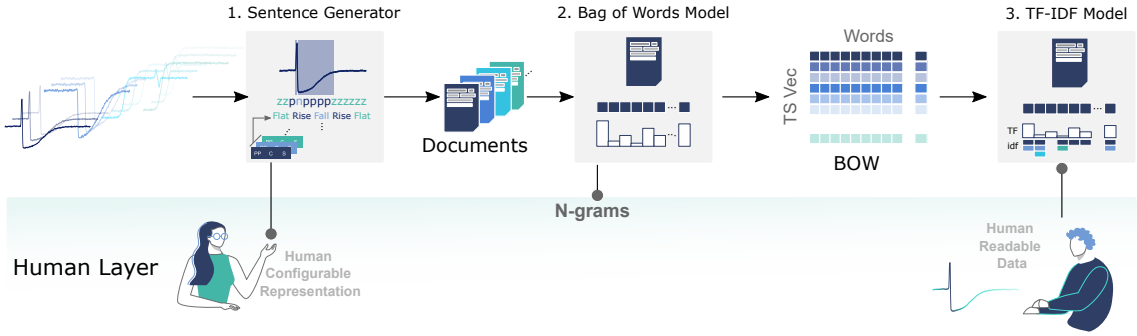


Figure 7.4: Steps of transforming time series into documents and corresponding BoW and TF-idf matrices for posterior classification. The steps are (1) *sentence generation*: all time series are converted to time series *documents* multiple SSTS queries (e.g. `Flat`, `Rise` and `Fall` from the derivative *connotation*); (2) *bag of words model*: each document is converted to a vector with the frequency of words in that document; and (3) *tfidf model*: transforming the BoW to TF-idf and possibly searching for relevant segments of the time series. The human layer shows where the human can apply his/her knowledge, namely selecting the SSTS queries and interpreting the highlights on the signal.

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. Profiting from such methods can result in different outcomes that can complement the ones resulting from current strategies, improve results or even open novel pathways for further research experiments in this domain.

We propose to use the existing text-mining knowledge on this novel representation for 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 the text mining domain, the differences between text *documents* are traditionally computed with a feature extraction process on text, such as BoW or TF-idf, returning a statistical representation of the words or n-gram. *Documents* with different words and/or sequences of words will have different weights on these models, which are used to perform a class separation on the *documents*. We take inspiration from this process and apply it to time series. For instance, the following time series  can be very broadly translated into `Flat Rise Fall Rise Flat` or `Flat Peak Valley Flat`, while the following signal  would be translated into `Flat Fall Rise Flat` or `Flat Valley Flat`. This type of description is easily performed with SSTS, as showed on Figure 7.6. Figure **??** shows the main steps to perform a transformation of the time series into a BoW/TF-idf 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 SSTS. The fact that the time series is translated into text also provides a layer of readability and interpretability to the data, as we will explore throughout this section as well.

7.2.1 SSTS to Generate Time Series Documents

The generation of time series documents (See Section **??**) is made by converting samples of the signal to characters, search patterns that are tagged with specific words in which these samples are converted to, and order them by their time index. With this, the time series can be translated into sentences and be described as a text document.

In order to perform this description, the words have to be associated with a specific SSTS query. Each query has the three stages of SSTS and searches for the patterns that match on the time series. The patterns that are matched are associated with the corresponding *word*. As an example, the query `{PP: Sm 25, C: D1 0.05, S: p+}` searches for instances where the time series is increasing and attributes to these *subsequences* the word `Rise`. A specific set of SSTS queries are used to build a full description of the signal dynamics in higher leveled structures, mostly related with amplitude and derivative *connotations*.

The selection of SSTS queries is made by the analyst, either using a pre-defined set of queries or creating his/her own, more appropriate for the type of signals to analyze. This

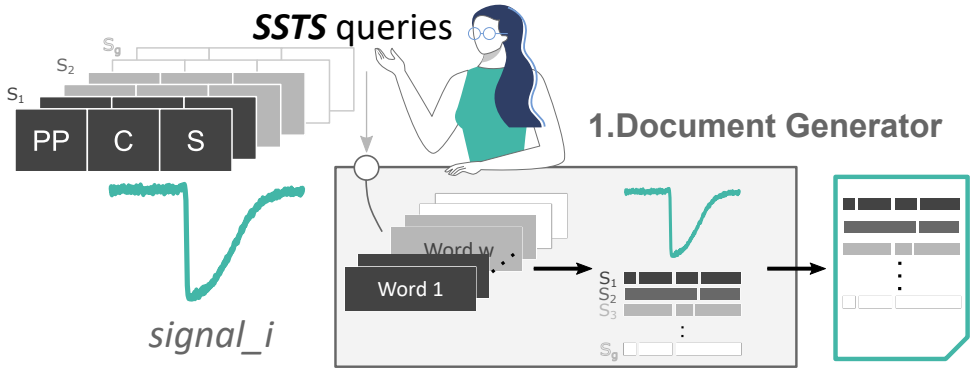


Figure 7.5: Steps for the generation of sentences from a raw time series and organization as a time series *document*. Here: PP - Pre-processing, C - connotation and S - Search are each SSTS queries used to search for the patterns and attribute the matched *subsequences* to a *word*. The colour tones (dark grey, light grey and white) indicate each sentence group ($S_1, S_2, ..., S_g$), which has multiple SSTS queries. The result is a *document* with multiple sentences.

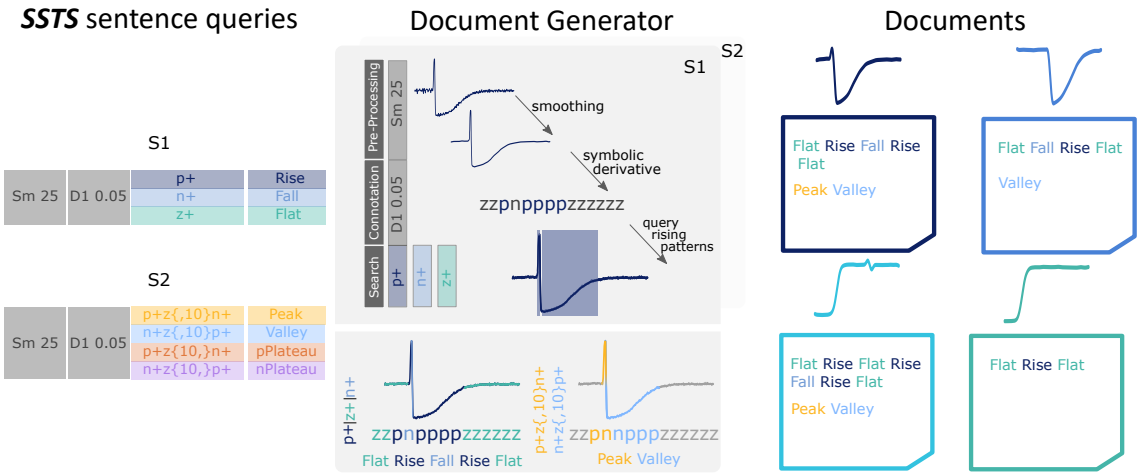


Figure 7.6: Example of converting a time series into a time series document with SSTS. **SSTS sentence queries**: Two groups of sentences (S1 and S2) are used to extract words on a signal. Each sentence group has multiple SSTS queries, to which a word is associated (e.g. `p+` → `Rise`). **Document Generator**.Top: Using SSTS to detect the rising *subsequences* (`p`) 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* - regex `p+` searches for all sequences with one or more `p` characters. **Document Generator**.Bottom: Example of sentence generation. Using the other search queries (`p+, n+, z+`), we can find the derivative patterns and convert them into ordered words. **Documents:** the time series documents generated for each of the exemplified signals with the two sentence groups used.

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 (such as simply explaining if the signal is `Rising` or `Falling`). However, in some cases, the overall dynamic of signals might be dissimilar to other dimensions. In order to have a rich foundation to perform a sentence description of a time series, we created the set of SSTS queries presented on Table 7.3, 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 even 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 in Figure 7.6. The process highlights the usage of SSTS 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 SSTS 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.

Having now a method to translate time series into text, we can use text mining methods directly on the *documents*. Typically, text data is vectorized with the BoW or TF-idf models, a process explained in the next Section.

7.2.2 Vectorization of Time Series Documents

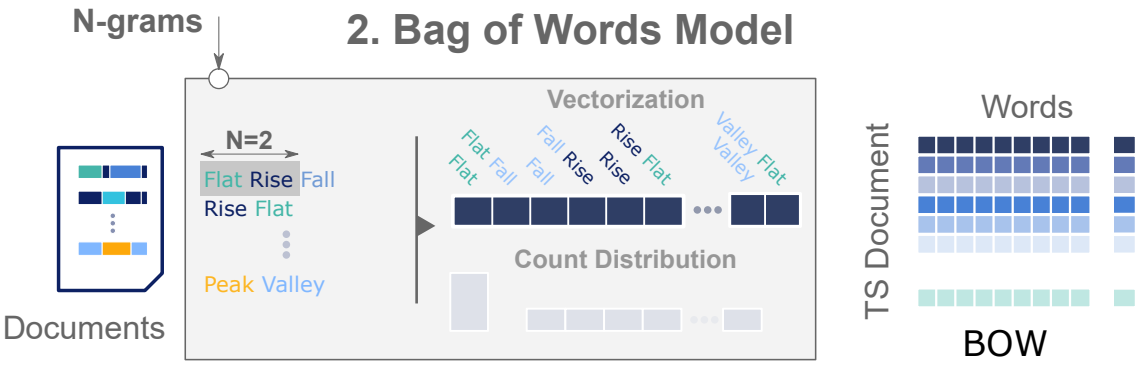
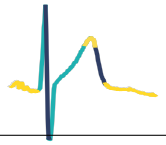
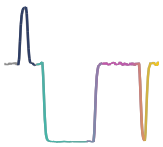
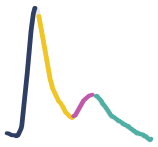
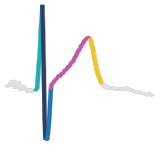
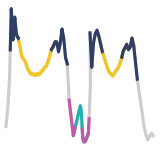


Figure 7.7: Process to transform a time series document into a vectorized representation of words and n-grams. The user can set the size of the n-gram. The documents are transformed into the BoW having a count distribution, which is the tf.

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.

Table 7.3: The connotation variables, search regex 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, *thr* is the threshold for the derivative functions. Each word has a representation on an example of a signal.

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	

After converting the time series into *documents*, the text is analyzed to build a matrix of *word* or *n-gram* frequencies, the BoW model. In this case, the features extracted are purely statistical, but can provide a relevant measure of differences between documents. Each row of the BoW model is a time series *document*, represented by columns that have the number of occurrences of a *word* or *n-gram*. Figure 7.7 shows the vectorization of a time series *document* as one row of the BoW model.

The BoW model can be used in several ways. Following guidelines from the text mining domain, it can be used for unsupervised or supervised tasks, for topic modeling and keyword extraction. The BoW model can also be trained with Naive Bayes Classifiers, Support Vector Machine or Linear Regressors [118]. In addition, the BoW model feature matrix can be converted into the TF-idf 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 3 (see Section 3.7.2), one of the reported issues of the BoW model is evaluating the relevance of a word based on its frequency in all documents, while not considering that words that occur in all documents might be less relevant. Therefore, we decided to use the TF-idf model, which increases the relevance of a word using its raw occurrences, while reducing its importance in proportion to the number of *documents* that contain that word. The model is defined by being a ratio between the tf and the inverse document frequency (idf) (equation 3.19).

Both the BoW and TF-idf 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 TF-idf model, all the other time series *documents*. The difference between the distribution of each *document* can be used with a conventional distance measure to compare them. In Figure 7.8 is showed the distribution of *word's* weights for four different time series *documents* of the *Trace* dataset (see Section 5.1), based on a TF-idf 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 time series based on them. On the right of the distribution, the reader can see a dendrogram of sorting examples from the *Trace* dataset. We used the cosine distance between vectors of time series documents (Symbolic Distance) and compared it with the ED. Surprisingly, the ED 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 ED.

As the reader can notice, the ED 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 have the same shape. On the other end, the proposed strategy follows a distance measure based on the presence/absence of specific

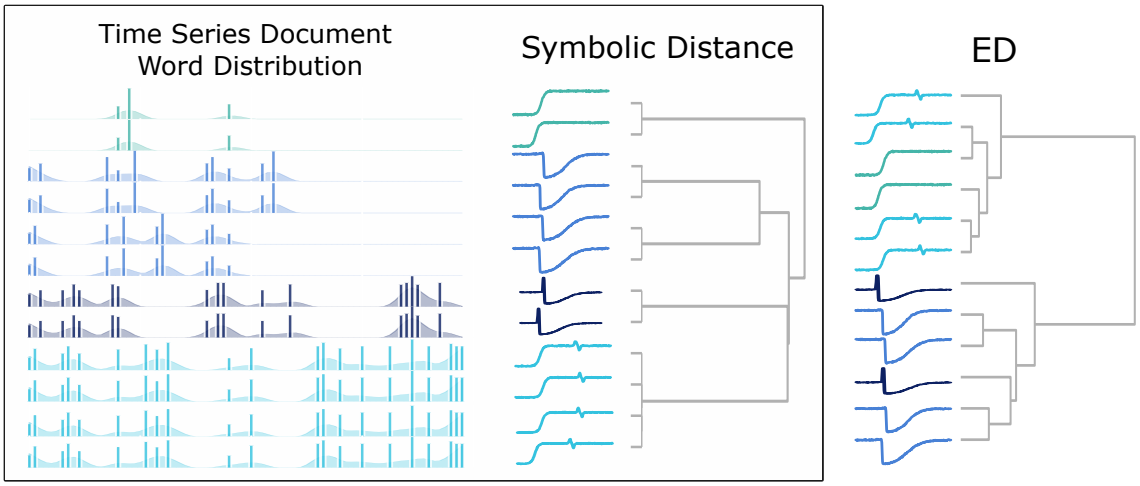


Figure 7.8: Comparing the clustering process of the ED and the proposed symbolic method (Symbolic Distance). In the proposed method, each signal has a word distribution vector, represented on the left. The signals are clustered based on these.

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 differentiates classes of signals. This advantage is taken because of the weighting factor attributed to each *word* or *n-gram* from the TF-idf model. As explained by Pavel Senin *et. al*, the weights can be used as a weighting vector for each word extracted, highlighting the corresponding shape on the original signal and measuring its importance for the classification process [136].

The process to highlight the areas of the signal that are more relevant and able to explain the difference between classes is illustrated in Figure 7.9. From the BoW model, the TF-idf model is computed, following equations 3.17, 3.18 and 3.19. 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 SSTS 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 (WTS) with the weighted contributions of all *words*:

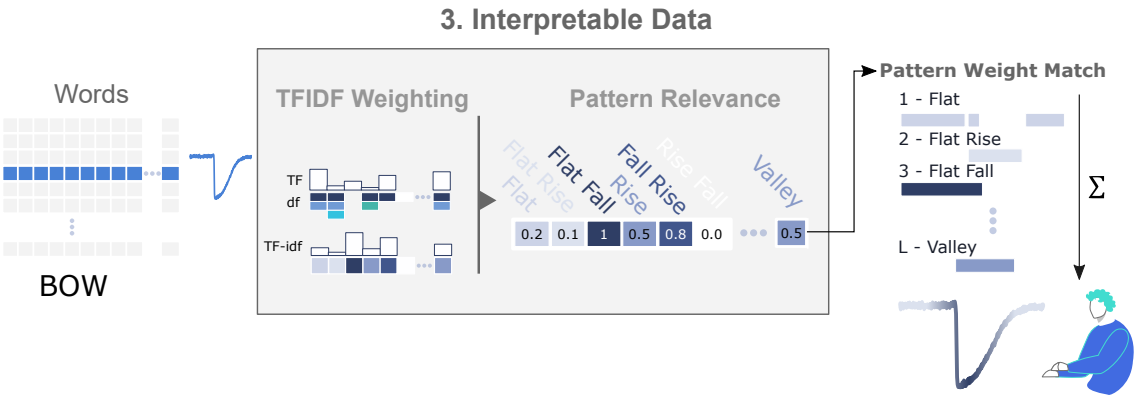


Figure 7.9: From the TF-idf matrix has weights for each word or n-gram (pattern relevance). The search of these words and cumulative attribution of weights on the segments matched can represent a feedback tool. In this case, the valley of the signal is highlighted.

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

In Figure 7.9, the exemplified signal has a higher relevance on the `valley`, being the most characteristic element of that time series.

7.3 Towards Natural Language for Pattern Search

The fact that an analyst can search with regex 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. In this journey to explore the topic of text-based queries for time series pattern search, we established the possibility of making an association between features and keywords to perform a search similar to what is done with web-searches toolbars. Thus, we propose a novel method, QuoTS, for pattern search, but in this case, it is based on natural language keywords and linguistic operators. In contrast with the previously explained method, SSTS, QuoTS associates *words* with features, and such as SSTS, contributes to the democratization of pattern search on time series.

7.3.1 Browsing Patterns Search

When a user opens any search toolbar (such as *Google*, *Safari*, among others) and searches for a specific *topic*, he/she will write a set of keywords that can match the *topic* he/she is searching. The results of this process are a list of pages ranked based on how well they matched the *keywords* used. Of course that in this example, search toolbars might have additional information, and not solely rely on the *keywords* (maybe including geolocation, browser history, etc...) for the search process. However, let us take into account this simple *keyword* based search process that we typically use when browsing the internet and apply it to pattern search on time series.

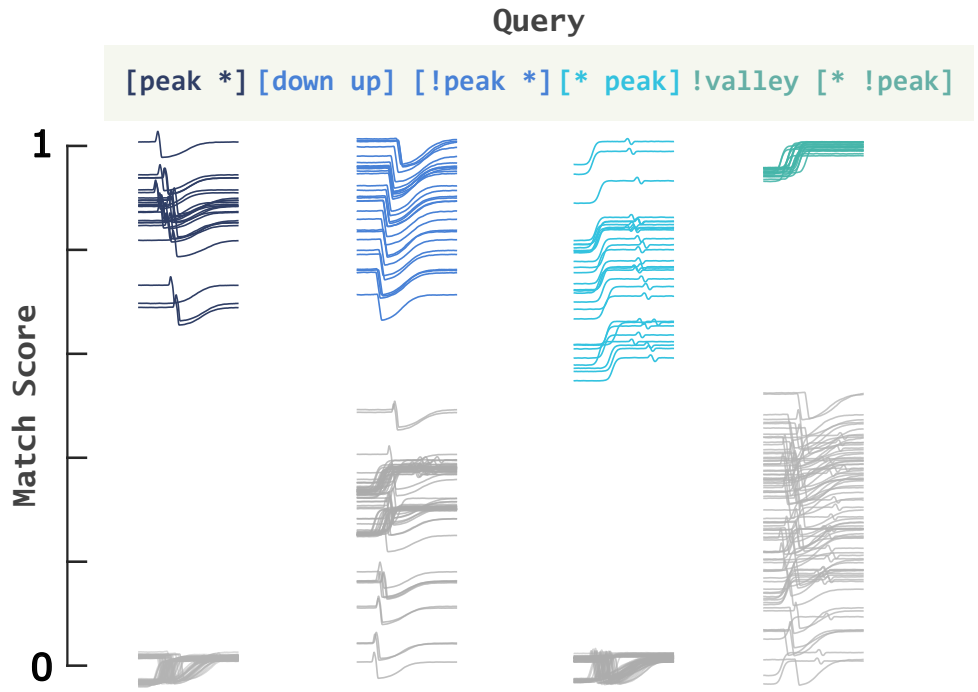


Figure 7.10: Using our proposed query language, we can create short, intuitive natural language queries to rank the 100 exemplars in Trace, separating our sought class from the remaining data. Here `*` is *anything*, `!` is *not*, and *square brackets* are a grouping operator (more details in Section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

As we mentioned above, patterns can be described with *words* by associating text 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 behaviour, he/she can simply *Google it*, by describing the data he/she expects to see, given that behaviour. For example, we considered the four-class Trace dataset (see Section 5.1), 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 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 in 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 in 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 in Figure 7.11. In order to perform the

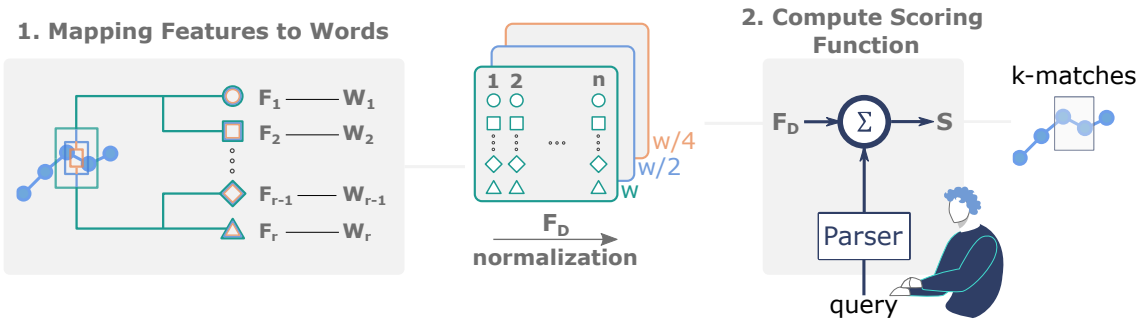


Figure 7.11: QuoTS steps. It shows the feature extraction process and matches each feature to words ($W_1, W_2, etc...$). These features are computed in three window (w) dimensions (w, $w/2$ *and* $w/4$). The user can input a query to search for a specific pattern. This query will be scored based on the parser and features. It then returns the top k-matches.

search with natural language keywords and operators, we represent the time series into a set of feature series. The features are extracted with a *moving window*, with total overlap. The process is made three times, with decreasing window sizes (w) dimensions (w, $w/2$ *and* $w/4$). 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 is 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 keyword representation is expressive enough to discriminate specific behaviours 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).

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

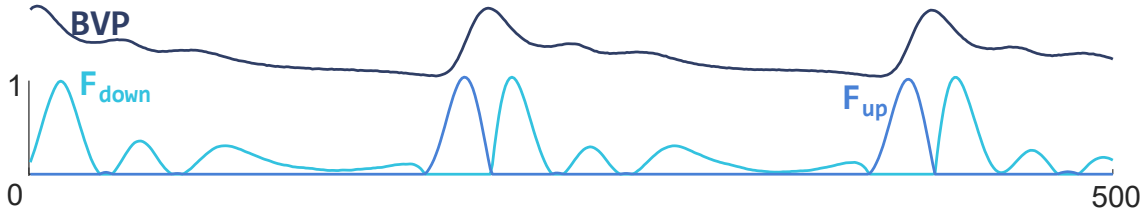


Figure 7.12: Example of a word feature vector. In this case F_{down} and F_{up} represent a negative and positive slope value.

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 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 with 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 7.3. 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 7.4).

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

In 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., $[W_1 W_2 \dots W_w]$. This list can be expanded and customized, but we want to demonstrate that with a minimal set of operators, most of the problems we present are solvable.

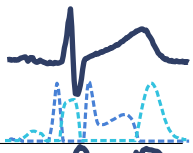
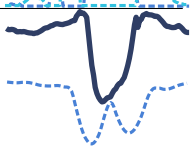
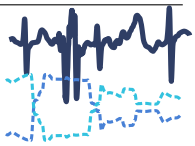
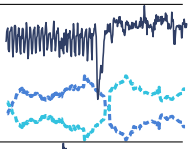
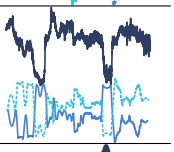
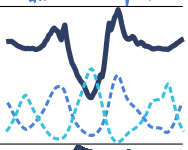
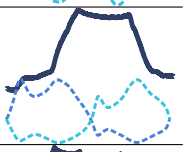
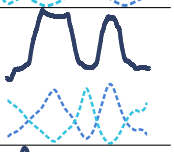
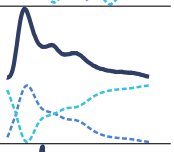
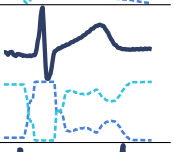
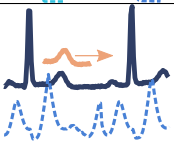
Web search engines have many operators at the user's disposal, but since a list of words is usually powerful enough to retrieve and correctly sort most of the desired results, very few (or none!) are often used. We believe that this is the case for this application as well but acknowledge that simple 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 in 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 7.3. In these cases, or when the word has no direct opposite, it can be useful to penalize the presence of the word in a *subsequence*. This operator does that by applying Equation 7.3 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.

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. The colors of the *words* are the corresponding colors on the *example*. Each example has representative feature vectors. When a negation pair is present, it is added to the example.

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 3.14. (`simple = !complex`)	
Symmetric (**Asymmetric**)	The normalized ED (3.8) to the *subsequence*'s horizontally flipped self.	
Peak (**Valley**)	The logarithmic normalized ED (3.8) to the template of a peak (valley), modulated by a gaussian function.	
StepUp (**StepDown**)	The logarithmic normalized ED (3.8) to the template of a step-up(down) function.	
PlateauUp (**PlateauDown**)	The logarithmic normalized ED (3.8)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 ED 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 into the query language.	

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 can 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 7.11, 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 different window sizes. 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, QuoTS 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

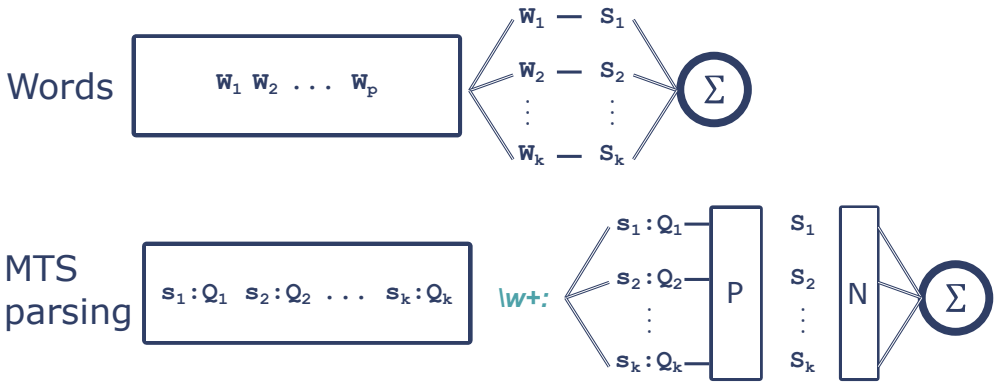


Figure 7.13: Process to parse the input text query. When having simple space separated words, each corresponding word feature vector is summed together. When in multidimensional mode, there is the *signal id* first.

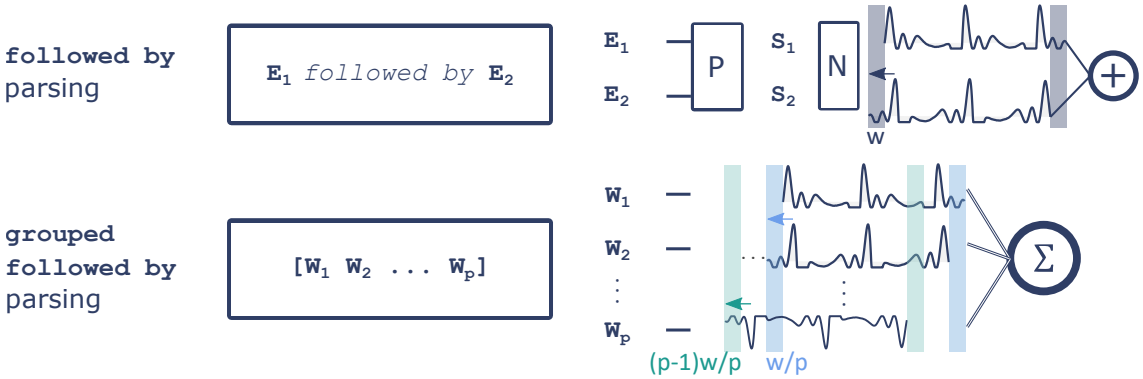


Figure 7.14: Second order of the parser. In case the `followed by` operator is used, the elements (E_1 and E_2) are parsed (P) and word feature vectors are summed, but the E_1 has to be delayed to count as the "previous" instance (w represents the selected window size). When using the more specific `grouped followed by` operator, square brackets are used and represent the window size highlighting the signal. Only words can be used in these brackets and are added together, delaying each word based on their position inside the brackets, for instance, word 2 (W_2) is delayed by the window size, divided by the number of words inside the brackets (w/p).

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 regex 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 regex 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 regex \[.+?\] (*anything inside square brackets*). The content of the square brackets is then combined (see Figure 7.14).

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 7.13 and 7.14 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 with the corresponding *word*, is a score (S) function and is 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 is treated individually by the standard parser (P on Figure 7.13), 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 7.13) to have equal weight. It is important to note that Q can be a query with *words* or the `grouped followed by` operator.

The temporal operators are parsed by the regex, 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 has 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

Experimental Analysis, Validation and Discussion

The previous chapters (Chapters 7 and 6) introduced the methods developed in this thesis. For each topic, a few examples were presented to help understand and follow the explanation and description of the proposed methods. In this chapter, we go further into testing and validating the algorithms to provide evidence that these are relevant and contribute to the state-of-the-art in the time series data mining field. The performance was evaluated on public datasets, introduced in Chapter 5.

Considering the diversity of methods developed, this chapter will be divided into each contribution following the topics presented in the previous Chapters. Figure **??** shows the content provided in this chapter, being divided in:

- **Information Retrieval with the SSM**: We presented three main applications of retrieving information from the SSM, namely for novelty segmentation, periodic segmentation and similarity association. The process for novelty segmentation was validated in public datasets and its performance compared with two referenced methods (WL - window based segmentation, and BS - binary segmentation). In addition, we applied this strategy in a change point detection benchmark, where we were able to compare its performance with all methods available in the benchmark. We show an application example for periodic segmentation and query-based segmentation in one public dataset, and results are computed based on ground-truth annotations. We additionally show a few examples of similarity measures after the segmentation process and how to apply this information retrieval in the summarization of biosignals.

- **Pattern Search with SSTS**: In this topic, we provide use-cases where SSTS is used to retrieve specific patterns. Each step of the proposed method are highlighted. Specific metrics (see Section 3.8.3) are used to compare the SSTS query used with a standard python-based implementation for the search problem.

- **Time Series Classification with HeaRTS**: Having demonstrated the possibility of using SSTS to search for patterns, we explore its usage to extract a textual representation of time series and apply text-mining methods to perform the classification. A public benchmark was used to perform this analysis and HeaRTS was compared with the 1-NN ED. We also provide evidence in several use-cases of how the weights of the TF-idf model can be used to highlight the most relevant areas of the signal, contributing to the interpretability of results.

- **Pattern Search with QuoTS**: The second proposed method for pattern search in time series will be evaluated with several use-cases. We also evaluate its ability to separate classes of hand gestures with simple queries. Finally, we also show an example of *pupetteering* as a toy problem, and as a possible application of this method.

8.1 Information Retrieval with the Self-Similarity Matrix

8.1.1 Novelty Segmentation

In this section, we present several examples of how the novelty function (recall Section 6.3.1, retrieved from the SSM, is useful for the segmentation of time series. We also provide a measure of the algorithm's performance considering ground truth events from public benchmarks, while also comparing our proposed solution with existing methods for change point detection from the *Turing Change Point Detection Benchmark* [25].

8.1.1.1 Statistical Performance Evaluation

In order to evaluate the performance of our proposed method in general scenarios, we applied the algorithm to Datasets 3, 4, 5, 6, 7 and 8 (see Sections 5.3, 5.4,5.5,5.6, 5.7 and 5.8). The evaluation was divided into *biosignals-related applications*, and the general change point detection benchmark (Dataset **??**). The biosignals experiments are associated with public datasets from Physionet, the *UCI Machine Learning Repository* and the *UCR Semantic Segmentation Benchmark*, involving different contexts (HAR, hand posture, and noise detection) and sensor types (ACC, GYR, EMG, ABP and ECG).

The quantitative evaluation on biosignals' public datasets was made by accumulating true positive (TP), false positive (FP), and false negative (FN) values with a tolerance zone around the ground truth events. The applied reasonable tolerance was the ground truth wrapped by a window size of the SSM computation, inside which a detected event was counted as a TP. The case that no estimated event was found inside the tolerance band was considered an FN. An estimated event outside the tolerance or duplicating to an already counted TP were regarded as an FP. The F1-score based on the precision and recall values was calculated from TP, FN, and FP values (recall definitions and equations from Section 3.8.2).

In the evaluation of biosignal segmentation, the performance of our proposed method was compared with existing approaches available on the Python library *ruptures*, namely the window-based segmentation (WL) and the binary segmentation (BS) [143]. The benchmark evaluation referred to the best score obtained from the state-of-the-art methods available on repository [25]. The evaluation procedure to detect TP, FP and FN was the one followed on [25].

The method has been computed in the same conditions and followed the same procedure for all datasets' records. The features used were the same for each record (see Appendixes A and B), varying the timescale parameter, the overlap size of the sliding window, and the kernel size. The peak detection strategy based on a threshold mechanism is the same for all records, while the threshold value varies from record to record. Results for publicly available datasets are listed in Table 8.1, and Table 8.2 expands the performance by F1-scores in detecting the change point events.

Table 8.1: Overall results for the performance of the method on novelty segmentation, including true positives (TP), false positives (FP) and false negatives (FN), precision (P), recall (R) and F1-score (F1) values. The last row provides the macro averaged F1-scores (*M.A. F1*) of the four datasets.

	Novelty						WL						BS					
Dataset	TP	FP	FN	P	R	F1	TP	FP	FN	P	R	F1	TP	FP	FN	P	R	F1
1	166	16	13	0.91	0.93	0.92	169	9	10	0.95	0.94	0.95	125	58	54	0.68	0.70	0.69
2	18	2	0	0.90	1.00	0.95	9	5	9	0.64	0.50	0.56	13	14	5	0.48	0.72	0.58
3	137	39	31	0.78	0.82	0.80	92	88	76	0.51	0.55	0.53	122	58	46	0.68	0.73	0.70
4	695	68	33	0.91	0.95	0.93	608	123	120	0.83	0.84	0.83	351	413	377	0.46	0.48	0.47
5	3433	469.0	366	0.88	0.90	0.89	2941	327.0	858	0.90	0.77	0.83	2618	1322.0	1181	0.66	0.69	0.68
6	771	194.0	105	0.80	0.88	0.84	672	201.0	204	0.77	0.77	0.77	646	280.0	230	0.70	0.74	0.72
M.A. F1	-	-	-	-	-	0.89	-	-	-	-	-	0.75	-	-	-	-	-	0.64

The illustrative examples provided in Section 6.4 corroborate the proposed method's capability in segmenting real, complex, and multimodal biosignals datasets. As Table 8.1 conveys, an overall macro-averaged 0.90 F1-score is achieved, while the competitors' overall F1-scores are 0.72 (WL), and 0.61 (BS), respectively. Tables B.1—B.2 in Appendix B details the parameters of window sizes, kernel sizes, and thresholds applied to the signals in each dataset.

For Datasets 3 and 6 (see Sections 5.3 and 5.6), the window-based methods, *novelty* and WL, performed much better than the BS method, mainly because the sliding window algorithm with a full set of features comprehensively characterizes changes in the signal. The standard WL uses cost functions searching for mean/variance value changes in the signal, which achieves a high F1-score in Dataset 3, even identifying transitions between dynamic activities, such as *Walking*/*Upstairs*. Our proposed method had a similar performance with a worse count in FP values, where the added features did not improve the segmentation performance. In contrast, our proposed method, complemented by additional features, had a much better performance than the WL method in Dataset 6. Adding features enabled a more robust and sensitive detection of pattern changes, although it missed some changes between similar patterns, like *Walking* and *Upstairs/Downstairs* in

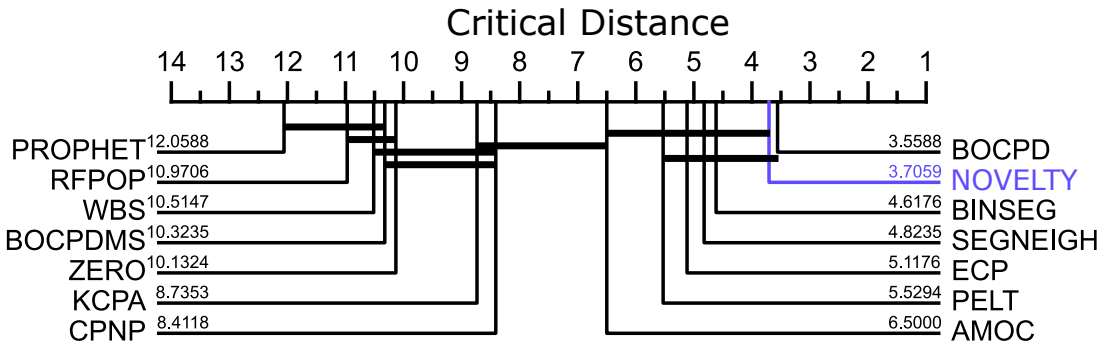


Figure 8.1: Critical distance diagram comparing the methods used in [25] (except *RBOCPDMS*) and the *novelty function* on Dataset 10 (see Section **??**). The performance measure corresponds to the F1-score for all single-dimension datasets of the benchmark, except for the ones identified in Table 8.2 with a gray background. A thick horizontal line groups a set of classifiers that are not significantly different in the statistical test [70].

Dataset 3, which are the primary source of the FN value. Similar to the FN values, the FP values of our proposed method are mostly superior to other methods. It nonetheless leaves room for discussion. Some events are not marked as changes in specific activities, but the signal pattern actually changes. For example, Figure 8.17(bottom) exposes pattern changes during an *Upstairs*/*Downstairs* activity unlabelled in the ground truth, possibly due to a flight of stairs. The novelty function is sensitive to such pattern changes, which inevitably contributes to the FP values during the comparison with the ground truth. Considering the good performance of both methods, further research should be made in other HAR domains to better understand the differences in performance between them.

For Dataset 8 (see Section 5.8) and, especially, Dataset 7 (see Section 5.7), the proposed method outperformed its competitors. Although ECG-based jump artefact detection is fundamental, the WL method could not find the segmentation borders, while the BS method worked better. In Dataset 7 (see Section 5.7), the same ECG signal was noised with different Signal-to-Noise-Ratio (SNR) levels to form a new set of resultant signals. Overall, our proposed method was able to detect the presence of noise up to 18 dB. At 12 dB, the proposed method achieved an average F1-score of 0.84 with references of 0.43 (WL), and 0.69 (BS).

8.1.1.2 Segmentation Benchmark

In order to compare the proposed method with other state-of-the-art approaches, we used a benchmark provided by the Alan Turing Institute [25] (Dataset 10). The performance was evaluated by change point event detection in each time series available, summarized in Figure 8.1, where each referenced method applied its best score in the benchmark (see Tables 8.2 and B.7).

As unfolded in Figure 8.1, the critical distance diagram ranks the proposed method second, suggesting no significant difference in performance among methods that only work in uni-dimensional datasets. The global average F1 measure of the proposed method is 0.87

Table 8.2: Comparison of the F1-scores between our proposed method (*novelty*) and other algorithms' benchmarks in Dataset 10 (see Section **??**). The calculation of all one-dimensional signals' average performance and all signal's average performance does not include columns with a gray background where no change point should be detected, or a signal error was present. Bold values represent the best F1-score for this specific dataset. T: timed out; F: failed compiling.

Dataset	NOVELTY	AMOC	BINSEG	BOCPD	BOCPDMS	CPNP	ECP	KCPA	PELT	PROPHET	RBOCPDMS	RFPOP	SEGNEIGH	WBS	ZERO
One-dimensional															
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_employment	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
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
Multidimensional															
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 (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

for both uni and multidimensional datasets. The two null scores are because no change point was supposed to be found in the corresponding time series. The results obtained in this benchmark restate that our proposed method is promising, having a performance that competes with several state-of-the-art methods in the problem of novelty segmentation. It should be stressed that the proposed method applies to multidimensional time series, while two of the best-ranked methods in Figure 8.1 do not. In addition, the proposed method retrieves not solely segmentation points but also higher-level information as periodic changes and cross-segment similarity measures, which is an advantage over the *BOCPD*.

8.1.1.3 Time Complexity

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 r is the number of features and n is the size of the inputted signal). Finally, the correlation of a kernel on the SSM's diagonal has a complexity of $O(nM^2)$ (recall that $M = 2L + 1$, being the size of the sliding kernel).

The sliding window to extract features has an $O(n)$ time complexity. The dot product between matrices has a conventional $O(m^2n)$ time complexity. Expressly, m and n represent the number of features and the inputted signal's size, respectively, in our proposed method. The correlation computation of a kernel on the SSM's diagonal has a complexity of $O(nM^2)$, where the sliding kernel's size M equals $2L + 1$.

8.1.1.4 Discussion of Novelty Segmentation Results

Several parameters affect the detection results of desired patterns, especially the window size, the overlap percentage, and the kernel size, which influence visual outputs and the novelty function. 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, corresponding to lower *zoom values*, allow the similarity calculation of longer *subsequences*, while smaller windows, serving like a *zoom-in* function, search for local details and unobtrusive changes.

- The overlap percentage, working as a down-sampler of the time series, is the *camera sensor*, which determines the image's pixel resolution. A full resolution of the SSM is only achieved with total overlap, and the lower the overlap percentage, the less accurate are the highlighted changes.

- The sliding kernel's size concerns the novelty function's sharpness of the detected changes. The larger it is, the smoother the output function will be. Potentially, the kernel size can be scaled to the window size, even with a slight accuracy decrease.

With enough computational resources available, the overlap percentage can be maximized so that the SSM can mirror the full details. Admittedly, such an operation is not necessary for many real applications, but reduces the variables to facilitate other parameters' tuning experiments, which is one of our subsequent research topics. The computational resource, i.e., the memory bandwidth and the calculation time (see Section 8.1.1.3) is a limitation in this current stage, since the SSM increases exponentially with the increase in the time series size. We ascertained that downsampling the time series with a lower overlap percentage is a valid option, advanced by a hierarchical search strategy for addressing the memory limitation, as exemplified in the walking-series instance in Section 6.4.1. Another potential efficiency-enhancing solution is to only compute the SSM's central diagonal with the kernel size corresponding to the interest areas of segmentation borders, which obtains efficiency gains in exchange for the sacrifice of periodicity and similarity measures between *subsequences*.

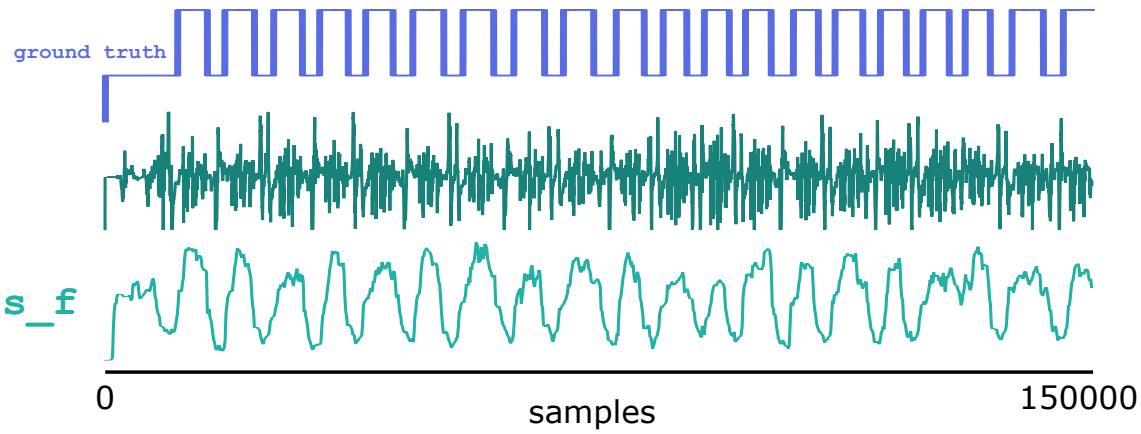


Figure 8.2: Segmentation of a walking pattern on a 1D signal with the similarity function (s_f). The window size was 5000 samples and the overlap percentage of 75%.

The threshold used to determine which peaks are considered as points of interest is not relevant to the SSM calculations but closely related to event detection and automatic segmentation. If the data is a black box, the choice of threshold is a matter of observation and speculation. With informed knowledge of the data, the threshold can be predetermined and experimented with rules based on ranking the detected peaks from highest to lowest:

- Set the total number (or quantity range) of points of interest as the threshold. E.g., an ECG series with x heartbeats; an ACC series with y recorded gaits of walking activity.
- Count the total number of peaks and specify a percentage as the threshold. This work takes such an approach because of the diverse datasets and signals involved.
- Add sliding windows to the novelty function based on the known periodicity information of the time series, and define the number of points of interest expected to be present in each window as the threshold.

8.1.2 Periodic and Query-based Segmentation

The detection of periods onset was experimented on dataset 12. It is a database with exclusively periodic data of activity signals (see Section 5.12). Twenty participant performed seventeen different activities, which are recorded in a multimodal time series. We exemplify the usage of a periodic search with the similarity function and the query-based segmentation on the SSM. These experiments were evaluated based on ground-truth annotations of a push-button from the dataset, with the metrics previously explained (see Section 3.8.2). The ground truth indicates the onset of the activity. As an example, we show a walking pattern to analyze:

Using the similarity function (sf) enhances the periodic nature of the time series when computed (see Section 6.3.2). This method was applied to this dataset, being the results

displayed in Table 8.3 for all activities, averaged by all participants. The periodic search was made using the window size of one period. The tolerance window was half the window size. The *similarity function* was smoothed by a smoothing function (see Section 3.2.2) with a window size equal to the window size used to extract the features.

Table 8.3: Detected cycles and DE results of the detection of onset of activity, over signals of database 12.

Activity	Detected Cycles	TP	FP	FN	F1	DE(%)
1	19/19	TP	FP	FN	F1	13,25
2	19/19	TP	FP	FN	F1	8,81
3	19/19	TP	FP	FN	F1	8,82
4	19/19	TP	FP	FN	F1	5,82
5	19/19	TP	FP	FN	F1	9,73
6	19/19	TP	FP	FN	F1	6,69
7	19/19	TP	FP	FN	F1	6,18
8	19/19	TP	FP	FN	F1	5,26
9	19/19	TP	FP	FN	F1	6,66
10	19/19	TP	FP	FN	F1	6,97
11	31/31	TP	FP	FN	F1	7,22
12	19/19	TP	FP	FN	F1	8,19
13	19/19	TP	FP	FN	F1	8,96
14	19/19	TP	FP	FN	F1	12,30
15	19/19	TP	FP	FN	F1	6,81
16		TP	FP	FN	F1	
17		TP	FP	FN	F1	

Even though the signal is confusing, the method is still able to find the periodic nature. Further, we will show that the method is also robust in more complex datasets, such as with signals from the occupational domain. These signals can have more complex periodic subsequences, where even some motion variety exists from cycle to cycle.

Although this method was successful in segmenting the signal in the correct number of segments, the proposed solution is not yet optimal. It does not always provide the moment where the *path* starts on the SSM. This moment should be the one detected to capture the true starting point of a period. The fact that we are not able to find the exact moment the *path* starts means that an offset is present on the cycle detection. In the future, a different approach can be developed to segment the *paths* on the SSM. Nevertheless, the SSM can still be used to segment periodic signals with the help of the analyst, by interactively selecting a segment of the signal as a query, or simply by indicating the index that corresponds to the beginning of the period. The method to perform this is simply applying the concept of query-based search but with columns/rows of the SSM, as explained in Section **??**. This query-based search was also used in the same dataset and the results can be found in Table **??**.

The query-based search was performed on the same dataset (dataset 12, Section 5.12. The query used was one period of the signal, being the corresponding portion on the SSM

scanned along with the method explained in Section 6.3.4.

It is also important to mention that this problem is not necessarily hard to solve for this dataset, considering that it is a controlled dataset with one periodic signal. But our intention with this experiment was to demonstrate the ability of the SSM in (1) highlighting if a segment of the multimodal signal is periodic with the presence of *paths*, and (2) segmenting the periods of this segment without any prior knowledge on the signal. In this case, we used the exact period of the signal, but the method is highly flexible and can show the paths in a large span of window sizes. Solely finding periodicity can be done with other methods, but in this case the SSM shows all variations on a time series, with novelty and periodicity. We have showed several examples in Section **??** where a signal had segments with and without periodicity, such as in Figure **??**. This process can be useful to have an overview of the signal, or simply to summarize it.

8.1.3 Similarity Profiles and Summarization Examples

8.1.4 Information Retrieval in Occupational Data

The current information era, in conjunction with technological developments, is promoting a shift in the way industries and companies manage and perform their work. With the inclusion of sensing devices, dashboards and machine learning algorithms, many tasks can be better evaluated with direct quantitative measures. This can be made for machines to prevent breakdowns and optimize their performance, but can also be used for humans, specially to evaluate their occupational health, and prevent work-related disorders. This prevention implies a specialized intervention to perform changes in the workplace, the collaborator's training process and/or other organizational strategies. The process of deciding a need for intervention implies a previous risk evaluation of the collaborator's or team's routines. However, before the actual ergonomic risk evaluation, a considerable number of steps in preparing the data are needed.

As we mentioned in Section **??**, data preparation is an important part of any analysis or task that requires further deployment in supervised methods, namely the time-consuming segmentation and labelling processes. Hence, tools capable of segmenting time series to support and ease the labelling process of motion and posture data are highly valued and highly necessary for the risk analysis and the deployment of semi-supervised and/or supervised methods. In this case, motion data in real scenarios are even more complex since it is highly rich and diverse in behaviours. For instance, although cyclic and repetitive activities performed in industrial scenarios are mostly consistent from cycle to cycle, perfect conditions cannot be met all the time. Eventually, the working process can inadvertently be stopped or delayed. In addition, ergonomic risk assessment methods have to analyze each working cycle, which means that a previous segmentation of each one of these instances has to be made. Sub-activities that make the sequence of actions comprehended in the working cycle can also be sub-segmented for further recognition and association with risk measures. For instance, the *ergonomist* might have an interest in understanding

which actions from the working cycle have a significant association with a high or low risk measure, and adapt the working station with intervention strategies that could help prevent future disorders.

In this Section, we explore a few examples in using the SSM for information retrieval in occupational motion data of collaborators performing specific workstations. The data was recorded at the *Autoeuropa Volkswagen* production line in cooperation with this thesis (dataset 14 - see Section **??**). The information provided by the SSM supports the analysit in identifying the following events:

- **Active working periods:** *Active and non-active segments* indicate sub-sequences of the time series where the collaborator was performing the cyclic working tasks or not (e.g.: stop on the working line that leads to a pause in the working activity). With this information, we can focus the attention to active working periods, which are the segments of interest to perform risk analysis. For this, the *novelty function* can provide the moments of change between active and non-active regimes, as we will show further;

- **Working cycles:** Sub-sequences of highly similar sequences of movements that are being repeated in time. This type of segmentation pre-assumes that the time series under analysis will be mostly (or entirely) defined by the repetition of a motion. The illustrated examples show only tasks that are cyclic. The *similarity function* highlights moments with periodic behaviour.

- **Sub-cyclic segments:** Sub-cyclic segments inside each working cycle can be compared to evaluate how much it contributes to the risk score of a workstation. Segmenting the same portion of the working cycle over time can provide relevant comparative measures for decision making or intervention in the production line (e.g. it can be decided that the portion of the cycle has to be change to reduce the exposure to risk factors). The usage of a query-based mechanism with an example of the *subsequence* is used to demonstrate the findings.

8.1.4.1 Novelty and Periodic Segmentation in Occupational Data

In Figures 8.3 and 8.4 we present two different examples of data acquired while the collaborators. In Figure 8.3, collaborator 1 was performing the cyclic task 1, while in Figure 8.4, collaborator 2 was performing two different tasks (1 and 2). It is clear in both Figures that the SSM provides a visual feedback that highlights moments where the signal change (*blocks*), as well as where cycles occur (*paths*). In the first case, the collaborator waited (**P**) to start the task (**A**), up until the collaborator was interrupted several times by a stoppage on the production line to then continue his/her tasks. The novelty function is able to detect all of these main transitions, while the similarity function highlights where the working cycles (**C**) are.

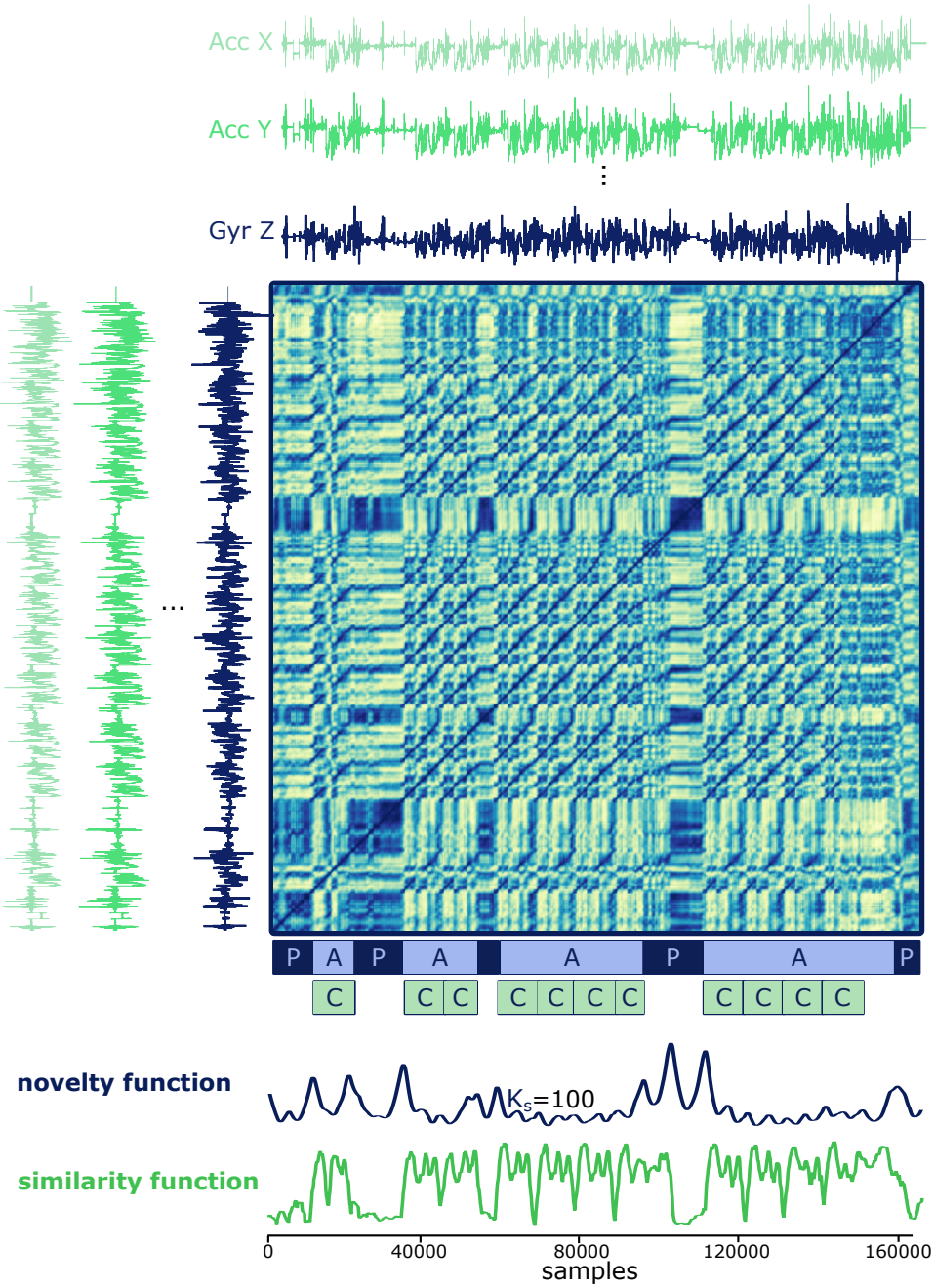


Figure 8.3: Illustrative example that shows how to use the SSM for information retrieval in motion data of occupational scenarios. A set of signals, presented as *Acc X, Y* and *Gyr Z* show a sample of the data used (Acc is the ACC and Gyr is the GYR data). The data from the wrist and elbow were used for this analysis. The segments are illustrated and labelled below the SSM with ***P*** - *interruption/pause in the production line*, ***A*** - *active working period*, and ***C*** - *working cycle*. The novelty and similarity functions are presented to show the match with the segmentation labels.

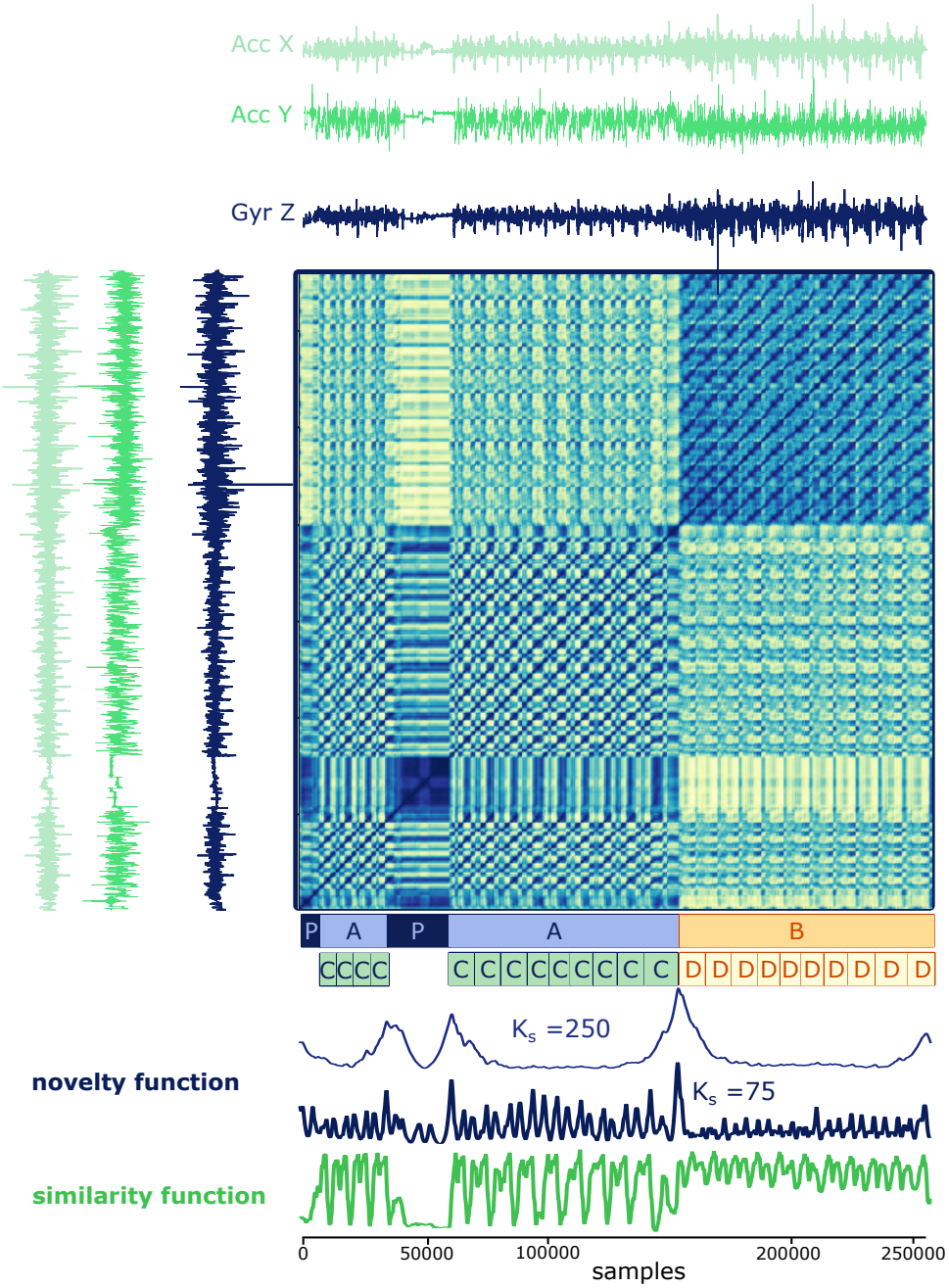


Figure 8.4: Illustrative example that shows how to use the SSM for information retrieval in motion data of occupational scenarios. A set of signals, presented as *Acc X, Y* and *Gyr Z* show a sample of the data used (Acc is the ACC and Gyr is the GYR data). The data from the wrist and elbow were used for this analysis. The segments are illustrated and labelled below the SSM with ***P*** *- interruption/pause in the production line,* ***A*** *- active working period for workstation 1* and ***C*** *- working cycle, and* ***B*** *- active working period for workstation 2 and* ***D*** *- working cycle*. The novelty and similarity functions are presented to show the match with the segmentation.

In the second example, collaborator 2 was performing the same workstation (**A**) as collaborator 1 but shifted to another workstation (**B**). The SSM indicates well when transitions occur, namely where the collaborator was interrupted while performing workstation **A**, and when shifting to workstation **B**. This is highlighted with the novelty function. The cycles are also visible with *paths*, for both workstations. The similarity function highlights well where the working cycles (**C** and **D**) happen. However, the search for working cycles should be performed after a first segmentation between homogeneous segments in order to have better results.

It is also relevant to mention that when the collaborator is in non-active periods (**P**), the *similarity function* has a very low value. As we mentioned in Section 6.3.2, a low similarity function value also corresponds to being very dissimilar to all the other *subsequences* of the signal. In this case, it makes sense, considering that the majority of the signal is related with cyclic activities, being **P** instances less frequent and therefore less similar with all the other *subsequences*. This happens in both examples.

The *novelty* and *similarity* functions were used with the purpose of detecting the listed events (see List 8.1.4) on Dataset 14 (see Section **??**). For the sake of brevity, the results for both item one and two of the list are displayed in Appendix B (see Tables B.8 and B.9, respectively). The first step involved detecting changes between active and non-active periods of the working activity, which was performed with the novelty function. The periods were described as active or non-active based on a threshold on the similarity function. The results show the ability to segment the signals with an overall F1-score of 0.96 (see Table B.8). After identifying the active periods, working cycles were segmented. From the 157 working cycles, 154 cycles were detected (see Table B.9).

8.1.4.2 Query-based Segmentation in Occupational Data

Another aspect of occupational risk evaluation in industrial scenarios is to compare the exposure to risk factors of sub-segments of the working cycle. This strategy could support professionals in identifying specific sequences of sub-activities that occur in-cycle, and search them over the entire set of working cycles for comparative purposes. In Figure 8.5 we show an example where the subsequence search matches exactly two distinctive in-cycle patterns (1 - blue; and 2 - green) of the working activity. Using these subsequences as a query template, we can match their onset over the successive cycles by finding the major valleys of the resulting distance functions (see Equation 6.8).

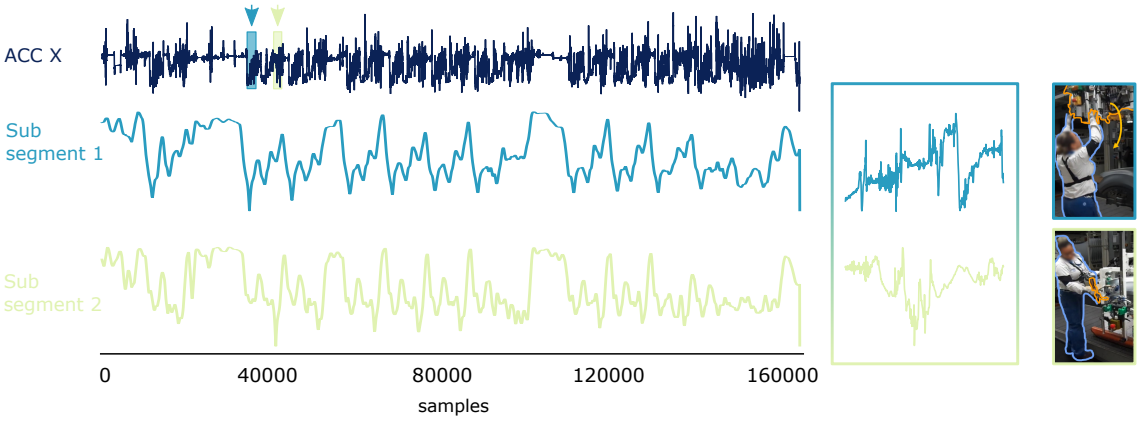


Figure 8.5: Illustrative example that shows how to use the SSM for information retrieval in motion data of occupational scenarios. The query-based search with a template is made on the SSM of the signal. This search procedure highlights where the query re-occurs on the signal with the distance functions (*subsegment 1* and *subsegment 2*). The templates are highlighted on the signal (Acc X) and on the right of the resulting subsegments. The small thumbnails next to the template illustrates the actions being performed on the cycle by the collaborator (highlighted in blue) and the tool being used (highlighted in orange). On the first subsegment, the collaborator was reaching over the head and pulling down the working tool, while on the second subsegment, the collaborator was screwing the pieces on the side of the car.

8.1.4.3 Summarization of Occupational Data

8.1.5 Discussion

8.2 Pattern Search with SSTS

In order to show the value of using a symbolic representation of the time series and search with regex on that representation, we provide six examples of how to use SSTS for pattern search on time series and compare the text complexity between the python implementation and the query used, with Halstead measures.

8.2.1 Illustrative Evaluation of SSTS in Various Application Scenarios

We will present six examples (three more detailed): the second and third examples consist of simple problems that require the detection of local maxima and minima. The sixth example is a more challenging task, which requires the usage of more sophisticated mechanisms of the regex module. The examples are listed as follows(*Example 1* is omitted since it was discussed in the previous section):

- Example 2 - Step Detection: Detect the instants in time when a right or left heel floor contact is achieved during a normal straight walk from the magnitude of an accelerometer signal. This information can be used to create a step detector based on accelerometer data. Since the sensor is located on the right pocket, global minima

will indicate the right heel contact and local minima will correspond to left heel contact;

- Example 3 - Segmentation of the systole of the ABP wave: The dicrotic notch corresponds to the physiological event of the aortic valve closure, which triggers the increase of the aortic pressure and signifies the end of the systolic phase. In this problem, it is asked to segment the systolic phase of the ABP wave;
- Example 4 - Electrocardiogram peak detector: Detect all the major peaks from an ECG record;
- Example 5 - Straight Line Trajectory Tracking: The automatic detection of trajectory features, such as straight walks and turns can be used to evaluate trajectory anomalies from a vector of cartesian coordinates in 2D-space;
- Example 6 - Since the first 5 s of the lifting exercise are unstable, the problem involves detecting the first stable lifting step, which occurs approximately 5 s after the beginning of the exercise from an accelerometer signal.

For the sake of brevity, only three of the examples are presented.

8.2.1.1 Example 2 - Step Detection in accelerometer signals

In 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 8.4.

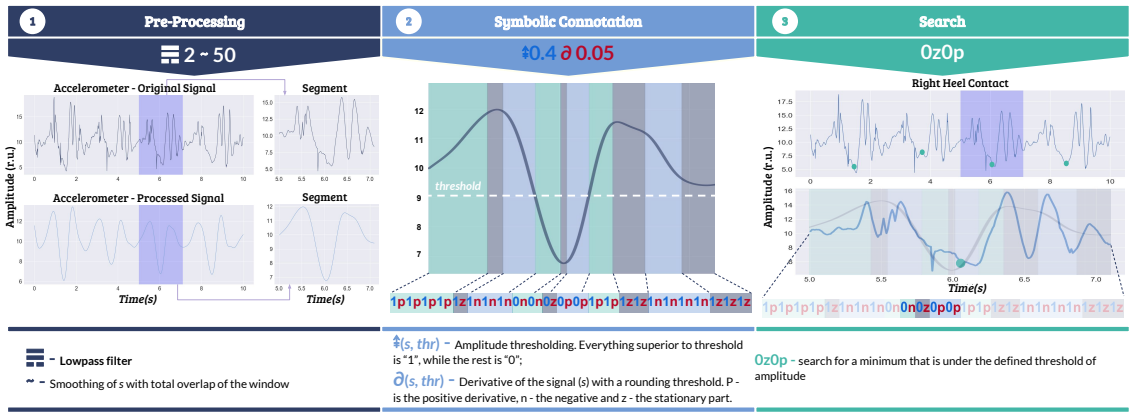


Figure 8.6: Example 2 - Solution pipeline of Step Detection example. At the bottom of the figure are summarized the operators and methods used in each step. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - A - Amplitude Comparison, Red - D1 - First Derivative). A dotted line is shown to represent the threshold level. Each color band in the signal corresponds to a specific primitive. A positive match is highlighted with green in the search step.

The signal is initially pre-processed to ease the identification of the subject's steps. This is achieved using a low pass filter (**LP**). The result is presented in the second image of the

first step in Fig. 8.6. A highlight is also present and delineates a segment of the signal that has a minimum peak, which corresponds to right heel contact. The string representation of this segment is depicted in the second step.

The string is a sequence of primitives composed of two connotation methods (**A** in blue, **D1** 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 than 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. 8.6.

8.2.1.2 Example 3 - Dicrotic notch detection in ABP signals

The ABP waves are morphologically represented by a high positive slope that corresponds to the systolic uptake and ends at the peak of the systolic pressure. After this behavior, follows the systolic decline, which ends with the aortic valve closure, named the *dicrotic notch* in the signal representation [112].

These types of signals are commonly affected by low-frequency noise that modulates the signal and is contaminated with high-frequency noise of low amplitude. The typical procedure is to bandpass the ABP signal to remove both types of noise. Fig. 8.7 depicts how a band pass filter that cuts frequencies under 1 Hz and above 20 Hz (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.

In the symbolic connotation step, the reasoning follows the signal morphological description and uses two methods for the symbolic representation. The first, AD (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 to distinguish between the rising slope that occurs at 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 ABP 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 ABP wave, the regular expression has to start with the first `1p` primitive and end

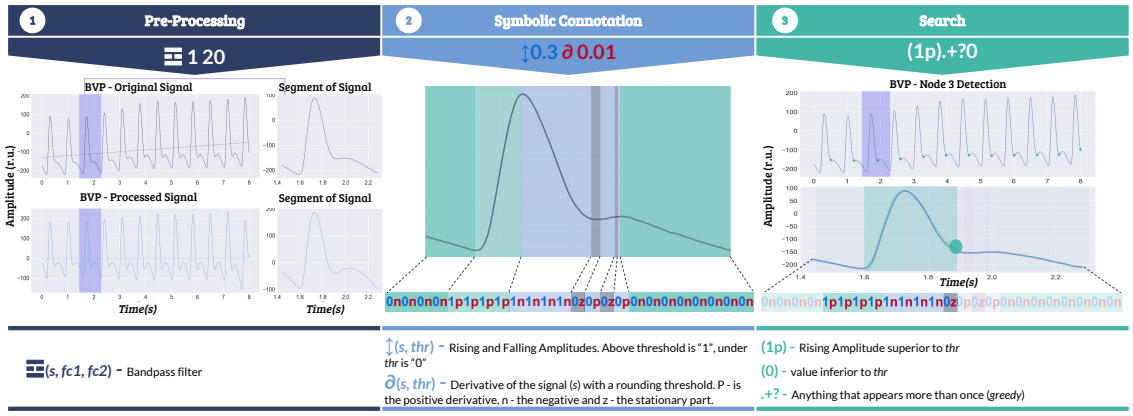


Figure 8.7: Example 3 - Solution pipeline of Dicrotic notch detection example. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between methods is represented with colors (Blue - AD, Red - D1). For each color band in the signal, there is a corresponding primitive. The match is highlighted with green in the search step.

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.

8.2.1.3 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, and (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.

Fig. 8.8 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 AD (represented in blue) turns all sample values of the first signal that are higher than the threshold level into `1`, while the remaining samples are converted to `0`. The second set is applied to the

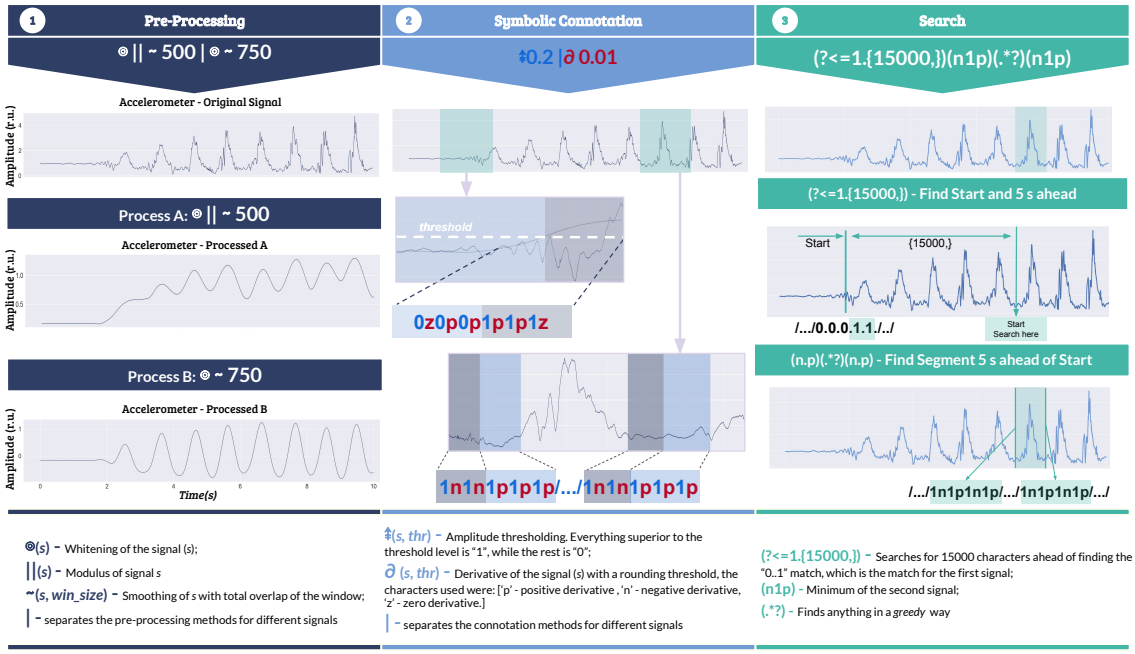


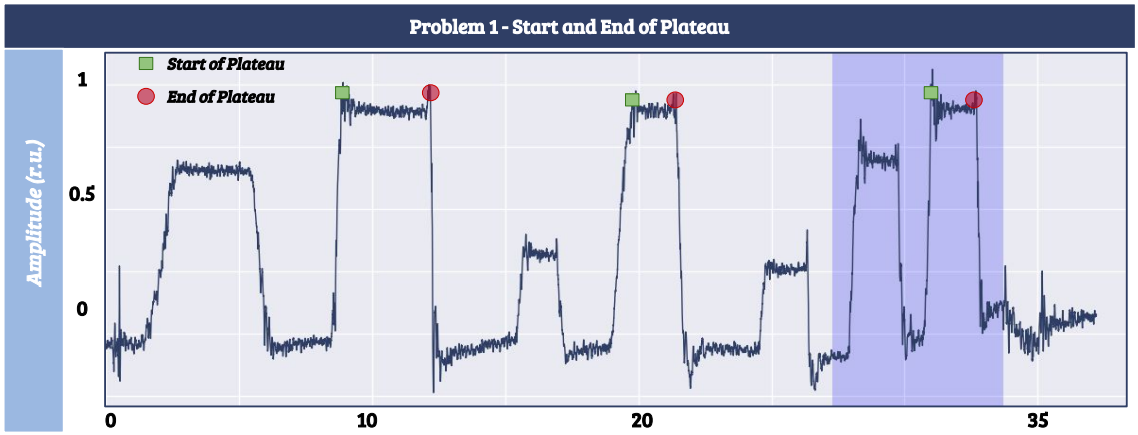
Figure 8.8: Example 6 - Solution pipeline of Stable Lifting Detection. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - **AD**, Red - **D1**). The threshold level is identified with a white dotted line. Each color band in the signal corresponds to a specific primitive. The match is highlighted with green in the "Search" step.

second signal and uses the derivative of the signal D1(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`).

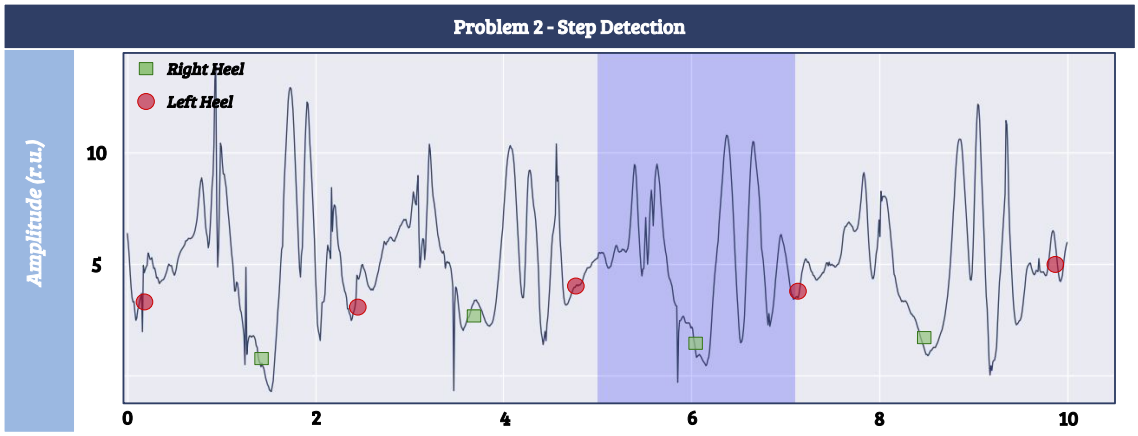
The regular expression written to solve the example is `(?<=1.{15000,})(n1p)(.*?)(n1p)`. Decomposing this expression in the two steps of the example results in: (1) `(?<=1.15000,)` and (2)`(n1p)(.*?)(n1p)`. In (1), a *lookbehind* operator (recall rules from Table 3.1 in Section 3.7.3) is used, therefore, 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.

8.2.1.4 Summary of all examples

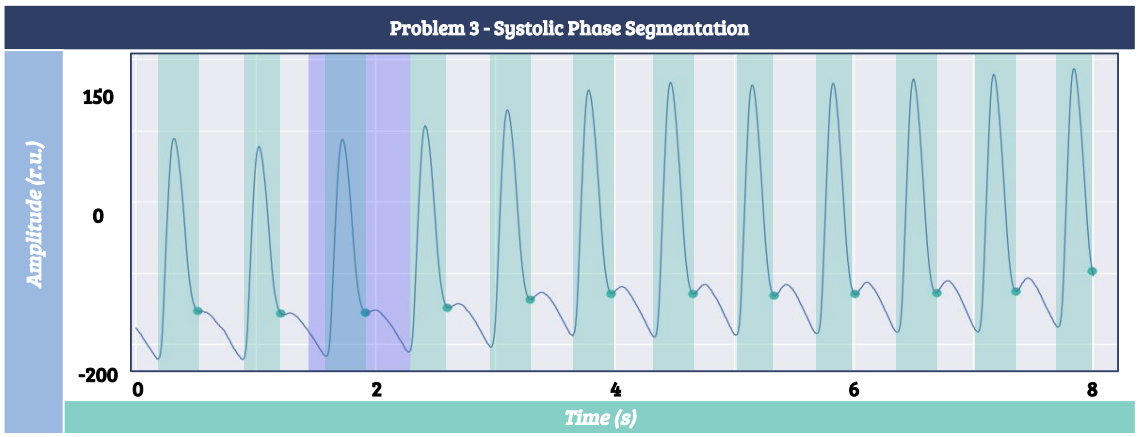
At this point, introductory examples have been explained and can be used to understand the workability of the tool. Table 8.4 and Fig. 8.9 summarize the example's resolution for



(a)

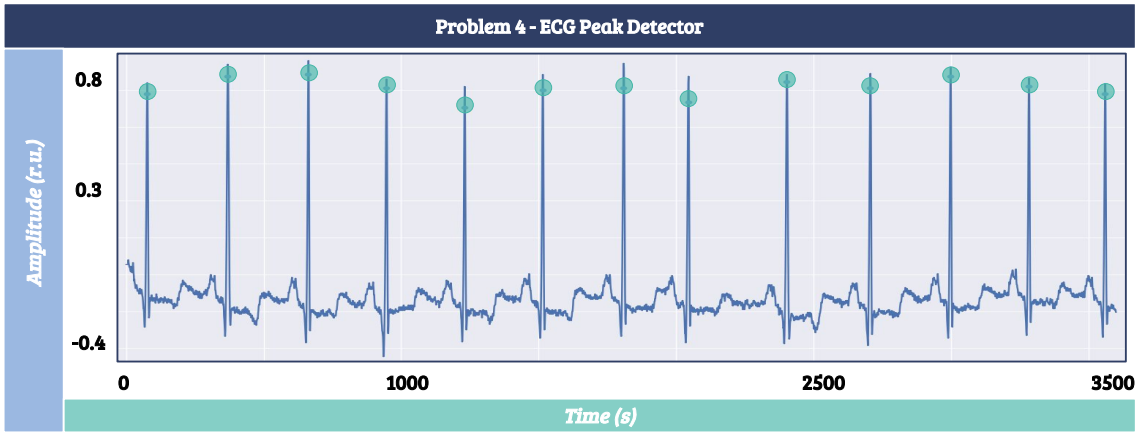


(b)

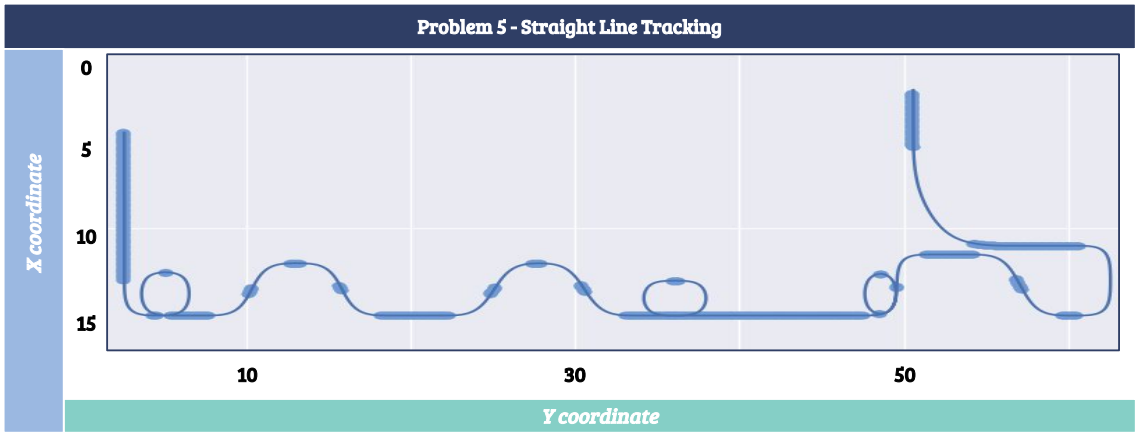


(c)

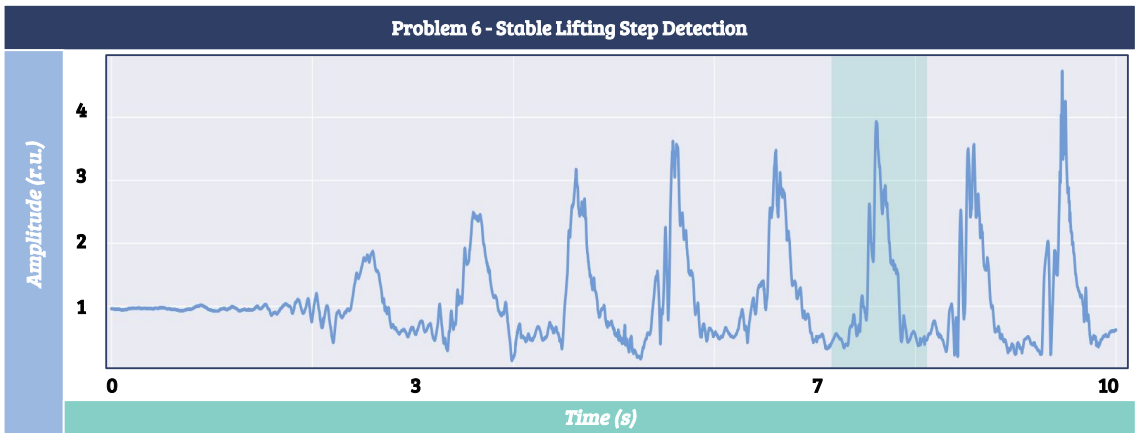
Figure 8.9: Summary of the resolution of the problems 1 (), 2 (Step Detection), and 3 (Segmentation of the systole on the ABP wave), listed in the previous Section with the SSTS.



(d)



(e)



(f)

Figure 8.9: (Continuation of the previous figure). Summary of the resolution of the problems 4 (Electrocardiogram peak detector), 5 (Straight line trajectory tracking), and 6 (Lifting exercise), listed in the previous Section with the SSTS.

Table 8.4: Summary of the SSTS queries used to solve the six illustrative problems.

Example	Fig.	Pre-Processing	Connotation	Search
Start of Plateau	8.9.a	S 500	A 0.8 D1 0.05	`p1n`
End of Plateau	8.9.a	S 500	A 0.8 D1 0.05	`z1n`
Step Detection (Left)	8.9.b	LP 2 Sm 50	A 0.4 D1 0.05	`0z0p`
Step Detection (Right)	8.9.b	LP 2 Sm 50	A 0.3 D1 0.05	`1z1p`
Dicrotic Notch	8.9.c	BP 1 20	A 0.3 D1 0.01	`(1p).+?0`
Electrocardiogram Peak Detector	8.9.d	BP 5 50	D1 0.01	`pn`
Straight Line Tracking	8.9.e	none	D2 0.05	`z*?`
Stable Lifting Detection	8.9.f	Mag Abs Sm 1000 | Mag Sm 750	A 0.2 | A -0.2 D1 0.01	`(?<=1.15000,)(n.p)(.*?)(n.p)`

each step of the pipeline. Additional examples are also included and can be found in the GitHub repository of the tool[1].

8.2.2 Measure of Expressiveness

In order to evaluate parameters of expressiveness with SSTS, we evaluated the legibility and difficulty in generating the solution for the corresponding task. In the programming field, *Halstead* measures are typically used for this purpose. These measures describe the complexity of the script directly from the source code, based in a set of metrics calculated with the number of distinct oprt and oprd, and the Toprt and Toprd (recall Section 3.8.3).

The complexity evaluation will be performed to compare the script generated by the *classical* approach with the *syntactic* approach. Concerning the *classical* approach, the list of operators and operands are selected from the *operator* module from python [47], which includes the list of arithmetic, logical, comparison, bitwise, assignment, and special operators; and, the functions that are used in the resolution process will be set as operators and their corresponding arguments as operands. For example, consider this code line written in python:

$$pks = peakdelta(s, delta = np.percentile(s, 70) - np.percentile(s, 30)) \tag{8.1}$$

- **Total List of Operators**: *'peakdelta'*, *'percentile'*, *'−'* and *'percentile'*
- **Total List of Operands**: *'s'*, *'delta=np.percentile(s, 70) - np.percentile(s, 30)'*, *'s'*, *'70'*, *'s'*, *'np.percentile(s,30)'*, *'np.percentile(s,70)'* and *'30'*
- **List of Distinct Operators**: *'peakdelta'*, *'percentile'* and *'−'*
- **List of Distinct Operands**: *'s'*, *'delta=np.percentile(s, 70) - np.percentile(s, 30)'*, *'np.percentile(s, 70)'*, *'np.percentile(s, 30)'*, *'70'* and *'30'*

In the second case, both **connotation** step and **search** procedure will be inspected for the **difficulty** measures. Regarding the **connotation** step, the evaluation will be made similarly to the previous method for inspecting functions, in which the function is set as an

[1] https://github.com/novabiosignals/SyntacticSeachOnTimeSeries

operator and its corresponding arguments as operands. For the **search** procedure, the set of instructions typically used in regex (except the "." character) is defined as operators[71]. These can be revisited in the second section. The remaining characters of the regular expression string, except the parenthesis, will be set as operands. In this are included the "." operator that matches any character, and the "\d" and "\w" that matches all digits and alphabetical characters respectively. Regarding the parenthesis, when occurring ("{}" or "[]"), will be set as one operator. For example:

$$Connotation : Sm\ 500\ A\ 0.8\ D1\ 0.05$$
$$Search : \texttt{(z1n).*?}$$

- **Total List of Operators**: *'Sm'*, *'A'*, *'D1'*, *'( )'* and *'*?'*
- **Total List of Operands**: *'500'*, *'0.8'*, *'0.05'*, *'z'*, *'1'*, *'n'* and *'.'*
- **List of Distinct Operators**: *'Sm'*, *'A'*, *'D1'*, *'( )'* and *'*?'*
- **List of Distinct Operands**: *'500'*, *'0.8'*, *'0.05'*, *'z'*, *'1'*, *'n'* and *'.'*

Table 8.5: Evaluation of the complexity in the resolution of examples with the *syntactic* approach and with the *classical* approach. Voc - Vocabulary, Lgth - Length, CL - Calculated Length, V - volume, D - difficulty, E - effort.

Example	Syntactic Resolution						Classical Resolution					
	Voc	***Lgth***	***CL***	***V***	***D***	***E***	***Voc***	***Lgth***	***CL***	***V***	***D***	***E***
Start/End of Plateau	5.0	5.0	8.0	11.6	1.5	**17.41**	12.0	13.0	33.3	46.6	2.5	**116.5**
Step Detection (Left & Right Heel)	5.0	6.0	8.0	13.9	1.75	**24.38**	19.0	22.0	63.6	93.5	4.2	**388.2**
Dicrotic Notch	7.0	7.0	13.6	19.7	2.0	**39.3**	21.0	25.0	73.0	109.8	4.7	**517.7**
Electrocardiogram Peak Detection	2.0	2.0	2.0	2.0	1.0	**2.0**	9.0	12.0	20.3	38.0	3.0	**114.0**
Straight Line Tracking	2.0	2.0	0.0	2.0	1.5	**3.0**	25.0	33.0	93.5	153.2	5.3	**811.3**
Stable Lifting Detection	10.0	18.0	24.4	59.8	3.6	**217.8**	22.0	23.0	77.3	102.6	5.0	**512.8**

Table 8.5 presents the performance measurements for both methods of solving the demonstrated examples. For the evaluation, the pre-processing step was omitted, which means that only the regex is used to compute the difficulty measures for the *syntactic* approach, and only the script after the pre-processing steps is used to compute the difficulty measures for the *classical* approach.

8.2.3 Discussion

The previous section demonstrated the potential of the proposed framework in delivering a more expressive methodology of solving query search tasks in time series. For the

complete group of examples presented, all the steps involved were written in a symbolic manner. This fact eased the way in which time series are interpreted and how the final search is achieved.

The results from the *Halstead* measurements summarized in Table 8.5 demonstrate that, in general, the complexity is decreased using the SSTS approach in comparison with the classical solution. In the circumstances in which the difference is not significant, such as the *Example 6* (Stable Lifting Detection), the complexity measurements increase significantly. This fact suggests that for more complicated problems, the complexity of the regex increases.

8.2.3.1 Symbolic series generation

Overall, most of the symbolic series that are generated by the symbolic connotation step are simple to read and interpret. Nevertheless, the complexity of the series linearly increases with the number of operators used to perform the symbolic connotation. Consequently, for problems that have inherent and higher complexity, and require the use of several connotation groups, it is expected that the legibility of the symbolic sequence will also be reduced. *Example 6* constitutes an example of the mentioned drawback, as it involves two distinct resolution steps and revealed to be much more complicated to understand without any previous insights.

8.2.3.2 Required background knowledge

Each step of the SSTS pipeline has an important role in solving the examples, especially the two initial steps (pre-processing and symbolic connotation). Both steps aim to simplify the signal, such that the search procedure can be composed as easily as possible. This approach aims to prepare the signal so that a sequence of symbols can be used to simplify the search step. Therefore, choosing an adequate set of pre-processing methods and proper coordination with the symbolic connotation methods will reduce the search task complexity, both in the writing and interpretation processes.

SSTS users' should have background knowledge in both signal processing techniques and regex. In fact, without prior consolidation of these topics, the user might have an additional effort in generating a solution to solve the required problems. Firstly, for circumstances in which the user experiences difficulty in using such methods, the pre-processing and the symbolic connotation steps will not be so intuitive. Secondly, for in case the user is not familiarized with regex, it will be hard to generate appropriate expressions to search for patterns on the symbolic time series, especially for more difficult problems, which require the use of advanced operators.

As a future objective, we expect to increase the expressiveness of the tool and reduce the required knowledge to accomplish signal processing tasks and regex queries. In order to achieve a more simplified pipeline, a set of pre-processing and symbolic connotation templates for specific types of data (e.g. electrocardiogram) should be available to the

user. Furthermore, the search step could be also simplified using meta-regex. In addition, a flexibility in the search process should be considered. regex are powerful tools, but lack flexibility, in the sense that the pattern is or not matched by the regex. Instead, a score function based on the query should be considered to make a flexible search and provide an error margin to the user.

We expect SSTS to have a broad range of applications that can extend to the query-based search problematic. Other topics in time series data mining, such as *signal generation*, *segmentation*, *feature extraction*, *labelling* and *classification*, could benefit from this reasoning. In this thesis, only the classification topic was explored. In the next Section, we present our results.

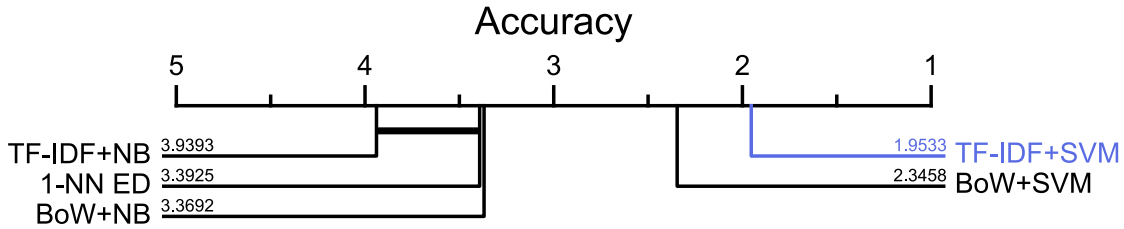
8.3 Time Series Classification with HeaЯTS

This section is evaluated the ability to perform time series classification with the proposed HeaRTS 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 SVM classifiers with either a BoW or a TF-idf 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.

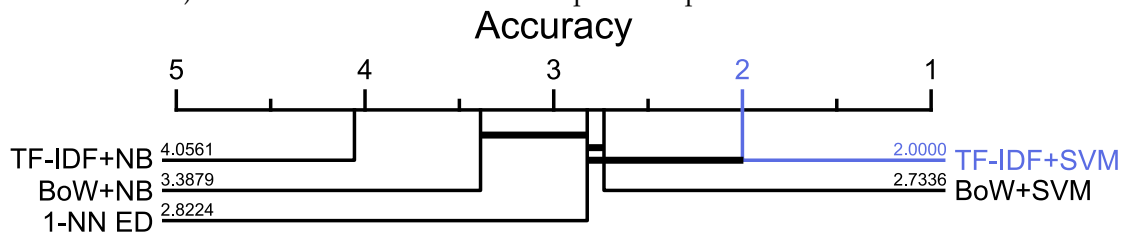
In addition to these experiments, we performed an individual analysis of several use-cases to understand which is the interpretability and readability of the data based on this linguistic translation. This study should tell us if the words extracted are valuable to indicate the *subsequences* of the time series that are relevant to identify it and distinguish it from others.

8.3.1 Classification Performance

Figure 7.8 shows us how vectorized documents can be compared with the cosine distance to distinguish time series. Simply by using part of the SSTS queries, it is possible to generate TF-idf 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). Therefore, to validate the ability of the proposed method to perform time series classification, it was computed on all datasets of the UCR Benchmark and its performance compared with the standard 1-NN ED. The process was divided into two stages. First, the parameters to compute HeaRTS were optimized on the standard train



(a) Critical distance diagram for the methods that rely in a textual representation of the time series (BoW and TF-idf) and euclidean distance for the optimized parameter search.



(b) Critical distance diagram for the methods that rely in a textual representation of the time series (BoW and TF-idf) and euclidean distance for the 10 fold cross-validation process.

and test sets from the *pyts* distribution. The optimized performance was compared with the 1-NN ED method. Second, we used the optimized method on 10-fold cross-validation and compared the average accuracies with the 1-NN ED (the implementation from *pyts* was used to compute the results of the 1-NN ED method).

As mentioned on Chapter **??**, the method translates the time series into text with SSTS queries from Table 7.3. These have 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 BoW and TF-idf models can be built with different *n-gram* sizes. These three parameters were optimized with a grid search method over a range of values:

$$\begin{aligned} w_s &\in [1, 10, 25, 50, 100, 250] \\ thr &\in [0.001, 0.01, 0.05, 0.1] \\ n-gram &\in [1, 2, 3, 4, 5, 6] \end{aligned} \tag{8.2}$$

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

As can be seen, the strategies that rely on a SVM classifier are more robust. The TF-idf turns out to not be reliable when used with the Bayesian Classifier, which was expected since this type of classifier is purely probabilistic, which is why it works better with the BoW. On the other end, the TF-idf model works well with SVM classifiers, being the best method studied. We then compared it in more detail with the 1-NN ED classifier, as

presented in Figures 8.10a, **??** and 8.12.

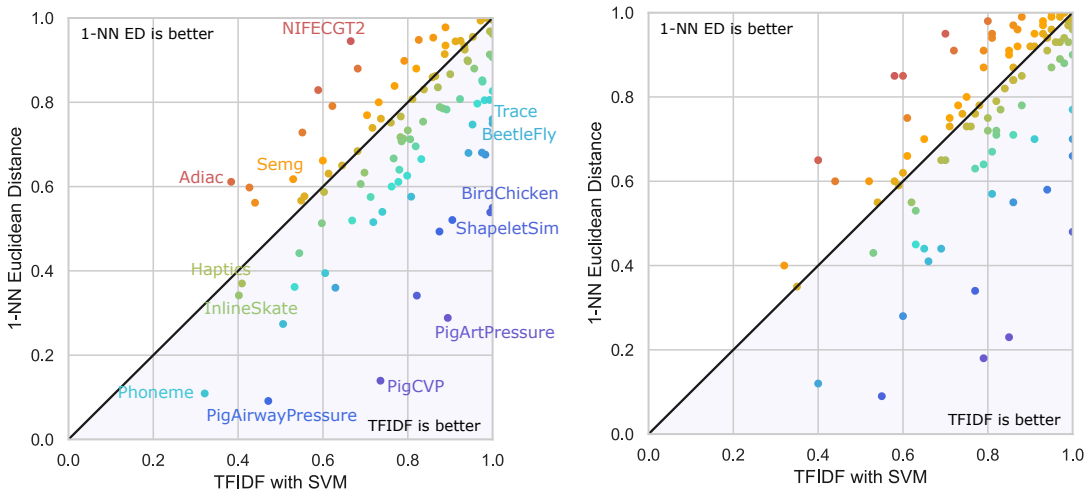


Figure 8.11: right) Classification accuracies for the Bag of Synthatic Patterns in comparison with the euclidean distance combined with 1-NN classifier (1-NN ED) with standard train and test; left) with a 10 fold cross validation.

The average accuracy for the total 107 datasets is 79% for HeaRTS and 71% for the 1-NN ED. Special cases are highlighted on Figure 8.11.right. 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.

The same conclusions are confirmed on the 10 fold cross-validation (Figure 8.11.left. Surprisingly, the proposed solution with a TF-idf and SVM classifier is very consistent with the previous results, even increasing to 80% the average accuracy for all datasets. What also is surprising is the improvement of the 1-NN ED method, which increased its accuracy to 75% on average. The critical distance for this performance is now less evident (Figure **??**, but the proposed solution is the overall best method. Even when compared to the other methods, it is overall better in most datasets of the benchmark. The only method that competes is the BoW with an SVM classifier, which has a very similar performance (upright scheme of Figure 8.12.

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.

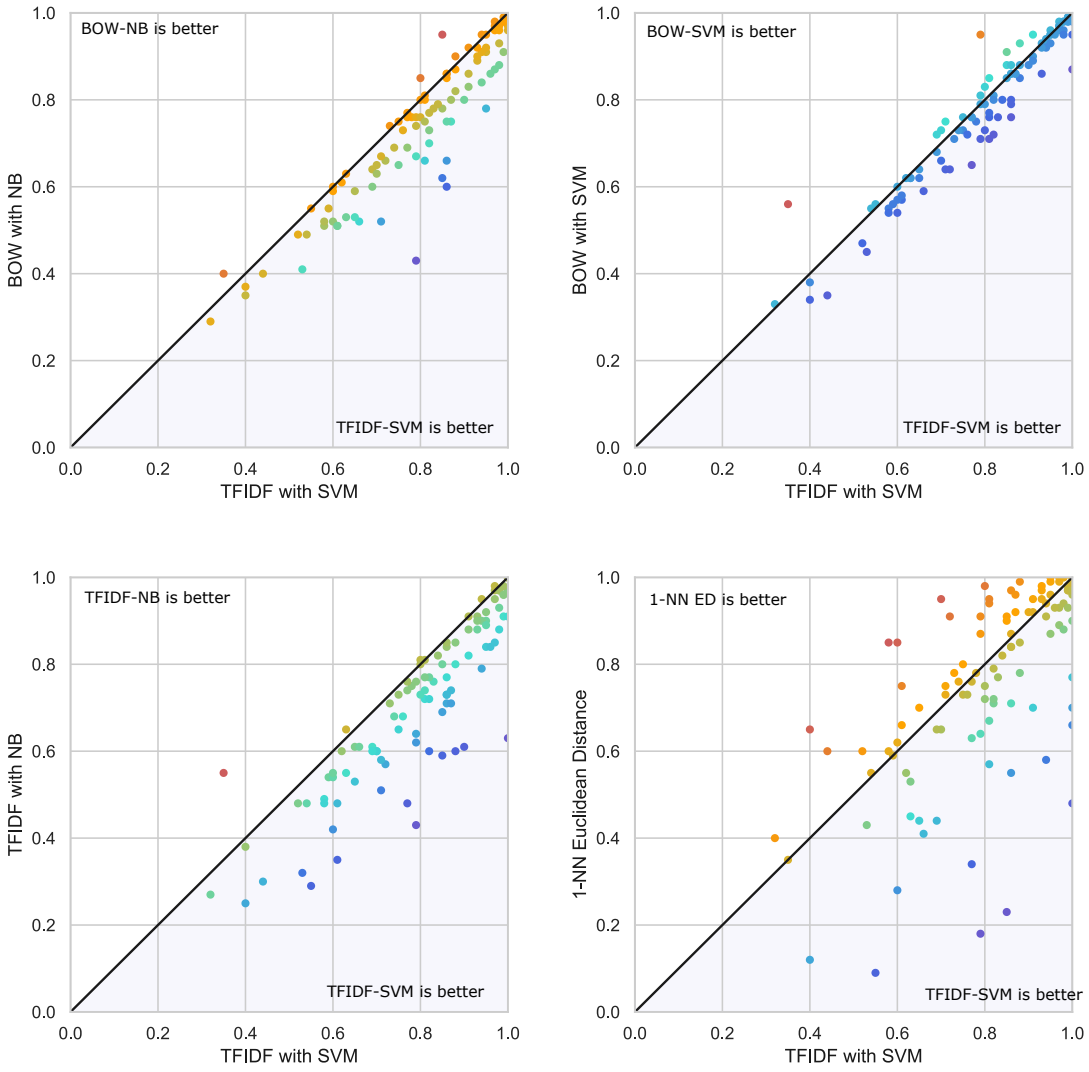


Figure 8.12: Classification accuracies for all experimented methods versus the accuracies of the TF-idf method with an SVM classifier.

8.3.2 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 TF-idf 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 (8.13, 8.14 and 8.15) are displayed showing the highlighted areas based on the TF-idf weights.

On Figure 8.13 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`

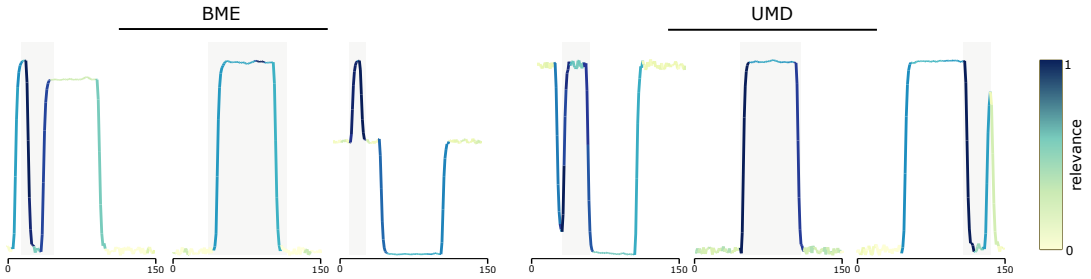


Figure 8.13: Interpretable results with highlighted shapes for the *UMD* and *BME* datasets. The *UMD* dataset was classified with with an accuracy of 99.3%, computed with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$. The *BME* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$

`followed by a negative plateau`. The second class has simply a `plateau`. The main difference between all those classes is the peaks and what follows them, 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.

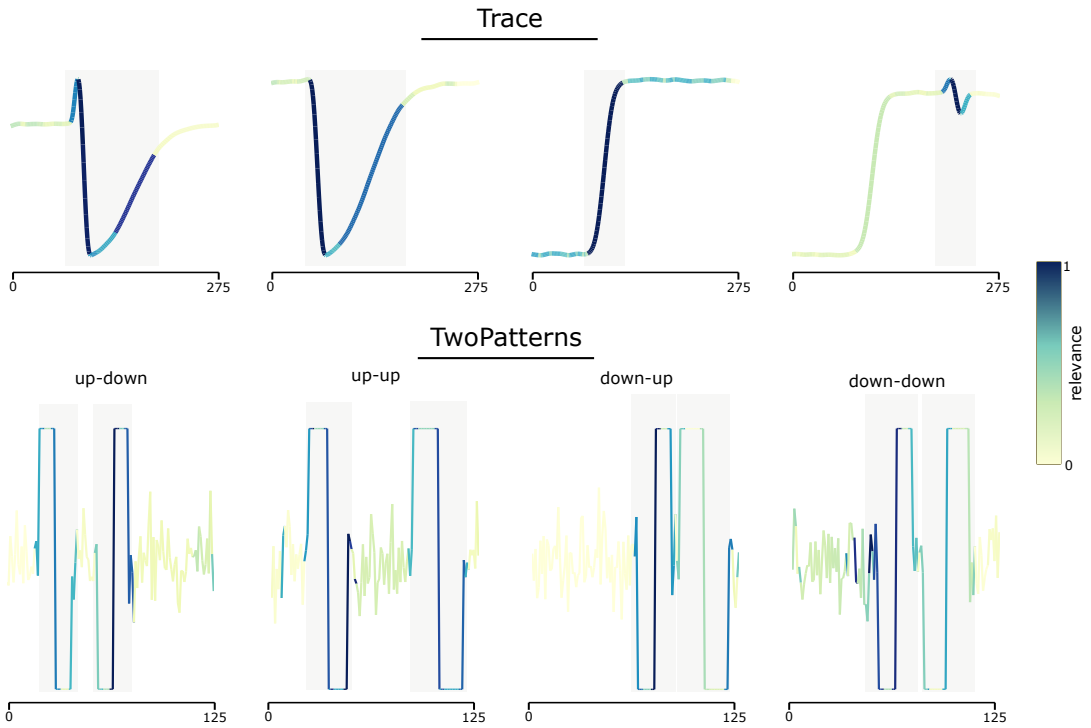


Figure 8.14: Interpretable results with highlighted shapes for the *Trace* and *TwoPatterns* datasets. The *Trace* dataset was classified with with an accuracy of 100.0%, computed with a $w_s = 25$, $thr = 0.05$ and a $1 - gram$. The *TwoPatterns* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.1$ and a $5 - gram$

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 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. Classes 3 and 4 have notoriously different valued elements. While class 3 has the rising phase as the relevant shape, class 4 has a `small peak followed 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.

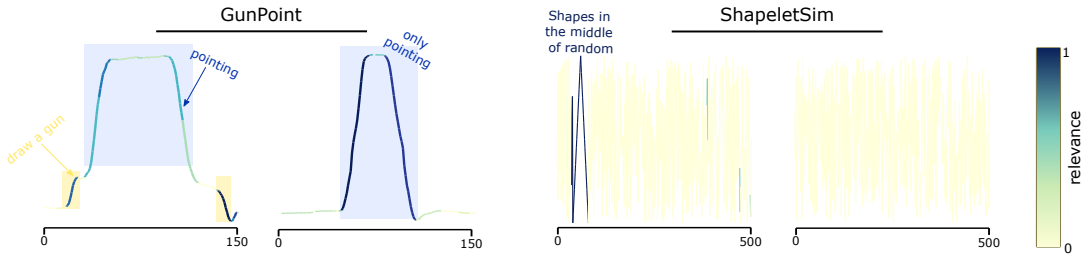


Figure 8.15: Interpretable results with highlighted shapes for the *Gunpoint* and *ShapeletSim* datasets. The GunPoint dataset was classified with with an accuracy of 99.0%, computed with a $w_s = 10$, $thr = 0.05$ and a $2-gram$. The *ShapeletSim* was classified with an accuracy of 99.4% with a $w_s = 1$, $thr = 0.05$ 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 subject 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`.

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 can highlight the representative shape of the signal.

8.3.3 Discussion

In this section, we wanted to answer two main research questions: (1) can we solve time series classification tasks based on their 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 use traditional strategies from the text mining domain. Not only is it possible to compare the vectors of time series document with the cosine distance (as we have seen on Figure 7.8, but BoW or TF-idf models can be used with a linear SVM

to solve classification tasks. The provided results can even be improved considering that the linear SVM classifier was not optimized.

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 parts of the signal are more relevant, 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. In these cases, it is hard to truly find meaningful differences and explain why these classes are different, especially with words. This limits the further interpretability potential of this method because the differences might not be perceptible by the naked eye. On one end, the performance can be improved using a richer and correct set of textual descriptors. We mean richer because the differentiation is as good as the description performed, and we mean correct because the translation process from the numerical domain to the text domain is not error-proof, which means that a mistranslation may occur and make the results worse. On the other end, the words used might not be capable of expressing the differences between classes. Many cases with high accuracies can be highlighted, but their interpretability falls short. For instance, for the dataset *BeetleFly*, a binary classification problem. The method has 100% accuracy but does not show expressively the difference between classes.

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 SSTS queries we can make a limited description that works well in simple signals described mostly by their derivative dynamics. The keywords extracted for these time series still have limited readability being even harder in the complex. However, if more expressive queries are used to make a more robust translation, the keywords extracted can be more valuable and relatable to the time series we are looking at. Therefore, a more diverse set of queries should be used to make possible the extraction of more relevant keywords. At some point, a reduction and selection of the most relevant keywords should be made, as it is performed with feature selection and dimensionality reduction.

8.4 Pattern Search with QuoTS

In this Section, we demonstrate the potential utility of QuoTS to search for relevant subsequences in real datasets. Table 7.4 from the previous Chapter highlights the keywords and operators if needed to follow these examples. This method, in contrast with SSTS, uses features and not a symbolic representation of the time series.

In order to show evidence that QuoTS is useful in a multitude of scenarios, we present several examples of its application to find relevant patterns on real domain time series. The section will start by presenting how well it matches hand gestures; then show its applicability in searching for relevant patterns in several problems and use-cases; and

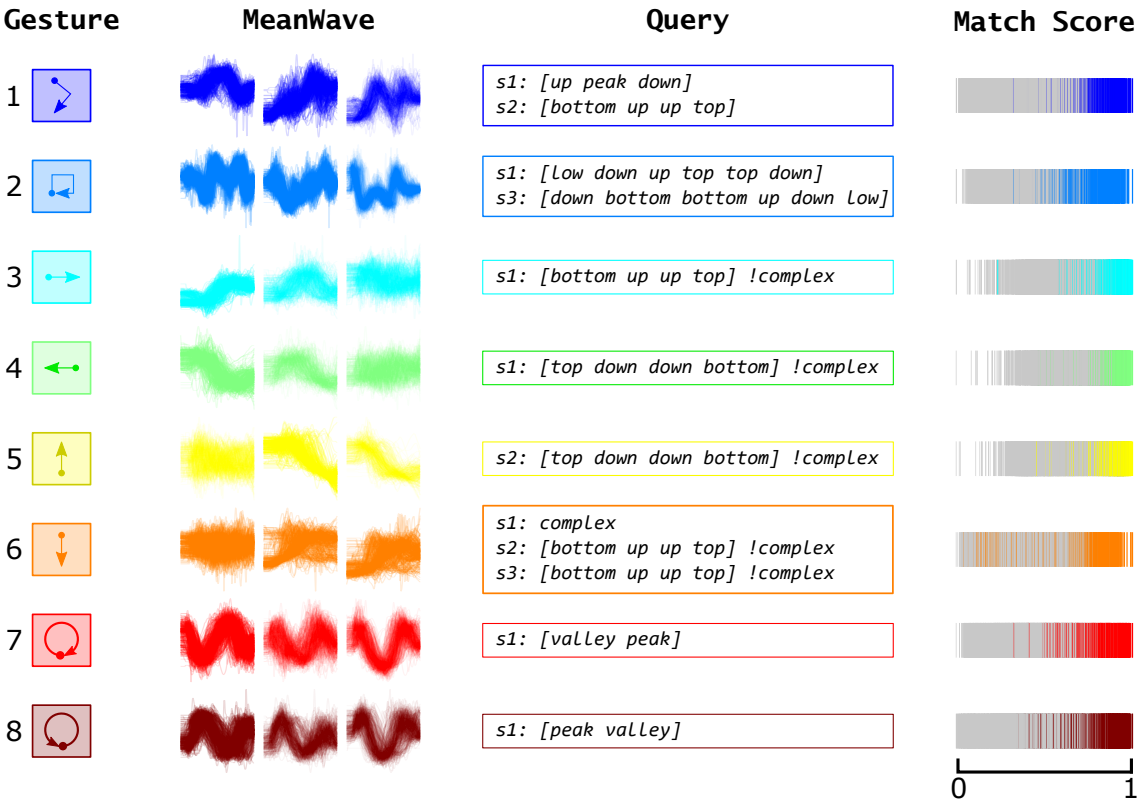


Figure 8.16: *UWaveGestureLibrary* subsequence matches with QuoTS. Column-wise are showed the gesture class, the corresponding mean wave for all subsequences, the query used to match the subsequence class and the corresponding match score for all subsequences, highlighting with the color of the specific class.

finally, we present the ability to include additional words based on existing patterns, with the example of *puppeteering* a toy car model.

8.4.1 *QuoTS* Matches Gestures

We start by demonstrating the ability of QuoTS to sort how well individual shapes match a written query. For this, we used the *UWaveGestureLibrary* dataset from the *UCRArchive* as a proxy [92], 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 [35, 137]. 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 8.16 and quantitatively (Table 8.6). 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

	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 8.6: Results for the top 10 and top 100 sorted gestures classes (G) when using the queries from Figure 8.16.

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 of 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 8.6. 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 interested in examining at least the top 10 results. Nevertheless, the reader will appreciate that even if we consider the top 100 matches, QuoTS still achieved impressive results.

Although we simply wanted to demonstrate that with a set of meaningful words we can 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 of 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 especially 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 [92].

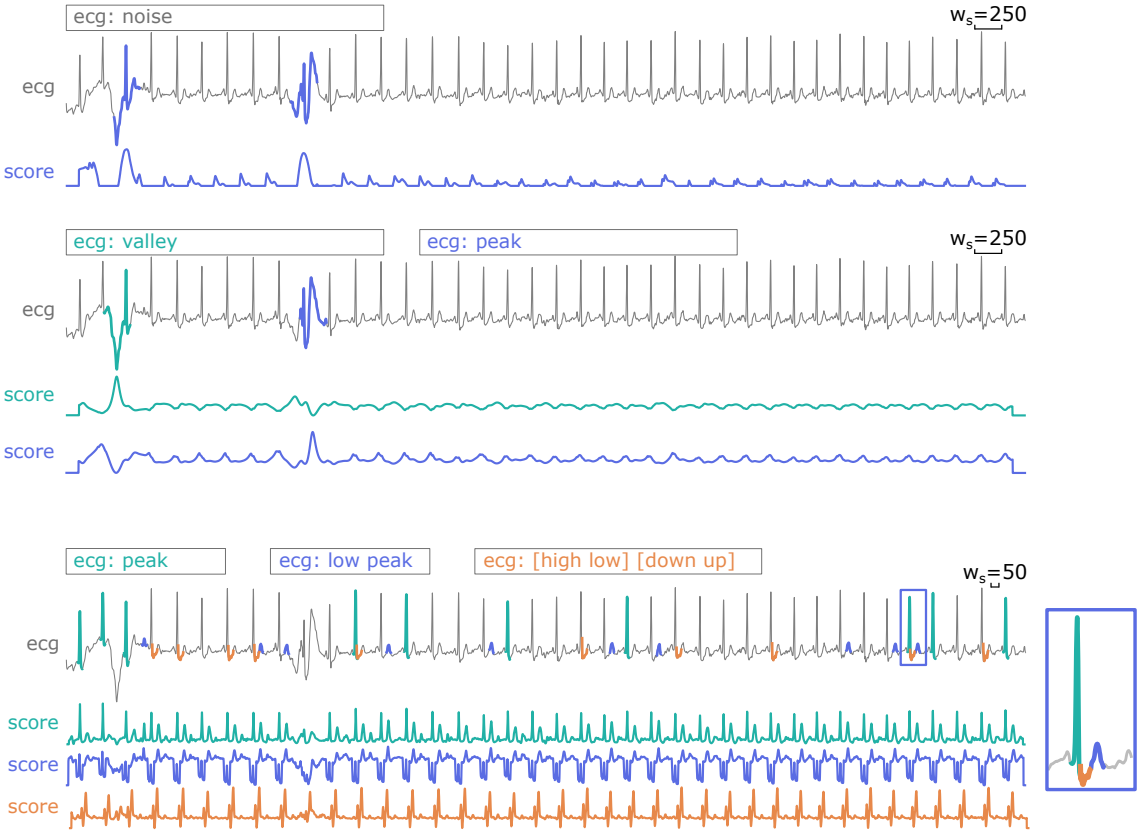


Figure 8.17: ECG use case to identify noisy sections, as well as specific segments of the ECG pattern. The queries are written in text boxes. On the side, an example of a match is showed as a larger pattern.

8.4.2 *QuoTS* in Selected Use Cases

In this section, we present several examples of where QuoTS can be used in continuous real data. We start with an ECG signal that has motion artifacts.

As a start, we were searching for the motion artifact segment that appears in two areas of the signal. This artifact can be detected in two ways, as we are showing in Figure 8.17. The first would be to use the keyword `noise`, which searches for parts of the signal that have a higher difference from a moving average of the signal. In this case, the higher difference occurs when the motion artifacts appear. We could also search for each of these motion artifact sequences simply by pure visual intuition of their shape. For instance, the first motion artifact looks like a `valley`, while the second looks like a `peak`, which ends up matching the desired segments. This process was made by looking for subsequences with 250 samples.

On the ECG signal there are also other segments of interest, namely from the *PQRS* complex. In this case, we searched for the *R* `peak`, the *S* `valley` and the *T* `peak`. The first subsequence was simply matched by searching for the keyword `peak`, with a window size of 50 samples, which contrasts with the 250 samples from the previous example. The keyword matched well the highest peak, but other peaks were present with lower

amplitudes, such as the *T* `peak`. This one was matched with the query `low peak`. In between these two subsequences, there is the *S* `valley`. In this case, we show the usage of the special `followed by` operator. The query `[high low] [down up]` indicates that the first half of the searched subsequence should have a *high change in amplitude going down* and end with a *low change in amplitude going up*. This is exactly what is matched, as highlighted and indicated by the corresponding *score* function. To be clear, the colored highlighted segments represent the 10-most relevant matches for the written query. The *scoring* functions show that the other matches would be found as well.

The ECG signal is a perfect example of how a text-based search can help find relevant occurrences, as there are specific medical conditions that are represented by variations in the original shape of the ECG signal. The next Figure (8.18) shows several examples of the detection of three different types of arrhythmia from the *St Petersburg INCART 12-lead Arrhythmia Database* (Physionet) [55].

The ground truth follows the annotations present on the original files. Three types of annotations were found, associated with different types of arrhythmia events (A -  , F -  and V - ). Using QuoTS, we can search for these occurrences in the presented segment. The first event is characterized by having a normal beat irregularly before it was supposed to. It is then possible to find two ECG beats in a shorter window. This means that if we search for two main peaks inside a short window, separated by anything (`.`) `III: [. peak . peak .]` (in this case 500 samples on the lead III), it should be possible to find where these events occur. We displayed the 5-top subsequences that match the query and the corresponding scoring function. Four out of the five events are highlighted, but the scoring function indicates that the "missing" subsequence has a high match value as well. The wrongly identified event fits the description as well, being a different type of arrhythmia, but with a different shape (type F). This shape is normally a high change in amplitude, therefore if we state that we reject any high amplitude change subsequence, we should be able to match the five desired subsequences. Adding `!high` to the previous query (`III: [. peak . peak .] !high`) leads to correctly matching all the desired subsequences. To be clear, we chose lead III without any specific purpose, being possible to use the other leads for the same purpose. What matters is that the desired subsequence has a particular characteristic that differentiates it from the rest.

The second example corresponds to the arrhythmia of type F, characterized by a high change in the amplitude of the ECG with an irregular beat. In this case, we can match this event with the simple query `AVL: [high peak]`. Other events are matched but have a much lower amplitude. The other events matched are from the type V, which does not present a long flat section after the irregular beat. In this case, the lead AVR shows a very specific difference from the other two types of arrhythmia, which is the fact that the beat occurs on top, deviating from the middle of the signal. The shape continues to be a `valley` but on `top` (`AVR: valley top`). It is relevant to point out that the original annotations (ground truth) do not include one of the highlighted subsequences. Although not confirmed with a specialist, the shape of the event in all ECG leads are the same for

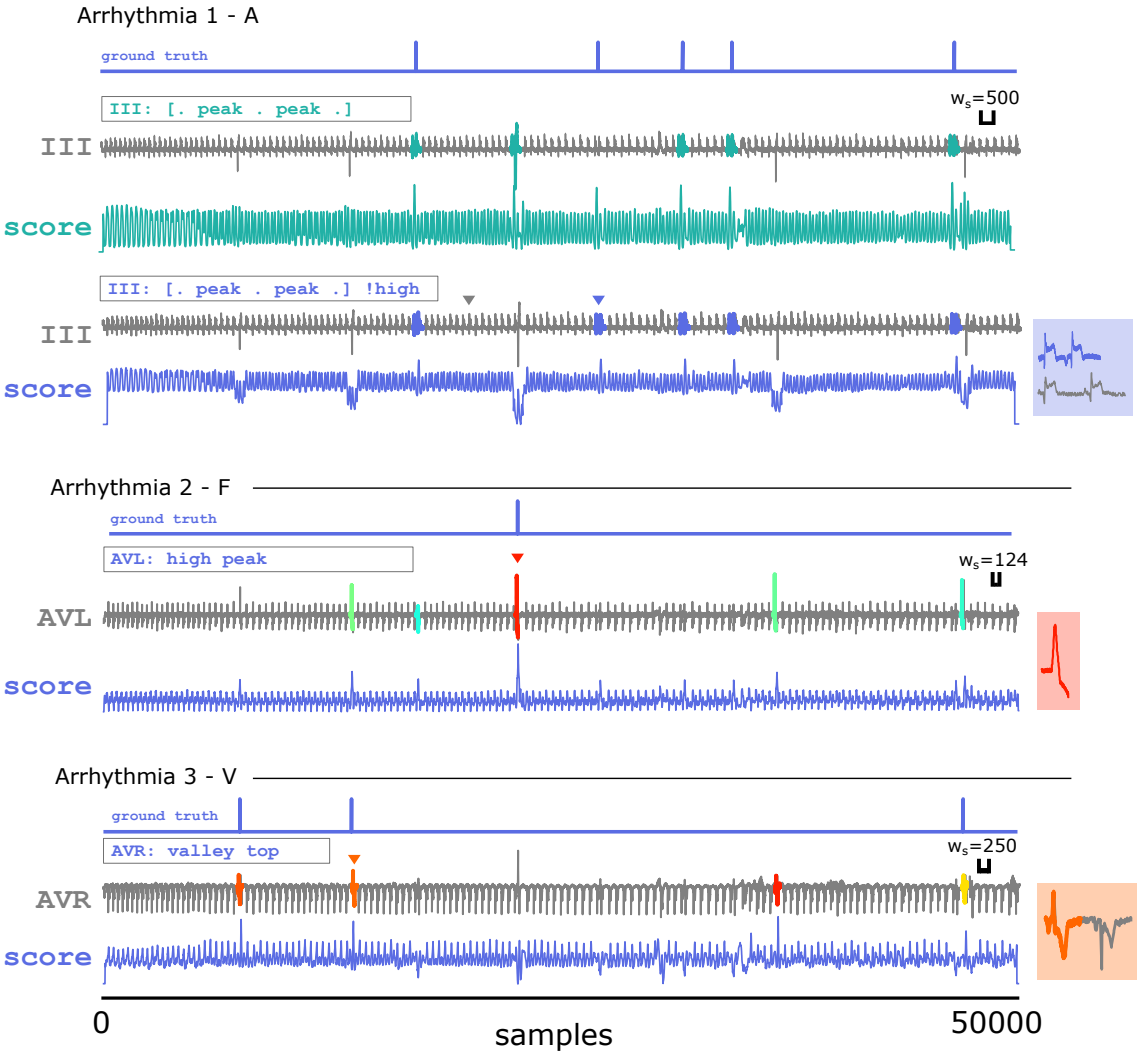


Figure 8.18: Examples of the detection of three specific cases of arrhythmia. The ground truth is selected from the annotations of specialists in the area. The queries are presented in text boxes and the found patterns are highlighted. On the side, an example of a match in a larger size is shown.

this type of arrhythmia, so we suspect that there was a missing label that the query was able to indicate.

In addition to searching for specific shapes on signals, QuoTS can be very useful for exploratory purposes in multivariate datasets. For instance, consider the following examples on a dataset acquired with the purpose to identify workload on drivers [134]. In this experiment, drivers physiological signals, such as ECG and SCR, were monitored. Exploring this dataset with QuoTS led to discovering several interesting occurrences, as shown in Figure 8.19. In this Figure are signaled specific events with road signs. We matched these signs' positions by finding their *latitude* and *longitude* on the map and matching them with the instant in time these coordinates occur on the dataset. In addition to this information, we recall the common terms used for auto telemetry, namely: Surge - breaking (peak) and accelerating (valley); and Sway - turning right (peak) or left (valley).

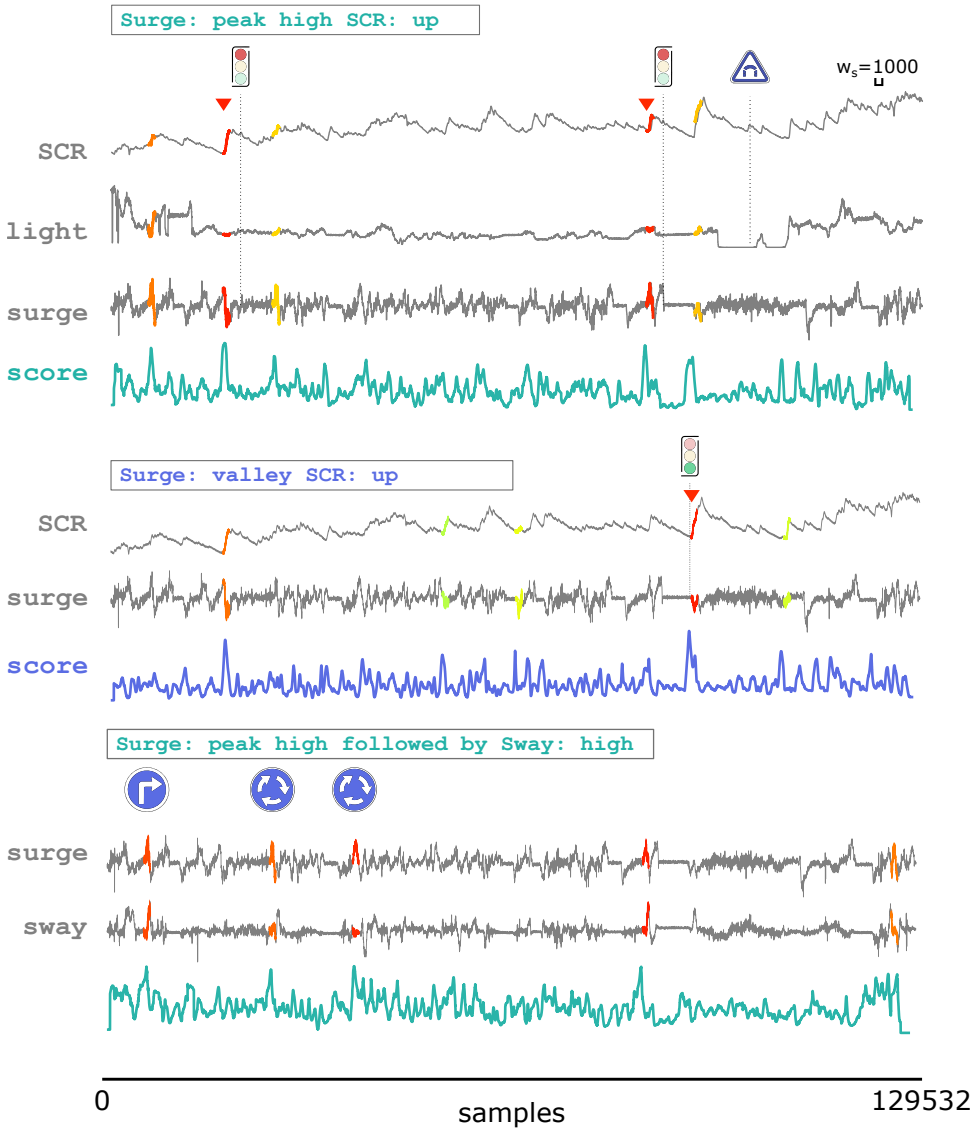


Figure 8.19: Example of multivariate patterns on a dataset from auto telemetry with physiological signals of the driver. The queries are written in text boxes. Several matches are highlighted based on the queries written. These events are associated with specific occurrences during the driving session, symbolized by traffic signs. These events were matched by searching the coordinates on the map. [135]

The first found events were identified by searching for moments where the driver was pulling the breaks and at a similar moment in time, the driver was having an increasing value of skin conductance (indicative of higher sweat and possibly momentary stress). The query can be written as `Surge: peak high SCR: up`. We found that the presence of such events is very high on two occasions, in both cases, right before a traffic light. The second traffic light was turned red because the car stopped. This led to searching for a moment where the driver started accelerating and the skin conductance was increasing (`Surge: valley SCR: up`), which points to two main locations on the signal. The second is exactly when the car started driving after stopping at the traffic light, suggesting

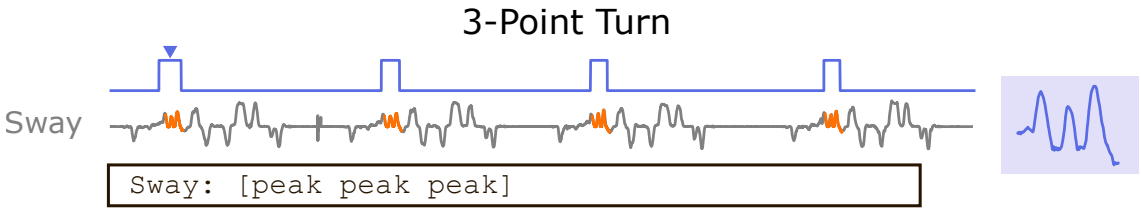


Figure 8.20: Example of searching for a 3 point turn event with QuoTS.

to be a relevant reaction from the driver. We did not have access to the videos, and can only guess what happened during the driving experience, but we can conclude that these identified moments are relevant to be observed afterward on the video, highlighting that QuoTS can be used to explore the signals very quickly, identify moments of interest (especially correlating multivariate variables) and search them on the video to validate what happened.

In addition to these events, we searched for moments where the driver had to break and then turned. There is a time dependency between these two occurrences and can use the `followed by` operator for this purpose. Therefore, we searched for `Surge: peak high followed by Sway: high`. The results pointed out several locations on the signal, three of them associated with turning right at a crossing and two roundabouts.

8.4.3 Matching *Known* Shapes with Words

We believe that QuoTS can be developed to be 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 it is possible to perform a *mimicry* of the process being searched and add the query designed with the mimicry process as a *word feature vector* inside our vocabulary. We show a few examples of possible applications, starting with "puppeteering" a model car in performing a 3-point-turn. First, we will show the signal where we searched this event and how we used QuoTS to find it with a text query.

Figure 8.20 is showed the search for a 3-point turn on a signal acquired with a smartphone inside a car while performing several driving exercises in a parking lot. Details of the experiment can be found at the Github repository [2]. The query used to match this event is very intuitive, being `[peak peak peak]`, which is exactly what this pattern is ().

Having found the desired pattern with text, we thought that it would also be possible to find it using a specific pattern and give it a name, to use on QuoTS. In this case, the pattern was acquired with a toy model car, on which a smartphone was attached (as indicated in Figure 8.21) 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

[2] https://github.com/Anonymous14151/QuoTS

on motion data or how the inertial data from the motion/manoeuver can be used as a template on QuoTS:

• **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.

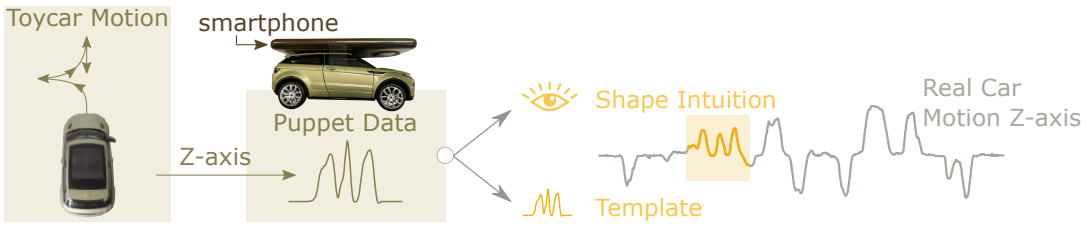


Figure 8.21: Creating query prompt data by puppeteering a car model. The puppet data for a 3-point turn (smoothed with 10 samples moving average) is presented on the left, while the real data from a real car is presented on the right.

We added the puppet data and the word `3pointturn` to our vocabulary. Searching for it in the signal matched all the desired shapes. Note that the template has a different amplitude and time scale considering that the motion was made with our hand, but 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.

9

Conclusion

In this work, fundamental topics in time series data mining were researched. These topics were motivation by considering the need for tools that help analysts to better understand what happened during the recording process and find relevant instances on signals that can be related with specific occurrences in the physical world. These motivations are intrinsically related with tools that are more visually interpretable and search mechanisms that are more expressive and intuitive. The work developed contributed to each of these domains with relevant standard mechanisms that can be further developed into practical tools for real scenarios. Not only these tool can be practical, but the methods presented bring novelty into the state of the art, either by "borrowing" more traditional methods from other domains (such as the SSM in audio information retrieval or Natural Language Processing (NLP) techniques for text mining) or by introducing novel concepts, specially in terms of text representation of biosignals and text-based query search mechanisms.

In this Chapter, we highlight the main contributions of this thesis in each of this domains. Comments are also given regarding the contributions and applicability to occupational health scenarios. Finally, overall scientific production and collaborations during the period of this thesis are also provided.

9.1 Main Contributions on General Topics

As we mentioned in Chapter 1, this thesis contributes to three main topics, namely *Sensing*, *Analysis* and *Decision Making*:

- *Sensing:* In this context, a deep understanding of the existing technology to record biosignals was made. At some point, the focus moved towards technology existing to monitor occupational variables. A written work is available regarding the usage of fiber-optics for the monitoring of motion and postural variables in automotive industries, which can be found in [120] and it is compared to other existing methods. In addition, a study of the state of the art sensors existing on the market was made to understand the fit in the occupational health problematic. A presentation is available at [121], showing the existing materials on the market and that cover human motion

and physiology, more focused for occupational health. The contributions in this area are more related with the knowledge gained and how it was used to more appropriately prepare an acquisition plan regarding the acquisition of occupational variables in several settings, namely automotive industry (Volkswagen Autoeuropa), clothing manufacturers and office/desk jobs.

- *Analysis:* This work has demonstrated a deeper exploration of the topics related with the analysis of biosignals and time series in general. The contributions include the usage of the SSM for the segmentation of time series (*novelty* and *periodic* functions), summarization and creation of similarity profiles for (semi-)automatic clustering. A novel symbolic representation of time series was introduced, and examples were provided in how to apply it for pattern search with regex (SSTS) and text-based classification (HeaRTS). A novel search mechanism was also developed, closer to natural language search, with keywords and operators (QuoTS). A more detailed explanation of these contributions are provided on a further section.

- *Decision Making:* The main purpose of the studied and developed methods was to help analysts when inspecting time series, but also move towards democratizing pattern search on time series. The proposed strategies provide several levels of understanding and help in several layers of decision making. Of course it will depend on the purpose, context, output delivered by the method and the category of expertise of the analyst.

  Starting with the analysis of structural information with the SSM, it provides a visual output that has characteristic structures that will appear independently of the type of data or context. The main information available on the SSM will always be *blocks*, *paths* and *similarity*. As we have seen, these can help the analyst identify specific occurrences on time series, with special interest in biosignals applications. Therefore, by learning how to retrieve information from the SSM, more awareness is given to the analyst and more conscious decisions can be made based on that information.

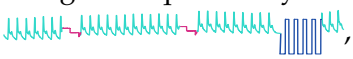
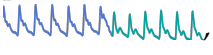
  From a standard perspective, the analyst can also perform automatic or semi-automatic segmentation and labeling/annotation of data to accelerate the preparation process to train supervised machine learning algorithms. This can either be done with the visual support of the SSM or the methods developed to extract the information automatically.

  Regarding the process of pattern search with more expressive queries, we believe the tools provide valuable resources to both the analyst that works in the time series domain and is experienced in the computer science field (e.g. a researcher that develops machine learning methods for biosignals processing), and the analyst that

is not an expert in computer science but has knowledge in a specific domain of time series (e.g. a physician that is experienced in ECG data). In this case, the process is more interactive and requires writing a regex/text query. This should be useful to accelerate the search process by an experience computer scientist, because it should be more quick to make the search with this system than developing an algorithm for that purpose. In addition, it contributes to democratize the search mechanism to more analysts than only computer scientists, because if the analyst is able to describe the shape being searched, it should be possible to find it (considering the right connotation methods/word feature vectors are used). In that aspect, the information retrieval would quickly help the analyst in taking more informed decisions.

9.2 Scientific Contributions

9.2.1 Unveiling the *Grammar* of Time Series

One of the major topics of this work regarded the segmentation of time series into smaller segments, based on *novelty* and *periodicity*. In addition, it was also discussed the benefit of relating the resulting subsequences by how similar these are. As an example, we showed the ABP signal  , which can be divided into 7 segments, having the structure A B A B A C A. We demonstrated with strong evidences that the usage of the SSM is reliable in performing this type of task. From the SSM, the novelty function can be extracted and the similarity profiles can be compared to perform a segmentation and association between subsequences. In addition, the segmentation might be periodic, meaning that a signal, such as  , can be separated into A and B, but also, AAAAAAABBBBBB. We also demonstrated that using the SSM, we can compute the similarity function, which highlights the cyclic nature of the subsequence.

The performance of the method was validated for the novelty segmentation process. It was compared to several SOA methods, showing to be competitive in tasks related with change point detection and segmentation. In addition, several use-cases from various fields were presented as examples, showing that the algorithm is agnostic to the type of signal, applicable to multidimensional and multimodal signals, and not requiring any previous knowledge on the data, such as the number of segmentation points. It also shows potential to be used for unsupervised annotation of data, being developed towards this purpose.

In this work, datasets were chosen to cover as much real scenarios as possible. Also, common benchmarks were used to validate the methods to guarantee independence from private datasets. Besides, the proposed method was also demonstrated to work well with multimodal and multidimensional occupational data.

There are still several improvements to be made, namely considering the excessive memory that is required to compute the SSM in cases where the signal is very large.

The fact that a matrix has to be computed limits the ability to analyze in one run the entire signal. This process is specially relevant for periodic segmentation and computing similarity profiles. As previously explained, for novelty segmentation, the process can be adapted to only compute the SSM along the diagonal with the size of the kernel width. An additional limitation is the fact that the method is not invariant to trend. If events occur in slow trend changes, that is, a continuous linear change, the method will have more difficulty in identifying the segmentation point. The usage of pre-processing, additional time series representations, or time series decomposition methods could help in counteracting this effect.

The ability of the method to be adapted in Online scenarios has not been discussed, but a solution should also be considered in the future.

9.2.2 Using Language for Time Series Data Mining

Representing data into different data types provides a new look on the original data. It may lead to find segments of interest that were not visible in the original data type and/or may benefit from the large experience in mining on this new data type. These are the first arguments for the ideas we presented in this work in transforming time series from the numerical domain to the text/symbolic domain. A new look on time series is possible, which enables to adapt some of the existing data mining techniques with a textual approach, and it can benefit from the large knowledge on text processing or NLP.

As a first concept, language and time series can apparently be a strange combination. However, we provided additional evidence that there is a bridge and a potential to perform several tasks with success, namely in text-based query pattern search with SSTS and QuoTS, as well as classification with HeaRTS.

The concept of symbolic representation has started with SAX [86], which was a great inspiration for the work developed further with SSTS. We developed this method with the purpose of making pattern search with more expressive queries, more closely related with the way we look and interpret visually the existing shapes on time series. Several examples were provided showing the possibility of using regex (text patterns) on time series symbolic representation. Additionally, having higher levels of representation could be useful to search for increasingly higher-leveled structures, such as `peaks`, `plateaus` or a combination of these. This idea was what led us perform the next method for time series classification, HeaRTS.

We imagined that if time series could be *translated* into text documents, these could be differentiated based on the words and sentences that would represent them. Inspired by NLP techniques used for this purpose, such as BoW and TF-idf, we performed classification of time series documents, by creating a high-leveled distance measure that relies in the presence of structures, such as `peak`, `plateau`, `up`, `down` and `flat`, and the order of these words in a sentence with *ngrams*. We then showed that it was possible to use traditional NLP methods to perform this task on the UCR classification benchmark. It

was able to have a better performance than the 1-NN ED and showed to be competitive in this field. Especially because there is the possibility of extracting valuable information from the textual translation, namely by using the TF-idf weights to highlight areas of relevance on the signal or keywords that mostly represent the topic of the signal (such as topic modeling on text domain). We believe we introduced a novel idea with these processes that can be helpful for search but also for explaining which are the differences between signals. There is still a lot to improve, since as we showed, it would only be interpretable for time series with simple characteristics explained by the *connotation* and *queries* developed and used to describe the signals.

Having queries closer to the way we express what we see was one of the main motivations of this work. This led to the development of SSTS in the symbolic domain, but we believe as well that features are a good match to specific words that we used to describe parts of signals. This led to the development of QuoTS, which uses word-feature vectors to search for specific subsequences on time series by how well these match the set of keywords used, similarly to how we type keywords on *Google* to search for web pages. We provided evidence of its usage in several types of signal and with several types of problems, from motion gestures, to ECG patterns or telemetry data, in multidimensional time series. We highlight the potential to use this method to search subsequences based on visual intuition but also for *words* that are *known*, namely by *puppeteering* or *mimicking* shapes in practice. This is specially relevant if keywords can be transformed for each domain, being domain specific. Considering that the vocabulary can change from domain to domain, for instance, `peak` in medicine can mean an ECG peak, while in automotive telemetry, it might mean *turn right*. This domain specific match can help other non-experienced analysts to use it to search for specific patterns.

9.3 Other Contributions

9.3.1 Managing Rotation Plans with Exposure, Diversity and Team Homogeneity

In close collaboration with Volkswagen Autoeuropa and the Faculty of Human Motricity of Lisbon (FMH), we developed a method to automatically suggest job rotation schedules based on ergonomic standards available at the factory. These standard factors are from the AutoErgo tool, based on EAWS measures. The motivation for the development of such a tool was to help team leaders to manage job rotation schedules more quickly and in a more informed way. Team leaders organize the working schedule for their team by assigning each worker to a sequence of workstations for the entire week, which is a time consuming tasks and not always informed in the risk level that each tasks represents for a worker. In this method, risk exposure, diversity in exposure, as well as team homogeneity, are taken into consideration when suggesting a daily rotation plan. The process was made by developing a genetic based optimization algorithm that followed an objective function

developed by our team. The motivation, algorithm and results can be seen at [12].

This work was developed at the initial stage of the PhD, being valuable to get a better intuition of biosignals related with human motion. Having gained this intuition helped in understanding relevant aspects of signals and transfer this gain knowledge into the algorithms developed. For instance, problems such as segmentation were common in the signals acquired, which promoted the development of methods that could solve them (SSM). In addition, the shape of signals, associated with specific motions, were interesting to study in order to get inspired in the possible symbolic translation (SSTS).

9.3.2 MicroErgo - Concept for Personal Assessment of Occupational Risk in Desk/Office Jobs

A lot of focus has been given to occupational health scenarios during this thesis. Especially for the main projects in which the group was involved. One of these projects is Prevention of Occupational Disorders in the Public Administration with AI (PrevOccupAI), which has the purpose of preventing occupational disorders in office jobs, namely from the public administration. One of the ideas conceptualized during the project was a self-assessment tool for office workers, based on the idea of *microCovid* [40]. The purpose was to help create more awareness about the biomechanical, environmental and mental occupational variables that affect our health. This would be a beneficial approach for any company to self-assess their occupations or even for remote workers who are not always aware if their desk setup is good or not for their biomechanical health, for example. The work can be found here [124].

9.3.3 In using Direct Measures for Occupational Health Assessment

The methods studied and developed in this thesis are general and applicable to any type of time series. This means that these are applicable to direct measures from the occupational domain for information retrieval, as showed on the last section of the previous chapter. We showed that this context was always considered and highly influenced by problems from industry and office/desk jobs.

During this period, a complete understanding of occupational variables was made. This was essential to understand the sensors that could be interesting to use to monitor these variables. Only inertial variables were used to perform a motion capture of upper body segments, which would give most of the angular information needed to study postural variables present on EAWS (this was how Dataset **??** was acquired and more details can be found there.). The usage of direct measures in this context helped in understanding the level of risk a specific workstation represents for a worker. For instance, it is possible to understand for a specific working cycle, which percentage of time it has a high, medium and low risk (using standard ergonomic measures from RULA.). It was also possible to conclude that the same workstation is performed differently by workers with different anthropometric features, which means that a specific workstation should

not be considered to have the same risk score for all workers. These conclusions can be found at [128]

The usage of direct measures in this context is therefore highly valuable, considering that standard methods can be used to (mostly) automatically calculate risk scores for each worker and each workstation. Using these measures, a specific workstation can be studied in detail, by means of understanding which processes contribute with the highest risk, for example. Another scenario involves studying how to adapt new workstations to improve productivity or reduce occupational risk. In either case, these direct measures can be used to measure the risk of the processes that were added/removed/modified to the workstation being proposed and understand if it truly is beneficial or not.

The value of direct measures is also related to the existing and continuously increasing knowledge in data mining. Methods, such as the ones developed in this work, can be used to extract relevant information from this data. As we showed, segmentation and pattern search are examples of possible mechanisms for information retrieval in these datasets. Specific *known* shapes can be searched with query-based mechanisms, either by text or *subsequences* used as examples. At some point, supervised learning methods can be made to create working profiles for workstations and workers that consider differences in anthropometric features, specific types of processes and the associations between these.

Finally, another possible usage of these direct measures, would be to design automatically job rotation schedules that use this personal and individual information. As we showed previously, an algorithm was developed with this purpose, but the measures considered were from EAWS standards, which, as reflected in [128], do not consider differences between workers. This provides an additional level of detail that could help better assign workers to workstations, based on their level of capacity.

9.3.4 Volatile Organic Compounds Classification

A Master Thesis in collaboration with the Biomolecular Engineering Group, from the Chemistry Department of the NOVA University of Lisbon. We developed the first version of HeaRTS for the classification of Volatile Organic Compounds (VOC)s. This resulted in a publication that can be found here [6].

9.4 Scientific Production

During the period of this thesis, the work developed has been disseminated *via* scientific publications. In addition, several research collaborations were made that also resulted in collaborative publications. The outcomes are hereby presented.

9.4.1 Journal Publications

- M. L. Nunes et al., "Posture Risk Assessment in an Automotive Assembly Line Using Inertial Sensors," in IEEE Access, vol. 10, pp. 83221-83235, 2022, doi: 10.1109/ACCESS.2022.3196473.

- Mollaei N, Fujao C, Silva L, Rodrigues J, Cepeda C, Gamboa H. Human-Centered Explainable Artificial Intelligence: Automotive Occupational Health Protection Profiles in Prevention Musculoskeletal Symptoms. Int J Environ Res Public Health. 2022 Aug 3;19(15):9552. doi: 10.3390/ijerph19159552. PMID: 35954919; PMCID: PMC9368597.

- Assunção, Ana; Mollaei, Nafiseh; Rodrigues, João; Osório, Daniel; Veloso, António; Cautela, Filomena; Gamboa, Hugo. A genetic algorithm approach to design job rotation schedules ensuring homogeneity and diversity of exposure in the automotive industry, Heliyon, Volume 8, Issue 5, e09396 (2022). https://doi.org/10.1016/j.heliyon.2022.e09396.

- Ramos, G.; Vaz, J. R.; Mendonça, G. V.; Pezarat-Correia, P.; Rodrigues, J.; Alfaras, M.; Gamboa, H.. "Fatigue Evaluation through Machine Learning and a Global Fatigue Descriptor". Journal of Healthcare Engineering 2020 (2020): 1-18. http://dx.doi.org/10.1155/2020/6484129.

- Rodrigues, João; Folgado, Duarte; Belo, David; Gamboa, Hugo. "SSTS: A syntactic tool for pattern search on time series". Information Processing and Management 56 1 (2019): 61-76. http://dx.doi.org/10.1016/j.ipm.2018.09.001.

9.4.2 Book Chapters

- Santos, Sara; Folgado, Duarte; Rodrigues, João; Mollaei, Nafiseh; Fujão, Carlos; Gamboa, Hugo. "Exploring Inertial Sensor Fusion Methods for Direct Ergonomic Assessments". In Communications in Computer and Information Science, 289-303. Springer International Publishing, 2021;

- Gamboa, Patricia; Quaresma, Cláudia; Varandas, Rui; Canhão, Helena; de Sousa, Rute Dinis; Rodrigues, Ana; Jacinto, Sofia; et al. "Design of an Attention Tool Using HCI and Work-Related Variables". In IFIP Advances in Information and Communication Technology, 262-269. Portugal: Springer International Publishing, 2021;

- Rodrigues, João; Gamboa, Hugo; Mollaei, Nafiseh; Osório, Daniel; Assunção, Ana; Fujão, Carlos; Carnide, Filomena. "A Genetic Algorithm to Design Job Rotation Schedules with Low Risk Exposure". In IFIP Advances in Information and Communication Technology, 395-402. Portugal: Springer International Publishing, 2020.

- Cepeda, Catia; Rodrigues, Joao; Dias, Maria Camila; Oliveira, Diogo; Rindlisbacher, Dina; Cheetham, Marcus; Gamboa, Hugo. "Mouse Tracking Measures and Movement Patterns with Application for Online Surveys". In Machine Learning and Knowledge Extraction, 28-42. Springer International Publishing, 2018.

9.4.3 Conference Proceedings

- Maryam Shahcheraghi, Ryan Mercer, João Manuel de Almeida Rodrigues, Audrey Der, Hugo Filipe Silveira Gamboa, Zachary Zimmerman, and Eamonn Keogh, "Scaling Time Series Similarity Matrices to Massive Data", IEEE International Conference on Data Mining (ICDM), Orlando, FL, USA, December 2022. *(Accepted)*

- Silva, Sara; Cepeda, Catia; Rodrigues, João; Probst, Phillip; Gamboa, Hugo. "Assessing Occupational Health with a Cross-platform Application based on Self-reports and Biosignals". Paper presented in BIOSTEC, Virtual, 2022.

- Alves, Rita; Rodrigues, João; Ramou, Efthymia; Palma, Susana; Roque, Ana; Gamboa, Hugo. "Classification of Volatile Compounds with Morphological Analysis of e-nose Response". Virtual, 2022.

- Mollaei, Nafiseh; Cepeda, Catia; Rodrigues, Joao; Gamboa, Hugo. "Biomedical Text Mining: Applicability of Machine Learning-based Natural Language Processing in Medical Database". Virtual, 2022.

- Rodrigues, Joao; Probst, Phillip; Gamboa, Hugo. "TSSummarize: A Visual Strategy to Summarize Biosignals". 2021.

- Santos, António; Rodrigues, João; Folgado, Duarte; Santos, Sara; Fujão, Carlos; Gamboa, Hugo. "Self-Similarity Matrix of Morphological Features for Motion Data Analysis in Manufacturing Scenarios". 2021.

- Rodrigues, Joao; Gamboa, Hugo; Kublanov, Vladimir; Dolganov, Anton. "Storage of Biomedical Signals: Comparative Review of Formats and Databases". Paper presented in Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Yekaterinburg, 2019.

9.4.4 Methods

In this thesis, we contributed to the state of the art with several methods, hereby listed.

- TSSummarize: Summarization of Time Series - Using the SSM to provide relevant feedback on how the time series are structured and segments are related, towards automatic and unsupervised annotation of time series.

- SSTS: Synthatic Search on Time Series - performing search on a symbolic representation of times series with regular expressions.

- HeaRTS: Human Readable Time Series - Higher level classification process of time series with visual and possible keyword feedback on data differences.
- QuoTS: *Where* on Time Series? - Text-based query search on word-feature vectors.
- Unsupervised Automatic Annotation: Using the SSM to search for segmentation points on any time series, including novelty segmentation and periodic segmentation, and automatically cluster the segments into similarity groups.

9.4.5 Projects

- Project Operator
- Project PrevoccupAI

9.4.6 Revised Works

The content of this thesis used peer-reviewed sections of the listed publications above. The sections that used part of this text are listed as follows:

Section 2 - Essential definitions use the text from paper X

9.4.7 Awards

9.4.7.1 Fullbright

I was awarded in 2021 a Fullbright scholarship to pursue a research project at the Computer Science Department of the University of California, Riverside (UCR). The exchange program was made under the supervision of prof. Eamonn Keogh. It made possible a close collaboration in the development of QuoTS. We are now working in the submission of a conference paper and a journal paper on this topic.

9.4.7.2 Best Paper Award

The best paper award for the category was awarded to the publication "MicroErgo: A Concept for Self-Assessment of Occupational Risk" at the Seventh International Conference on Biosignals, Images and Instrumentation conference, 2021.

10

Future Work

The work developed in this thesis shows promising results that present improvement opportunities, and it also sets a fertile ground for novel promising ideas. In this section, we will discuss the possibility for improvement in each of the methods presented, as well as which novel ideas follow the currently developed strategies.

10.1 Overall Improvements to compute the SSM and Segmentation Process

The SSM was computed by using the standard approach found at [110]. The results show promise in using this strategy for several tasks. We believe there are improvements that can be made in the feature extraction process, and contribute to a better SSM. Currently, neither a dimensionality reduction method, such as Principal Component Analysis (PCA), nor a feature selection process, are performed prior to computing the SSM. Performing these might be a way of improving the general matrix representation and remove features that do not contribute to explain the similarity between subsequences. Additionally, feature stacking has been proved to help in increasing the accuracy of feature-based classification of time series [90]. This means that it should better represent differences between subsequences and should be tested on the current approach.

Another relevant improvement would be to find a way to minimize the number of parameters used for the novelty segmentation task. Currently, the size of the sliding window that extracts features and the size of the sliding kernel that computes the novelty function are independent. It would be interesting to reduce the number of variables by finding a relationship between the kernel and moving window, make the sizes adaptive based on a specific metric or perform a training process where these parameters are computed for a specific domain and kept the same for future iterations (as we showed the values of these parameters are close together for the same dataset and type of task).

The general idea when this method was used, was that changes on the signal should be represented by a change in the overall set of features. This leads to the idea that different changes can be associated with a specific group of features. In the future, we should study

which are the features that better describe a specific type of feature.

10.2 Unsupervised Automatic Segmentation and Labeling of Time Series

We have already mentioned in a previous chapter that the proposed methods help moving towards unsupervised and automatic segmentation and labeling of time series. We have showed that the segmentation process is well performed, but lacked the validation for automatic labeling. We believe that using the similarity profiles after a first segmentation process would provide this ability of automatically returning a completely segmented and annotated signal. In the future, we will test this approach on public datasets.

10.3 Hierarchical Segmentation of Time Series

When analyzing a long time series visually, the user has to zoom-in and zoom-out to search for areas of interest. We believe that using the proposed segmentation method we could perform a hierarchical segmentation, that is, apply multiple segmentation stages, with different window lengths. This would provide a multi-layered set of information that can highlight the areas of interest in different *zoom* levels.

As a pilot for this method, the user could define the number of hierarchies and corresponding window sizes to perform this process, but ideally, the process would be performed with an adaptive sliding window, that alters the dimensions based on a specific metric. This process would be helpful for long time series, with highly variable information along time.

10.4 Periodic Segmentation

The current approach for periodic segmentation is not able to search for the off-diagonals (paths) that are used for this purpose. We believe a better approach could be performed if the paths were highlighted. For this purpose we can perform image processing filters and then extract the resulting paths. Identifying their beginning would be the best way to find the initial sample of a period.

10.5 Online Unsupervised Segmentation

This work has not discussed the application of the proposed method for online purposes. We believe the method can be adapted for both novelty segmentation and periodic segmentation. Of course that if the entire SSM is computed over time with continuous incoming data, the memory required for this process would not be enough and the algorithm would fail very quickly. For this purpose, the method should be adapted by only keeping samples

from the segmentation point forward (with a fixed buffer size), and compare the next incoming samples with the kept ones until a relevant change is identified or a new off-diagonal starts. After this, the previous samples kept on a buffer up to the segmentation point can be erased and the method can search for the next relevant change point. In the future, an online version should be developed.

10.6 Tool for Time Series Profiling

Currently, we introduced *TSSummarize*, which provides a summarization of the time series based on segmentation points and similarity profiles. We believe the development of an interactive tool that has an internal report on the time series, such as, statistical patterns on the segments, number of segments, how similar they are, percentage of time each segment is represented on the signal, level of periodicity, how many periods are there in each segment, presence of anomalies/discords or motifs, among other measures. In the future, this should be considered.

10.7 SSTS Improvements and Further Applications

The current SSTS method has several improvements to be made. Some of them have been introduced when developing HeaRTS, but others still require additional research. The fact that we are performing a symbolic representation gives the opportunity to use compression techniques typically used for text, such as the run length encoding (RLE). Having a compressed representation can make the search process faster. In combination with this, it would be interesting to include the higher level translation performed in HeaRTS, which has standard queries for standard structures (peaks, up, plateau, etc...). Using a compression of the time series can be beneficial for HeaRTS to speed up the text representation process.

A regex query is a text pattern, which is convenient to express a general pattern. However, it is sometimes difficult to generalize. The fact that the regex is not flexible makes this very brittle to patterns that we are looking for but have a slight difference with our text pattern. For now, this flexibility can be introduced with pre-processing/simplification of the data. In the future, a flexible search process, based on a meta-regex mechanism should be developed to perform a less brittle search.

The fact that we are searching for patterns makes it convenient to perform interactive adaptations to the data. Several methods have been thought for the *edition* of time series subsequences found with the text pattern, for instance *ssts.annotate*, *ssts.split*, *ssts.modify*, *ssts.replace*, *ssts.reverse*, *ssts.repeat*, *ssts.recursive*. Some of these functions are inspired in text edition mechanisms, others are used for general edition processes. We find that having a tool that could be used to edit, search or adapt a signal would be interesting.

We have showed that SSTS can be used in combination with existing NLP methods for classification processes and pattern search. We believe that with the current rise in NLP

knowledge, a symbolic representation of time series can be useful to design novel ways of extracting information from time series. For instance, several methods are available for text topic modeling, such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) or Non-Negative Matrix Factorization (NMF). These methods could be directly tested with the symbolic/textual representation of the time series. Other strategies, that rely in neural networks, such as *bert* and *transformers* could be explored for time series problems, namely for time series classification and generation. We believe these approaches can even be more relevant regarding interpretability and explainability. These are complex problems with time series, and having text as a medium of communication between analysts and the time series could improve current approaches on this domain. This was explored with HeaRTS, but the process was only evaluating the differences between the data and not explaining why the classifier selected a specific class.

Regarding the topic of classification with HeaRTS, we believe it should be adapted to work with multidimensional data. This could be made by combining several classifiers for each dimension and then combine the classification results by a standard voting method.

SHOW THE EXAMPLES I HAVE WORKED ON WITH SSTS: EDITION TOOL GENERATION PROBLEM FLEXIBILIZE THE SEARCH PROCESS FEATURE EXTRACTION TOOL LABELLING TOOL

10.8 Further Developments for QuoTS

We have introduced a novel method for pattern search on time series with word-feature vectors. This approach is more expressive and provides an easy way to search for patterns. In the future, this expressiveness should be measured with a study group. A group should develop a solution for a problem without using *quots*, and then use QuoTS to solve the same problem. The average time in solving the problem would be measured to associate with the expressiveness. In the long run, this method should be considered for non-experts in time series as well, to understand what could be improved to make the process more expressive for non-experienced users.

In terms of the method itself, more keywords and operators could be introduced. Some of the keywords could even be represented by several features that contribute for the used keyword. The fact that a static window is used, should be improved. For example, SSTS can find patterns with any size, while QuoTS searches for patterns on a fixed window size. Another improvement is related with the feedback given. QuoTS could have a set of methods that help get intuition over what the keywords mean in the time series. For instance, the user could highlight a segment of the time series and the keywords with higher value would be presented.

Bibliography

[1] R. P. Adams and D. J. C. Mackay. "Bayesian Online Changepoint Detection". In: *arXiv: Machine Learning* (2007) (cit. on pp. 33, 35).

[2] R. Agrawal, C. Faloutsos, and A. Swami. "Efficient similarity search in sequence databases". In: *Foundations of Data Organization and Algorithms*. Ed. by D. B. Lomet. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 69–84. ISBN: 978-3-540-48047-1 (cit. on p. 13).

[3] R. Agrawal et al. "Querying Shapes of Histories". In: *Proceedings of the 21th International Conference on Very Large Data Bases*. VLDB '95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 502–514. ISBN: 1558603794 (cit. on p. 22).

[4] K. R. Agres et al. *Music, Computing, and Health: A roadmap for the current and future roles of music technology for health care and well-being*. 2021-02. DOI: 10.1177/2059204321997709. URL: osf.io/mgjwv (cit. on p. 54).

[5] M. Aickin and H. Gensler. "Adjusting for multiple testing when reporting research results: the Bonferroni vs Holm methods." In: *American Journal of Public Health* 86.5 (1996). PMID: 8629727, pp. 726–728. DOI: 10.2105/AJPH.86.5.726. eprint: https://doi.org/10.2105/AJPH.86.5.726. URL: https://doi.org/10.2105/AJPH.86.5.726 (cit. on p. 31).

[6] R. Alves. et al. "Classification of Volatile Compounds with Morphological Analysis of e-nose Response". In: *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies - BIOSIGNALS*, INSTICC. SciTePress, 2022, pp. 31–39. ISBN: 978-989-758-552-4. DOI: 10.5220/0010827200003123 (cit. on p. 140).

[7] S. Aminikhanghahi and D. J. Cook. "A Survey of Methods for Time Series Change Point Detection". In: *Knowledge and Information Systems* 51 (2017-05). ISSN: 0219-3116. DOI: 10.1007/s10115-016-0987-z. URL: https://doi.org/10.1007/s10115-016-0987-z (cit. on pp. 1, 32, 34).

[8] D. Anguita et al. "A Public Domain Dataset for Human Activity Recognition using Smartphones". In: 2013-01 (cit. on p. 44).

[9] D. Anguita et al. "Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine". In: *Ambient Assisted Living and Home Care*. Ed. by J. Bravo, R. Hervás, and M. Rodríguez. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 216–223. ISBN: 978-3-642-35395-6 (cit. on p. 44).

[10] A. Antoniou and C. Vorlow. "Recurrence plots and financial time series analysis". In: 10 (2000-01), pp. 131–145 (cit. on p. 35).

[11] *ARTHROKINEMAT | Hui Liu | Research Project*. URL: https://www.researchgate.net/project/Arthrokinemat (visited on 2021-04-25) (cit. on p. 49).

[12] A. Assunção et al. "A genetic algorithm approach to design job rotation schedules ensuring homogeneity and diversity of exposure in the automotive industry". In: *Heliyon* 8 (5 2022-05). DOI: 10.1016/j.heliyon.2022.e09396. URL: https://doi.org/10.1016/j.heliyon.2022.e09396 (cit. on pp. 4, 139).

[13] I. Auger. "Algorithms for the optimal identification of segment neighborhoods". In: *Bltn Mathcal Biology* 51 (1989), pp. 39–54. DOI: https://doi.org/10.1007/BF02458835 (cit. on pp. 33, 35).

[14] A. Bagnall and E. Keog. *Time Series Classification Website*. 2022. URL: http://www.timeseriesclassification.com/urldate%20=%20%7B4-06-2022%7D (cit. on p. 43).

[15] J. Bai. "Estimating Multiple Breaks One at a Time". In: *Econometric Theory* 13.3 (1997), pp. 315–352. DOI: 10.1017/S0266466600005831 (cit. on pp. 33, 35).

[16] G. Batista et al. "CID: An efficient complexity-invariant distance for time series". In: *Data Mining and Knowledge Discovery* 28 (2013-04). DOI: 10.1007/s10618-013-0312-3 (cit. on pp. 17, 18).

[17] K. Bauters et al. "An automated work cycle classification and disturbance detection tool for assembly line work stations". In: *ICINCO 2014 - Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics*. Vol. 2. SciTePress, 2014, pp. 685–691. ISBN: 9789897580406. DOI: 10.5220/0005024406850691 (cit. on pp. 36, 37).

[18] K. Bauters et al. "Automated work cycle classification and performance measurement for manual work stations". In: *Robotics and Computer-Integrated Manufacturing* 51 (2018-06), pp. 139–157. ISSN: 07365845. DOI: 10.1016/j.rcim.2017.12.001 (cit. on p. 36).

[19] V. Behravan et al. "Rate-adaptive compressed-sensing and sparsity variance of biomedical signals". In: *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 2015, pp. 1–6. DOI: 10.1109/BSN.2015.7299419 (cit. on pp. 1, 47, 48).

[20] J. P. Bello et al. "Content-based Methods for Knowledge Discovery in Music". In: *Springer Handbook on Systematic Musicology*. Ed. by R. Bader. Springer, Berlin, Heidelberg, 2018, pp. 823–840. ISBN: 978-3-662-55002-1. DOI: https://doi.org/10.1007/978-3-662-55004-5_39 (cit. on pp. 37, 39, 54, 58).

[21] D. Belo. "Learning Biosignals using Deep Learning". PhD dissertation. Nova University of Lisbon, 2020 (cit. on p. 34).

[22] G. M. Bhandari, R. S. Kawitkar, and M. P. Borawake. "Audio Segmentation for Speech Recognition Using Segment Features". In: *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol II*. Ed. by S. C. Satapathy et al. Cham: Springer International Publishing, 2014, pp. 209–217. ISBN: 978-3-319-03095-1 (cit. on p. 54).

[0] P. Bonato. *Advances in wearable technology and applications in physical medicine and rehabilitation*. 2005 (cit. on p. 33).

[23] M. Bosc et al. "Automatic change detection in multimodal serial MRI: application to multiple sclerosis lesion evolution". In: *NeuroImage* 20.2 (2003), pp. 643–656. ISSN: 1053-8119. DOI: https://doi.org/10.1016/S1053-8119(03)00406-3. URL: http://www.sciencedirect.com/science/article/pii/S1053811903004063 (cit. on p. 1).

[24] P. Bota et al. "A Semi-Automatic Annotation Approach for Human Activity Recognition". In: *Sensors* 19 (2019-01), p. 501. DOI: 10.3390/s19030501 (cit. on p. 36).

[25] G. J. J. van den Burg and C. K. I. Williams. "An Evaluation of Change Point Detection Algorithms". In: (2020-05). DOI: arXiv:2003.06222 (cit. on pp. 11, 32–35, 49, 97–99, 185).

[26] F. Caputo et al. "IMU-Based Motion Capture Wearable System for Ergonomic Assessment in Industrial Environment". In: *Springer Nature* (2019), pp. 215–225. DOI: 10.1007/978-3-319-94619-1_21 (cit. on p. 4).

[27] P. Casale, O. Pujol, and P. Radeva. "Personalization and user verification in wearable systems using biometric walking patterns". In: *Personal and Ubiquitous Computing - PUC* 16 (2012-06), pp. 1–18. DOI: 10.1007/s00779-011-0415-z (cit. on p. 44).

[28] J. Chen et al. "Wearable sensors for reliable fall detection". In: *2005 IEEE engineering in medicine and biology 27th annual conference*. IEEE. 2006, pp. 3551–3554 (cit. on p. 33).

[0] K.-H. Chen et al. "Wearable sensor-based rehabilitation exercise assessment for knee osteoarthritis". In: *Sensors* 15.2 (2015), pp. 4193–4211 (cit. on p. 33).

[29] L. Chen, C. Nugent, and G. Okeyo. "An ontology-based hybrid approach to activity modeling for smart homes". In: *IEEE Transactions on human-machine systems* 44.1 (2013), pp. 92–105 (cit. on p. 33).

[30] L. Chen et al. "Sensor-based activity recognition". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 790–808 (cit. on p. 33).

[31] M. Chen, S. Mao, and Y. Liu. "Big data: A survey". In: *Mobile networks and applications* 19.2 (2014), pp. 171–209 (cit. on p. 33).

[32] H. Cho and P. Fryzlewicz. "Multiple-change-point detection for high dimensional time series via sparsified binary segmentation". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 77.2 (2014-07), pp. 475–507. ISSN: 1369-7412. DOI: 10.1111/rssb.12079. URL: http://dx.doi.org/10.1111/rssb.12079 (cit. on p. 1).

[33] D. Cook and N. Krishnan. *Activity learning: Discovering, recognizing, and predicting human behavior from sensor data*. 2015-02, pp. 1–257. ISBN: 9781118893760. DOI: 10.1002/9781119010258 (cit. on p. 1).

[34] D. Crystal. *Making Sense of Grammar*. Pearson Education, 2004-01. ISBN: 9780582848634 (cit. on p. 3).

[35] Z. Cui et al. "Enhancing Interactions for In-Car Voice User Interface with Gestural Input on the Steering Wheel". In: *13th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI '21. Leeds, United Kingdom: Association for Computing Machinery, 2021, pp. 59–68. ISBN: 9781450380638. DOI: 10.1145/3409118.3475126. URL: https://doi.org/10.1145/3409118.3475126 (cit. on p. 126).

[36] P. Cunningham and S. J. Delany. "K-Nearest Neighbour Classifiers - A Tutorial". In: *ACM Comput. Surv.* 54.6 (2021-07). ISSN: 0360-0300. DOI: 10.1145/3459665. URL: https://doi.org/10.1145/3459665 (cit. on p. 23).

[37] R. Cutler and L. Davis. "View-based detection and analysis of periodic motion". In: Institute of Electrical and Electronics Engineers (IEEE), 2002-11, pp. 495–500. DOI: 10.1109/icpr.1998.711189 (cit. on p. 37).

[38] R. Cutler and L. Davis. "Real-time periodic motion detection, analysis, and applications". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2 (1999), pp. 326–332. ISSN: 10636919. DOI: 10.1109/cvpr.1999.784652 (cit. on p. 37).

[39] R. Cutler and L. S. Davis. "Robust real-time periodic motion detection, analysis, and applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000-08), pp. 781–796. ISSN: 01628828. DOI: 10.1109/34.868681 (cit. on p. 37).

[40] D. Dagradi et al. *microCovid Project*. 2020. URL: https://www.microcovid.org/ (visited on 2022-07-07) (cit. on p. 139).

[41] R. B. Dannenberg and M. Goto. "Music Structure Analysis from Acoustic Signals". In: *Handbook of Signal Processing in Acoustics*. Ed. by D. Havelock, S. Kuwano, and M. Vorländer. New York, NY: Springer New York, 2008, pp. 305–331. ISBN: 978-0-387-30441-0. DOI: 10.1007/978-0-387-30441-0_21. URL: https://doi.org/10.1007/978-0-387-30441-0_21 (cit. on pp. 59, 60).

[42] H. A. Dau et al. *The UCR Time Series Archive*. 2018. DOI: 10.48550/ARXIV.1810.07758. URL: https://arxiv.org/abs/1810.07758 (cit. on p. 43).

[43] H. G. Espinosa, J. Lee, and D. A. James. "THE INERTIAL SENSOR: A BASE PLATFORM FOR WIDER ADOPTION IN SPORTS SCIENCE APPLICATIONS." In: *Journal of Fitness Research* 4.1 (2015) (cit. on pp. 1, 33).

[44] J. Faouzi and H. Janati. "pyts: A Python Package for Time Series Classification". In: *Journal of Machine Learning Research* 21.46 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/19-763.html (cit. on p. 15).

[45] D. Folgado et al. "TSSEARCH: Time Series Subsequence Search Library". In: *SoftwareX* 18 (2022), p. 101049. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2022.101049 (cit. on p. 32).

[46] J. Foote. "Automatic audio segmentation using a measure of audio novelty". In: *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*. Vol. 1. 2000, 452–455 vol.1. DOI: 10.1109/ICME.2000.869637 (cit. on pp. 37, 59).

[47] P. S. Foundation. *10.3. operator - Standard Operators as functions*. 2017. URL: https://docs.python.org/3/library/operator.html (cit. on p. 116).

[48] P. S. Foundation. *regex 2017.09.23*. 2017. URL: https://pypi.python.org/pypi/regex/ (cit. on p. 78).

[49] Fraunhofer AICOS. *IOTIP | INTERNET OF THINGS IN A PACKAGE: WAFER LEVEL MODULAR ARCHITECTURE FOR INTERNET OF THINGS*. Tech. rep. URL: https://www.fraunhofer.pt/en/fraunhofer%7B%5C_%7Dportugal/news/news%7B%5C_%7Darchive/a-%7B%5C_%7Dmade-in-portugal-technology-for-iot-growth.html (cit. on p. 51).

[50] J. Friedl. *Mastering Regular Expressions*. 3^{rd}. O'Rilley Media, Inc, 2006 (cit. on p. 25).

[51] P. Gamboa et al. "Attention Classification Based on Biosignals during Standard Cognitive Tasks for Occupational Domains". In: *Computers* 11.4 (2022). ISSN: 2073-431X. DOI: 10.3390/computers11040049. URL: https://www.mdpi.com/2073-431X/11/4/49 (cit. on p. 4).

[52] Y. Ganesan, S. Gobee, and V. Durairajah. "Development of an upper limb exoskeleton for rehabilitation with feedback from EMG and IMU sensor". In: *Procedia Computer Science* 76 (2015), pp. 53–59 (cit. on p. 33).

[53] S. Gharghabi et al. "Domain agnostic online semantic segmentation for multi-dimensional time series". In: *Data Mining and Knowledge Discovery* 33 (2019), pp. 96–130. ISSN: 1573-756X. URL: https://doi.org/10.1007/s10618-018-0589-3 (cit. on pp. 33, 67).

[54] S. Gharghabi et al. "Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels". In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, pp. 117–126. DOI: 10.1109/ICDM.2017.21 (cit. on pp. 20, 35).

[55] A. L. Goldberger et al. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals". In: *Circulation* 101.23 (2000-06). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215, e215–e220 (cit. on pp. 47, 48, 129).

[56] A. L. Goldberger et al. "PhysioBank, PhysioToolkit, and PhysioNet". In: *Circulation* 101.23 (2000), e215–e220. ISSN: 0009-7322. DOI: 10.1161/01.CIR.101.23.e215. eprint: http://circ.ahajournals.org/content/101/23/e215.full.pdf (cit. on p. 65).

[57] D. H. Seidel et al. "Quantitative Measures of Physical Risk Factors Associated with Work-Related Musculoskeletal Disorders of the Elbow: A Systematic Review". In: *International Journal of Environmental Research and Public Health* 16 (2019-01), p. 130. DOI: 10.3390/ijerph16010130 (cit. on p. 3).

[58] C. L. Hamblin. "Translation to and from Polish Notation". In: *The Computer Journal* 5.3 (1962), pp. 210–213. DOI: 10.1093/comjnl/5.3.210. eprint: /oup/backfile/content_public/journal/comjnl/5/3/10.1093/comjnl/5.3.210/2/5-3-210.pdf. URL: +%20http://dx.doi.org/10.1093/comjnl/5.3.210 (cit. on p. 75).

[59] J. Han, M. Kamber, and J. Pei. "2 - Getting to Know Your Data". In: *Data Mining (Third Edition)*. Ed. by J. Han, M. Kamber, and J. Pei. Third Edition. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 39–82. ISBN: 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00002-2. URL: https://www.sciencedirect.com/science/article/pii/B9780123814791000022 (cit. on p. 19).

[60] Y. Hartmann, H. Liu, and T. Schultz. "Interactive and Interpretable Online Human Activity Recognition". In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE. 2022, pp. 109–111 (cit. on p. 33).

[61] T. Heldt et al. "Circulatory response to passive and active changes in posture". In: *Computers in Cardiology, 2003*. 2003, pp. 263–266. DOI: 10.1109/CIC.2003.1291141 (cit. on p. 65).

[62] R. A. Henning, S. L. Sauter, and E. F. Krieg. "Work rhythm and physiological rhythms in repetitive computer work: Effects of synchronization on well-being". In: *International Journal of Human–Computer Interaction* 4.3 (1992), pp. 233–243. DOI: 10.1080/10447319209526040. eprint: https://doi.org/10.1080/10447319209526040. URL: https://doi.org/10.1080/10447319209526040 (cit. on p. 4).

[63] R. Howard. "Wireless sensor devices in sports performance". In: *IEEE Potentials* 35.4 (2016), pp. 40–42 (cit. on pp. 1, 33).

[64] R. M. Howard, R. Conway, and A. J. Harrison. "A survey of sensor devices: use in sports biomechanics". In: *Sports biomechanics* 15.4 (2016), pp. 450–461 (cit. on pp. 1, 33).

[65] Y. Hu et al. "OmicCircos: A Simple-to-Use R Package for the Circular Visualization of Multidimensional Omics Data". In: *Cancer Informatics* 13 (2014). PMID: 24526832, CIN.S13495. DOI: 10.4137/CIN.S13495. eprint: https://doi.org/10.4137/CIN.S13495. URL: https://doi.org/10.4137/CIN.S13495 (cit. on p. 39).

[66] S. Imani, S. Alaee, and E. Keogh. "Putting the Human in the Time Series Analytics Loop". In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. San Francisco, USA: Association for Computing Machinery, 2019, pp. 635–644. ISBN: 9781450366755. DOI: 10.1145/3308560.3317308. URL: https://doi.org/10.1145/3308560.3317308 (cit. on p. 1).

[67] S. Imani, S. Alaee, and E. Keogh. "Putting the Human in the Time Series Analytics Loop". In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. San Francisco, USA: Association for Computing Machinery, 2019, pp. 635–644. ISBN: 9781450366755. DOI: 10.1145/3308560.3317308. URL: https://doi.org/10.1145/3308560.3317308 (cit. on p. 40).

[68] S. Imani et al. "Introducing time series snippets: a new primitive for summarizing long time series". In: *Data Min. Knowl. Discov.* (2020), pp. 1713–1743 (cit. on p. 38).

[69] E. S. Irastorza, Xabier, and S. Copsey. *OSH in figures: Work-related musculoskeletal disorders in the EU — Facts and figures*. European Agency for Safety and Health at Work, 2010 (cit. on p. 3).

[70] H. Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963 (cit. on pp. 31, 99).

[0] I. Jakob et al. "Robotic and sensor technology for upper limb rehabilitation". In: *PM&R* 10 (2018), S189–S197 (cit. on p. 33).

[71] S. L. Jan Goyvaerts. *Regular Expressions Cookbook*. 2nd ed. O'Reilly Media, Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2012. ISBN: 978-1-449-31943-4 (cit. on p. 117).

[72] L. C. Jatobá et al. "Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity". In: *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS'08 - "Personalized Healthcare through Technology"*. Vol. 2008. IEEE Computer Society, 2008, pp. 5250–5253. ISBN: 9781424418152. DOI: 10.1109/iembs.2008.4650398. URL: https://pubmed.ncbi.nlm.nih.gov/19163901/ (cit. on p. 36).

[73] Q. Ji et al. "Real-time gait event detection in a real-world environment using a laser-ranging sensor and gyroscope fusion method". In: *Physiological Measurement* 39.12 (2018), p. 125003 (cit. on pp. 1, 33).

[74] M. Kayam et al. "Assessing your algorithm: A program complexity metrics for basic algorithmic thinking education". In: *2016 11th International Conference on Computer Science Education (ICCSE)*. 2016-08, pp. 309–313. DOI: 10.1109/ICCSE.2016.7581599 (cit. on p. 29).

[75] E. Keogh. *How to do good research, get it published in SIGKDD and get it cited*. 2009-07. URL: https://www.cs.ucr.edu/~eamonn/Keogh_SIGKDD09_tutorial.pdf (cit. on p. 30).

[76] E. Keogh and R. Chotirat Ann. "Exact Indexing of Dynamic Time Warping". In: *knowledge and Information Systems* (2005-03). DOI: 10.1007/s10115-004-0154-9. URL: https://doi.org/10.1007/s10115-004-0154-9 (cit. on p. 17).

[77] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. "Towards Parameter-Free Data Mining". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '04. Seattle, WA, USA: Association for Computing Machinery, 2004, pp. 206–215. ISBN: 1581138881. DOI: 10.1145/1014052.1014077. URL: https://doi.org/10.1145/1014052.1014077 (cit. on pp. 39, 40).

[78] E. Keogh et al. "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases". In: 2001-01. URL: https://www.microsoft.com/en-us/research/publication/dimensionality-reduction-for-fast-similarity-search-in-large-time-series-databases/ (cit. on p. 15).

[79] L. Kirichenko et al. "Two Approaches to Machine Learning Classification of Time Series Based on Recurrence Plots". In: *2020 IEEE Third International Conference on Data Stream Mining and Processing (DSMP)*. 2020, pp. 84–89. DOI: 10.1109/DSMP47368.2020.9204021 (cit. on p. 35).

[80] N. Kumar et al. "Time-series Bitmaps: a Practical Visualization Tool for Working with Large Time Series Databases". In: 2005-04. DOI: 10.1137/1.9781611972757.55 (cit. on pp. 38, 39, 69).

[81] Ó. D. Lara and M. A. Labrador. "A survey on human activity recognition using wearable sensors". In: *IEEE Communications Surveys and Tutorials* 15.3 (2013), pp. 1192–1209. ISSN: 1553877X. DOI: 10.1109/SURV.2012.110112.00192 (cit. on p. 36).

[82] S.-J. Lee et al. "Exercise and cardiovascular load in workers with high occupational physical activity". In: *Archives of Environmental and Occupational Health* 75.6 (2020). PMID: 31456490, pp. 339–345. DOI: 10.1080/19338244.2019.1657059. eprint: https://doi.org/10.1080/19338244.2019.1657059. URL: https://doi.org/10.1080/19338244.2019.1657059 (cit. on p. 4).

[83] H. Lefebvre, C. Legner, and M. Fadler. "Data democratization: toward a deeper understanding". In: 2021-09 (cit. on p. 2).

[84] K. Li. and C. Zhou. "Estimation of Gait Parameters based on Motion Sensor Data". In: *Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies - BIODEVICES*, INSTICC. SciTePress, 2020, pp. 129–135. ISBN: 978-989-758-398-8. DOI: 10.5220/0008963901290135 (cit. on pp. 1, 33).

[85] J. Lin, R. Khade, and Y. Li. "Rotation-Invariant Similarity in Time Series Using Bag-of-Patterns Representation". In: *J. Intell. Inf. Syst.* 39.2 (2012-10), pp. 287–315. ISSN: 0925-9902. DOI: 10.1007/s10844-012-0196-5. URL: https://doi.org/10.1007/s10844-012-0196-5 (cit. on p. 40).

[86] J. Lin et al. "Experiencing SAX: a novel symbolic representation of time series". In: *Data Mining and Knowledge Discovery* 15.2 (2007-10), pp. 107–144. ISSN: 1573-756X. DOI: 10.1007/s10618-007-0064-z. URL: https://doi.org/10.1007/s10618-007-0064-z (cit. on pp. 15, 16, 38, 40, 137).

[87] H. Liu, Y. Hartmann, and T. Schultz. "CSL-SHARE: A Multimodal Wearable Sensor-Based Human Activity Dataset". In: *Frontiers in Computer Science* 3 (2021-10), p. 90. DOI: 10.3389/fcomp.2021.759136 (cit. on p. 50).

[0] H. Liu, Y. Hartmann, and T. Schultz. "Motion Units: Generalized Sequence Modeling of Human Activities for Sensor-Based Activity Recognition". In: *29th European Signal Processing Conference (EUSIPCO 2021)*. IEEE. 2021. DOI: 10.23919/EUSIPCO54536.2021.9616298 (cit. on p. 33).

[88] H. Liu and T. Schultz. "A Wearable Real-time Human Activity Recognition System using Biosensors Integrated into a Knee Bandage". In: *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 1: BIODEVICES*. INSTICC. SciTePress, 2019, pp. 47–55. ISBN: 978-989-758-353-7 (cit. on p. 33).

[89] H. Liu and T. Schultz. *A Wearable Real-time Human Activity Recognition System using Biosensors Integrated into a Knee Bandage*. 2019-01, pp. 47–55. DOI: 10.5220/0007398800470055 (cit. on p. 50).

[90] H. Liu and T. Schultz. "A Wearable Real-time Human Activity Recognition System using Biosensors Integrated into a Knee Bandage". In: 2019-02, pp. 47–55. DOI: 10.5220/0007398800470055 (cit. on p. 144).

[91] H. Liu and T. Schultz. "How Long Are Various Types of Daily Activities? Statistical Analysis of a Multimodal Wearable Sensor-Based Human Activity Dataset". In: *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 5: HEALTHINF*. 2022, pp. 680–688. ISBN: 978-989-758-552-4 (cit. on pp. 1, 33).

[92] J. Liu et al. "uWave: Accelerometer-based personalized gesture recognition and its applications". In: *Pervasive and Mobile Computing* 5.6 (2009). PerCom 2009, pp. 657–675. ISSN: 1574-1192. DOI: https://doi.org/10.1016/j.pmcj.2009.07.007. URL: https://www.sciencedirect.com/science/article/pii/S1574119209000674 (cit. on pp. 126, 127).

[93] S. Lobov et al. "Latent Factors Limiting the Performance of sEMG-Interfaces". In: *Sensors* 18 (2018-04), p. 1122. DOI: 10.3390/s18041122 (cit. on p. 46).

[94] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[95] C. M. Lu and N. J. Ferrier. "Automated Analysis of Repetitive Joint Motion". In: *IEEE Transactions on Information Technology in Biomedicine* 7.4 (2003-12), pp. 263–273. ISSN: 10897771. DOI: 10.1109/TITB.2003.821309 (cit. on p. 36).

[96] C. M. Lu and N. J. Ferrier. "Repetitive Motion Analysis: Segmentation and Event Classification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (2004-02), pp. 258–263. ISSN: 01628828. DOI: 10.1109/TPAMI.2004.1262196 (cit. on p. 36).

[97] A. Luttmann et al. *Preventing Musculoskeletal Disorders in the Workplace*. World Health Organization, 2003. ISBN: 92 4 159053 X. URL: https://www.who.int/occupational_health/publications/en/oehmsd3.pdf (cit. on p. 3).

[98] R. G. Lyons. *Understanding Digital Signal Processing*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1996. ISBN: 0201634678 (cit. on p. 75).

[99] I. P. Machado et al. "Human activity data discovery from triaxial accelerometer sensor: Non-supervised learning sensitivity to feature extraction parametrization". In: *Information Processing and Management* 51.2 (2015), pp. 204–214. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2014.07.008. URL: http://www.sciencedirect.com/science/article/pii/S0306457314000685 (cit. on p. 36).

[100] R. Malladi, G. P. Kalamangalam, and B. Aazhang. "Online Bayesian change point detection algorithms for segmentation of epileptic activity". In: *2013 Asilomar Conference on Signals, Systems and Computers*. 2013, pp. 1833–1837. DOI: 10.1109/ACSSC.2013.6810619 (cit. on p. 1).

[101] Y. Mangukiya, B. Purohit, and K. George. "Electromyography (EMG) sensor controlled assistive orthotic robotic arm for forearm movement". In: *2017 IEEE Sensors Applications Symposium (SAS)*. IEEE. 2017, pp. 1–4 (cit. on p. 33).

[102] M. Mannino and A. Abouzied. "Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–13. ISBN: 9781450356206. DOI: 10.1145/3173574.3173962. URL: https://doi.org/10.1145/3173574.3173962 (cit. on p. 22).

[103] P. Matias et al. "Time Series Segmentation Using Neural Networks with Cross-Domain Transfer Learning". In: *Electronics* 10 (2021-07), p. 1805. DOI: 10.3390/electronics10151805 (cit. on pp. 32, 34).

[104] L. McAtamney and E. Nigel Corlett. "RULA: a survey method for the investigation of work-related upper limb disorders". In: *Applied Ergonomics* 24.2 (1993), pp. 91–99. ISSN: 0003-6870. DOI: https://doi.org/10.1016/0003-6870(93)90080-S. URL: http://www.sciencedirect.com/science/article/pii/000368709390080S (cit. on p. 4).

[105] T. McNab, D. A. James, and D. Rowlands. "iPhone sensor platforms: Applications to sports monitoring". In: *Procedia Engineering* 13 (2011), pp. 507–512 (cit. on pp. 1, 33).

[106] J. J. A. Mendes Jr et al. "Sensor fusion and smart sensor in sports and biomedical applications". In: *Sensors* 16.10 (2016), p. 1569 (cit. on pp. 1, 33).

[107] H. B. Menz and D. R. Bonanno. "Objective measurement of adherence to wearing foot orthoses using an embedded temperature sensor". In: *Medical Engineering & Physics* 88 (2021), pp. 19–24 (cit. on p. 33).

[108] V. MN. Dam and B. Fitzgerald. *Pulsus Paradoxus*. 2022-01. URL: https://www.ncbi.nlm.nih.gov/books/NBK482292/ (cit. on p. 67).

[109] G. B. Moody, W. K. Muldrow, and R. G. Mark. "A Noise Stress for Arrythmia Detectors". In: *Computers in Cardiology* (1984) (cit. on pp. 1, 47).

[110] M. Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015. ISBN: 978-3-319-21944-8 (cit. on pp. 20, 35, 39, 54, 59, 60, 144).

[111] M. Müller and F. Zalkow. "FMP Notebooks: Educational Material for Teaching and Learning Fundamentals of Music Processing". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*. Delft, The Netherlands, 2019-11 (cit. on pp. 20, 35, 39, 59, 60).

[112] M. Nirmalan and P. M. Dark. "Broader applications of arterial pressure wave form analysis". In: *Continuing Education in Anaesthesia Critical Care and Pain* 14.6 (2014), pp. 285–290. DOI: 10.1093/bjaceaccp/mkt078. eprint: /oup/backfile/content_public/journal/ceaccp/14/6/10.1093/bjaceaccp/mkt078/2/mkt078.pdf. URL: +%20http://dx.doi.org/10.1093/bjaceaccp/mkt078 (cit. on p. 111).

[113] N. Nunes, T. Araújo, and H. Gamboa. "Two-modes Cyclic Biosignal Clustering based on Time Series Analysis." In: 2011-01, pp. 257–264. DOI: 10.13140/2.1.1524.3048 (cit. on p. 36).

[114] M. Nyan, F. E. Tay, and E. Murugasu. "A wearable system for pre-impact fall detection". In: *Journal of biomechanics* 41.16 (2008), pp. 3475–3481 (cit. on p. 33).

[115] E. Occhipinti. "OCRA: a concise index for the assessment of exposure to repetitive movements of the upper limbs". In: *Ergonomics* 41.9 (1998), pp. 1290–1311 (cit. on p. 4).

[116] Y. Ohgi. "Microcomputer-based acceleration sensor device for sports biomechanics-stroke evaluation by using swimmer's wrist acceleration". In: *SENSORS, 2002 IEEE*. Vol. 1. IEEE. 2002, pp. 699–704 (cit. on pp. 1, 33).

[0] S. Patel et al. "A review of wearable sensors and systems with application in rehabilitation". In: *Journal of neuroengineering and rehabilitation* 9.1 (2012), pp. 1–17 (cit. on p. 33).

[117] J. Paulus, M. Müller, and A. Klapuri. "Audio-based Music Structure Analysis". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*. Utrecht, The Netherlands, 2010, pp. 625–636 (cit. on pp. 37, 39, 54, 58).

[118] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 25, 84).

[119] J.-L. Reyes-Ortiz et al. "Transition-Aware Human Activity Recognition Using Smartphones". In: *Neurocomputing* 171 (2015-08). DOI: 10.1016/j.neucom.2015.07.085 (cit. on p. 45).

[120] J. Rodrigues. *Fiber optic sensors for ergonomic analysis in automotive industry*. 2019. URL: https://drive.google.com/file/d/1EHSkt8eQVzLEJUyHJLyTR4JFs-xD7C5I/view?usp=sharing (cit. on p. 134).

[121] J. Rodrigues. *Wearable Technology - Review of Existing Concepts and Ideas for Wearable Technology*. 2019. URL: https://drive.google.com/file/d/1EHSkt8eQVzLEJUyHJLyTR4JFs-xD7C5I/view?usp=sharing (cit. on p. 134).

[122] J. Rodrigues, D. Belo, and H. Gamboa. "Noise detection on ECG based on agglomerative clustering of morphological features". In: *Computers in Biology and Medicine* 87 (2017), pp. 322–334. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2017.06.009. URL: http://www.sciencedirect.com/science/article/pii/S0010482517301737 (cit. on p. 34).

[123] J. Rodrigues et al. "A Genetic Algorithm to Design Job Rotation Schedules with Low Risk Exposure". In: *Technological Innovation for Life Improvement*. Ed. by L. M. Camarinha-Matos et al. Cham: Springer International Publishing, 2020, pp. 395–402. ISBN: 978-3-030-45124-0 (cit. on p. 4).

[124] J. Rodrigues et al. "microErgo: A Concept for an Ergonomic Self-Assessment Tool". In: *2021 Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII)*. 2021, pp. 1–6. DOI: 10.1109/ICBSII51839.2021.9445156 (cit. on p. 139).

[125] J. Rodrigues et al. "SSTS: A syntactic tool for pattern search on time series". In: *Information Processing and Management* 56.1 (2019), pp. 61–76. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2018.09.001. URL: https://www.sciencedirect.com/science/article/pii/S0306457318302577 (cit. on p. 38).

[126] Y. Roh, G. Heo, and S. E. Whang. "A survey on data collection for machine learning: a big data-ai integration perspective". In: *IEEE Transactions on Knowledge and Data Engineering* (2019) (cit. on pp. 1, 34).

[127] D. Romero et al. "Towards an Operator 4.0 Typology: A Human-Centric Perspective on the Fourth Industrial Revolution Technologies". In: *International conference on computers and industrial engineering (CIE46) proceedings*. 2016-10 (cit. on p. 4).

[128] S. Santos et al. "Exploring Inertial Sensor Fusion Methods for Direct Ergonomic Assessments". In: *Biomedical Engineering Systems and Technologies*. Ed. by X. Ye et al. Cham: Springer International Publishing, 2021, pp. 289–303. ISBN: 978-3-030-72379-8 (cit. on pp. 1, 4, 52, 140).

[129] A. Santos. et al. "Self-Similarity Matrix of Morphological Features for Motion Data Analysis in Manufacturing Scenarios". In: *Proceedings of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies - BIOSIGNALS*, INSTICC. SciTePress, 2021, pp. 80–90. ISBN: 978-989-758-490-9. DOI: 10.5220/0010252800800090 (cit. on p. 1).

[130] P. Schäfer. "The BOSS is Concerned with Time Series Classification in the Presence of Noise". In: *Data Min. Knowl. Discov.* 29.6 (2015-11), pp. 1505–1530. ISSN: 1384-5810. DOI: 10.1007/s10618-014-0377-7. URL: https://doi.org/10.1007/s10618-014-0377-7 (cit. on p. 40).

[131] P. Schäfer and M. Högqvist. "SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets". In: *EDBT '12*. 2012 (cit. on p. 40).

[132] P. Schäfer and U. Leser. "Fast and Accurate Time Series Classification with WEASEL". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 637–646. ISBN: 9781450349185. URL: https://doi.org/10.1145/3132847.3132980 (cit. on p. 40).

[133] K. Schaub et al. "The European Assembly Worksheet". In: *Theoretical Issues in Ergonomics Science* 14.6 (2013), pp. 616–639. DOI: 10.1080/1463922X.2012.678283. eprint: https://doi.org/10.1080/1463922X.2012.678283. URL: https://doi.org/10.1080/1463922X.2012.678283 (cit. on p. 4).

[134] S. Schneegaß et al. "A data set of real world driving to assess driver workload". In: 2013-10, pp. 150–157. DOI: 10.1145/2516540.2516561 (cit. on p. 130).

[135] S. Schneegass et al. "A Data Set of Real World Driving to Assess Driver Workload". In: New York, NY, USA: Association for Computing Machinery, 2013. ISBN: 9781450324786. DOI: 10.1145/2516540.2516561. URL: https://doi.org/10.1145/2516540.2516561 (cit. on pp. 49, 131).

[136] P. Senin and S. Malinchik. "SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model". In: *2013 IEEE 13th International Conference on Data Mining*. 2013, pp. 1175–1180. DOI: 10.1109/ICDM.2013.52 (cit. on pp. 41, 85).

[137] G. Shakeri, J. H. Williamson, and S. Brewster. "Novel Multimodal Feedback Techniques for In-Car Mid-Air Gesture Interaction". In: AutomotiveUI '17. Oldenburg, Germany: Association for Computing Machinery, 2017, pp. 84–93. ISBN: 9781450351508. DOI: 10.1145/3122986.3123011. URL: https://doi.org/10.1145/3122986.3123011 (cit. on p. 126).

[138] H. Shatkay. "The Fourier Transform - a Primer". In: *Technical Report CS-95-37*. Brown University, 1995 (cit. on p. 13).

[139] L. Silva et al. "Respiratory Inductance Plethysmography to Assess Fatigability during Repetitive Work". In: *Sensors* 22 (2022-06), p. 4247. DOI: 10.3390/s22114247 (cit. on p. 4).

[140] M. Staudacher et al. "A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep". In: *Physica A: Statistical Mechanics and its Applications* 349.3 (2005), pp. 582–596. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2004.10.026. URL: http://www.sciencedirect.com/science/article/pii/S0378437104013640 (cit. on p. 1).

[0] M. Sung, C. Marci, and A. Pentland. "Wearable feedback systems for rehabilitation". In: *Journal of neuroengineering and rehabilitation* 2.1 (2005), pp. 1–12 (cit. on p. 33).

[141] H. Tankovska. *Global connected wearable devices 2016-2022*. 2020-09. URL: https://www.statista.com/statistics/487291/global-connected-wearable-devices/ (cit. on p. 1).

[142] E. Team. *The exponential growth of data*. 2018-04. URL: https://insidebigdata.com/2017/02/16/the-exponential-growth-of-data (cit. on p. 1).

[143] C. Truong, L. Oudre, and N. Vayatis. "Selective review of offline change point detection methods". In: *Signal Processing* 167 (2020-02), p. 107299. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2019.107299. URL: http://dx.doi.org/10.1016/j.sigpro.2019.107299 (cit. on pp. 32–35, 98).

[144] A. S. Uva et al. *Lesões Musculoesqueléticas Relacionadas com o Trabalho. Guia de Orientação para a Prevenção*. 2008 (cit. on p. 3).

[145] R. Varandas., D. Folgado., and H. Gamboa. "Evaluation of Spatial-Temporal Anomalies in the Analysis of Human Movement". In: *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 4: BIOSIGNALS*, INSTICC. SciTePress, 2019, pp. 163–170. ISBN: 978-989-758-353-7. DOI: 10.5220/0007386701630170 (cit. on pp. 3, 37).

[146] P. Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 21).

[147] Q. Wang et al. "Unsupervised Temporal Segmentation of Repetitive Human Actions Based on Kinematic Modeling and Frequency Analysis". In: *Proceedings - 2015 International Conference on 3D Vision, 3DV 2015*. Institute of Electrical and Electronics Engineers Inc., 2015-11, pp. 562–570. ISBN: 9781467383325. DOI: 10.1109/3DV.2015.69. arXiv: 1512.04115 (cit. on p. 36).

[148] H. Wang., M. Mohamed Refai, and B. F. van Beijnum. "Measuring Upper-Extremity Use with One IMU". In: *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 4: BIOSIGNALS*, INSTICC. SciTePress, 2019, pp. 93–100. ISBN: 978-989-758-353-7. DOI: 10.5220/0007253400930100 (cit. on p. 4).

[149] M. Wattenberg. "Arc diagrams: visualizing structure in strings". In: *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.* 2002, pp. 110–116. DOI: 10.1109/INFVIS.2002.1173155 (cit. on pp. 39, 69).

[150] A. Wege and A. Zimmermann. "Electromyography sensor based control for a hand exoskeleton". In: *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2007, pp. 1470–1475 (cit. on p. 33).

[151] J. Weiner et al. "Bremen Big Data Challenge 2017: Predicting University Cafeteria Load". In: *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer. 2017, pp. 380–386 (cit. on p. 33).

[152] G. Weiss, K. Yoneda, and T. Hayajneh. "Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living". In: *IEEE Access* PP (2019-09), pp. 1–1. DOI: 10.1109/ACCESS.2019.2940729 (cit. on pp. 45, 46).

[153] A. Wilden. "The Rules Are No Game: The Strategy of Communication by Anthony Wilden". In: (1987) (cit. on p. 75).

[154] T. Xue and H. Liu. "Hidden Markov Model and Its Application in Human Activity Recognition and Fall Detection: A Review". In: *Communications, Signal Processing, and Systems*. Springer Singapore, 2022, pp. 863–869. ISBN: 978-981-19-0390-8. DOI: 10.1007/978-981-19-0390-8_108 (cit. on p. 33).

[155] P. Yang, G. Dumont, and J. M. Ansermino. "Adaptive Change Detection in Heart Rate Trend Monitoring in Anesthetized Children". In: *IEEE Transactions on Biomedical Engineering* 53.11 (2006), pp. 2211–2219. DOI: 10.1109/TBME.2006.877107 (cit. on p. 1).

[156] D. Yankov, E. Keogh, and S. Lonardi. "Dot plots for time series analysis". In: *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*. 2005, 10 pp.–168. DOI: 10.1109/ICTAI.2005.60 (cit. on p. 35).

[157] L. Ye and E. Keogh. "Time series shapelets: a new primitive for data mining". In: 2009-06, pp. 947–956. DOI: 10.1145/1557019.1557122 (cit. on p. 40).

[158] C. C. M. Yeh et al. "Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile". In: *Data Mining and Knowledge Discovery* 32.1 (2018-01), pp. 83–123. ISSN: 1573756X. DOI: 10.1007/s10618-017-0519-9. URL: https://doi.org/10.1007/s10618-017-0519-9 (cit. on p. 36).

[159] O. Yuji. "Mems sensor application for the motion analysis in sports science". In: *Memory* 32 (2005), 128Mbit (cit. on pp. 1, 33).

[160] Y. Zhang et al. "Multi-scale signed recurrence plot based time series classification using inception architectural networks". In: *Pattern Recognition* 123 (2022), p. 108385. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2021.108385. URL: https://www.sciencedirect.com/science/article/pii/S0031320321005653 (cit. on p. 35).

[161] C. Zhou et al. "Ankle foot motion recognition based on wireless wearable sEMG and acceleration sensors for smart AFO". In: *Sensors and Actuators A: Physical* 331 (2021), p. 113025 (cit. on p. 33).

[162] C. Zhou et al. "Research and Development of Ankle-Foot Orthoses: A Review". In: *Sensors* 22.17 (2022). ISSN: 1424-8220. DOI: 10.3390/s22176596 (cit. on p. 33).

[163] C. Zhu and W. Sheng. "Human daily activity recognition in robot-assisted living using multi-sensor fusion". In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2009, pp. 2154–2159. ISBN: 9781424427895. DOI: 10.1109/ROBOT.2009.5152756 (cit. on p. 36).

A

TSFEL Feature List of Information Retrieval Experiments

Table A.1: Features applied in this work for creating the SSMs. The Time Series Feature Extraction Library (TSFEL) is utilized for feature extraction.

Temporal domain	**Statistical domain**	**Frequency domain**
Absolute energy	Interquartile Range	Entropy
Area under the curve	Kurtosis	Fundamental frequency
Centroid	Maximum	Max frequency
Cumulative centroid	Mean	Roll off
Distance	Mean absolute deviation	Roll on
Maximum peak	Median	Spectral distance
Mean absolute difference	Minimum	Spectral kurtosis
Mean difference	Root mean square	Spectral skewness
Median absolute difference	Skewness	Spectral spread
Total energy	Standard deviation	
	Variance	

B

Appendix 2 - Detailed Results

B.1 Novelty Segmentation

Table B.1: Parameter configuration and experimental results of each signal in Dataset 3 (see Section 5.3). W_{size}: window size; K%: kernel size in percentage of the window size; O%: overlap percentage of the window size. T%: amplitude threshold for event detection; Novelty: our proposed method with the novelty function; WL: window-based segmentation; BS: binary segmentation. The tolerance used to calculate the F1-score ($F1$) equals the window size for novelty function computation.

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
1	1500	1.5	0.90	0.01	0.91	1500	0.95	0.68
2	1500	1.5	0.90	0.01	0.94	1500	0.95	0.72

Table B.2: Parameter configuration and experimental results of each signal in Dataset 6 (see Section 5.6). W_{size}: window size; K%: kernel size in percentage of the window size; O%: overlap percentage of the window size. T%: amplitude threshold for event detection; Novelty: our proposed method with the novelty function; WL: window-based segmentation; BS: binary segmentation. The tolerance used to calculate the F1-score ($F1$) equals the window size for novelty function computation.

	Novelty					WL		BS
Signal	W_{size}	K(%)	O(%)	T(%)	$F1$	W_{size}	$F1$	$F1$
1	2000	0.62	0.95	0.19	0.90	2000	0.72	0.29
2	2000	1.25	0.95	0.28	0.86	1500	0.75	0.39
3	1500	1.27	0.95	0.10	1.00	2000	0.95	0.78
4	1500	0.63	0.95	0.14	1.00	2000	0.89	0.49
5	2000	1.25	0.95	0.28	0.92	1500	0.85	0.20
6	1500	0.63	0.95	0.28	0.88	1500	0.74	0.29
7	2000	1.25	0.95	0.19	0.95	2000	0.95	0.49
8	1500	1.27	0.95	0.10	0.93	2000	0.85	0.63
9	2000	0.62	0.95	0.28	1.00	2000	0.95	0.44
10	2000	1.25	0.95	0.28	0.97	1500	0.90	0.54
11	1000	1.25	0.95	0.10	1.00	2000	0.96	0.82
12	2000	1.25	0.95	0.14	0.83	2000	0.85	0.34
13	1000	0.62	0.95	0.37	1.00	2000	0.75	0.78
14	1000	1.25	0.95	0.32	0.95	1500	0.65	0.29
15	2000	0.62	0.95	0.14	0.93	1500	0.85	0.54
16	2000	0.62	0.95	0.28	0.90	2000	0.81	0.39
17	1500	2.53	0.95	0.10	0.98	2000	0.83	0.29
18	2000	0.62	0.95	0.50	0.97	1500	0.90	0.49
19	2000	0.62	0.95	0.23	0.93	1500	0.75	0.44
20	1500	0.63	0.95	0.10	0.95	2000	0.95	0.59
21	2000	1.25	0.95	0.19	0.95	2000	0.95	0.44
22	2000	1.25	0.95	0.19	0.84	2000	0.90	0.29
23	1500	1.27	0.95	0.28	0.90	1500	0.85	0.54
24	2000	0.62	0.95	0.19	0.86	2000	0.76	0.39
25	1500	1.27	0.95	0.37	0.95	2000	0.85	0.68
26	1500	0.63	0.95	0.19	0.98	2000	0.90	0.54
27	1500	1.27	0.95	0.19	0.83	1500	0.75	0.59
28	1500	1.27	0.95	0.37	0.90	1500	0.80	0.29
29	1500	1.27	0.95	0.23	1.00	2000	0.76	0.34
30	2000	2.5	0.95	0.10	0.92	2000	0.83	0.20
31	1000	1.25	0.95	0.19	1.00	1500	0.90	0.73
32	1500	1.27	0.95	0.41	0.89	2000	0.77	0.29
33	2000	0.62	0.95	0.23	0.83	1500	0.80	0.63
34	2000	0.62	0.95	0.28	0.93	1000	0.70	0.34
35	1500	0.63	0.95	0.14	0.95	2000	0.84	0.83
36	1000	2.5	0.95	0.14	1.00	1500	0.80	0.29

Table B.3: Parameter configuration and experimental results of each signal in Dataset 7 (see Section 5.7). W_{size}: window size; K%: kernel size in percentage of the window size; O%: overlap percentage of the window size. T%: amplitude threshold for event detection; Novelty: our proposed method with the novelty function; WL: window-based segmentation; BS: binary segmentation. The tolerance used to calculate the F1-score ($F1$) equals the window size for novelty function computation.

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
1	1500	6.33	0.95	0.14	1.00	5000	0.93	0.97
2	5000	6.25	0.95	0.19	0.97	5000	0.79	0.97
3	5000	12.5	0.95	0.23	0.80	5000	0.57	0.83
4	3500	12.57	0.95	0.23	0.50	3500	0.21	0.48
5	5000	12.5	0.95	0.23	0.42	3500	0.14	0.34
6	1500	6.33	0.95	0.10	1.00	2500	0.93	0.90
7	2500	2.52	0.95	0.28	1.00	3500	0.86	0.90
8	2500	6.3	0.95	0.19	1.00	3500	0.64	0.90
9	2500	6.3	0.95	0.28	0.87	2500	0.29	0.55
10	5000	6.25	0.95	0.28	0.63	5000	0.21	0.34
11	5000	6.25	0.95	0.37	0.36	2500	0.07	0.34
12	2500	2.52	0.95	0.19	1.00	3500	0.93	0.90

Table B.4: Parameter configuration and experimental results of each signal in Dataset 8 (see Section 5.8). W_{size}: window size; K%: kernel size in percentage of the window size; O%: overlap percentage of the window size. T%: amplitude threshold for event detection; Novelty: our proposed method with the novelty function; WL: window-based segmentation; BS: binary segmentation. The tolerance used to calculate the F1-score ($F1$) equals the window size for novelty function computation.

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
1	400	3.74	0.95	0.55	1.00	2000	0.67	0.40
2	400	2.49	0.95	0.68	0.80	400	1.00	0.40
3	150	4	0.95	0.82	0.80	150	0.00	0.00
4	150	2.66	0.95	0.32	1.00	2000	0.67	0.80
5	300	2.66	0.95	0.32	1.00	300	1.00	0.80
6	300	2.66	0.95	0.59	1.00	1000	0.67	0.80
7	150	2.66	0.95	0.64	1.00	2000	0.67	0.80
8	150	2.66	0.95	0.28	1.00	150	0.50	0.40
9	150	2.66	0.95	0.55	1.00	150	0.00	0.80

Table B.5: Parameter configuration and experimental results of each signal in Dataset 13 (see Section 5.13). W_{size}: window size; K%: kernel size in percentage of the window size; O%: overlap percentage of the window size. T%: amplitude threshold for event detection; Novelty: our proposed method with the novelty function; WL: window-based segmentation; BS: binary segmentation. The tolerance used to calculate the F1-score ($F1$) equals the window size for novelty function computation.

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
0	2000	2.49	0.36	0.95	0.93	2500	0.88	0.62
1	2500	1.25	0.01	0.95	0.88	2500	0.71	0.75
2	2000	1.25	0.16	0.95	0.89	2500	0.86	0.75
3	2500	1.26	0.41	0.95	0.80	2500	0.88	0.75
4	2500	1.25	0.31	0.95	0.88	2000	0.75	0.75
5	2000	1.25	0.31	0.95	0.88	2000	0.86	0.71
6	2500	0.75	0.50	0.95	1.00	2500	0.86	0.88
7	2000	1.25	0.31	0.95	0.94	2000	0.93	0.71
8	2000	1.87	0.01	0.95	0.93	2000	0.71	0.75
9	2500	1.25	0.01	0.95	0.88	2000	0.80	0.62
10	2500	1.26	0.01	0.95	0.93	2000	0.86	0.71
11	2000	1.25	0.50	0.95	0.88	2000	0.88	0.62
12	2500	0.75	0.21	0.95	0.93	2000	0.86	0.82
13	1000	1.87	0.41	0.95	0.89	2000	0.77	0.62
14	2500	0.38	0.41	0.95	0.93	2000	0.75	0.71
15	2000	1.25	0.21	0.95	0.86	2000	0.88	0.71
16	2500	0.75	0.46	0.95	0.93	2000	0.86	0.75
17	2000	1.25	0.21	0.95	0.93	2500	0.86	0.62
18	2500	0.38	0.36	0.95	1.00	2500	0.86	0.59
19	2500	1.26	0.16	0.95	0.86	2000	0.88	0.62
20	2500	0.50	0.41	0.95	0.89	2500	0.77	0.82
21	2000	0.62	0.36	0.95	0.84	2000	0.80	0.67
22	2000	1.25	0.26	0.95	0.88	2000	0.71	0.71
23	2000	1.25	0.41	0.95	0.93	2000	0.86	0.71
24	2500	0.63	0.46	0.95	0.93	2000	0.80	0.75
25	2000	1.25	0.21	0.95	0.86	2500	0.77	0.71
26	2000	1.25	0.21	0.95	0.93	2500	0.77	0.75
27	2000	0.50	0.46	0.95	0.88	2000	0.57	0.88
28	2500	0.75	0.16	0.95	0.88	2500	0.77	0.82
29	2000	0.75	0.31	0.95	0.94	2000	0.71	0.71

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	$K\%$	$O\%$	$T\%$	$F1$	W_{size}	$F1$	$F1$
30	2000	0.50	0.46	0.95	0.88	2000	0.77	0.88
31	2000	1.25	0.11	0.95	0.86	2500	0.62	0.59
32	2000	1.25	0.16	0.95	0.89	2000	0.86	0.88
33	2000	0.37	0.36	0.95	0.86	2000	0.77	0.82
34	2000	0.75	0.36	0.95	0.88	2000	0.88	0.75
35	1000	1.25	0.36	0.95	0.82	2500	0.67	0.62
36	2500	0.63	0.11	0.95	0.80	2500	0.77	0.82
37	2000	0.37	0.41	0.95	0.80	2000	0.67	0.82
38	1000	1.87	0.41	0.95	0.94	2000	0.53	0.62
39	2500	1.26	0.01	0.95	0.80	2500	0.67	0.59
40	2000	0.37	0.36	0.95	0.93	2000	0.80	0.62
41	2000	0.62	0.36	0.95	0.80	2000	0.71	0.59
42	1000	1.25	0.46	0.95	0.89	2000	0.77	0.62
43	2500	0.75	0.01	0.95	0.88	2500	0.67	0.71
44	2000	0.50	0.46	0.95	1.00	2500	0.67	0.88
45	2000	0.75	0.31	0.95	0.80	2000	0.67	0.59
46	2500	0.50	0.21	0.95	0.93	2000	0.77	0.88
47	2000	0.62	0.36	0.95	0.88	2000	0.77	0.56
48	2500	0.25	0.36	0.95	0.93	1000	0.75	0.88
49	2500	1.25	0.01	0.95	0.92	1000	0.57	0.67
50	2000	1.25	0.16	0.95	0.92	2500	0.67	0.71
51	1000	1.87	0.26	0.95	0.80	2500	0.67	0.62
52	2500	0.38	0.26	0.95	0.88	2000	0.80	0.59
53	2000	0.75	0.31	0.95	0.88	2000	0.80	0.88
54	2500	0.38	0.31	0.95	1.00	2000	0.86	0.67
55	2500	1.26	0.01	0.95	0.86	2000	0.80	0.62
56	2000	0.75	0.16	0.95	0.88	2000	0.80	0.62
57	2500	0.75	0.31	0.95	0.57	2500	0.71	0.50
58	2000	1.87	0.06	0.95	0.93	2000	0.75	0.71
59	2000	0.50	0.26	0.95	0.78	2000	0.67	0.71
60	2000	0.37	0.31	0.95	0.88	2500	0.77	0.62
61	2000	1.25	0.26	0.95	0.86	2000	0.88	0.62
62	2000	1.25	0.16	0.95	0.86	2500	0.77	0.75
63	2000	1.25	0.01	0.95	0.86	2500	0.77	0.62
64	2500	0.38	0.26	0.95	0.94	2500	0.77	0.71
65	2000	1.87	0.01	0.95	0.80	2000	0.80	0.62

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
66	2500	0.38	0.31	0.95	1.00	2000	0.88	0.88
67	2500	1.25	0.01	0.95	0.86	2000	0.77	0.62
68	2500	0.38	0.36	0.95	0.93	2000	0.88	0.75
69	2000	1.25	0.26	0.95	0.86	2000	0.80	0.75
70	2500	0.63	0.16	0.95	1.00	2000	0.86	0.62
71	2500	1.26	0.01	0.95	0.86	2000	0.80	0.67
72	2500	0.75	0.16	0.95	0.93	2000	0.88	0.88
73	2000	1.25	0.11	0.95	0.86	2000	0.88	0.62
74	2000	0.37	0.36	0.95	0.88	2500	0.77	0.75
75	2000	0.62	0.26	0.95	0.89	2500	0.77	0.71
76	2000	0.62	0.16	0.95	1.00	2500	0.77	0.62
77	2000	0.62	0.26	0.95	0.93	2000	0.71	0.75
78	2000	0.62	0.26	0.95	0.94	2000	0.77	0.62
79	2000	0.50	0.36	0.95	0.93	2000	0.80	0.62
80	2500	0.50	0.16	0.95	0.94	2000	0.86	0.75
81	2500	0.50	0.31	0.95	0.88	2000	0.77	0.59
82	2500	0.25	0.26	0.95	0.93	2000	0.86	0.59
83	2000	1.25	0.26	0.95	0.86	2500	0.77	0.59
84	2500	0.38	0.26	0.95	0.93	2000	0.71	0.71
85	2000	1.25	0.11	0.95	0.86	2000	0.80	0.75
86	2000	0.75	0.16	0.95	0.86	2000	0.71	0.62
87	2000	1.25	0.16	0.95	0.86	2000	0.77	0.62
88	2500	0.38	0.31	0.95	0.94	2000	0.80	0.75
89	2000	0.62	0.31	0.95	0.88	2000	0.86	0.62
90	2500	0.50	0.36	0.95	0.89	2000	0.86	0.78
91	2500	1.25	0.11	0.95	0.80	2500	0.77	0.56
92	2500	0.63	0.11	0.95	0.93	2000	0.86	0.56
93	2000	0.62	0.21	0.95	0.88	2000	0.71	0.59
94	2000	0.50	0.46	0.95	0.93	2000	0.71	0.67
95	2000	0.62	0.41	0.95	0.89	2000	0.77	0.62
96	2500	0.50	0.21	0.95	0.93	2000	0.77	0.88
97	2000	0.75	0.36	0.95	0.93	2000	0.86	0.62
98	2000	0.62	0.21	0.95	0.94	2000	0.86	0.71
99	2000	1.25	0.16	0.95	0.86	2000	0.77	0.59
100	2000	0.50	0.46	0.95	0.93	2000	0.86	0.75
101	2000	1.25	0.16	0.95	0.86	2000	0.86	0.62

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	$K\%$	$O\%$	$T\%$	$F1$	W_{size}	$F1$	$F1$
102	2500	1.26	0.36	0.95	0.93	2500	0.88	0.75
103	2000	1.25	0.31	0.95	0.84	2000	0.62	0.38
104	2000	0.62	0.36	0.95	0.94	2000	0.88	0.75
105	2000	0.62	0.41	0.95	0.88	1000	0.59	0.75
106	2000	1.25	0.26	0.95	0.93	2000	0.88	0.62
107	2500	0.75	0.21	0.95	1.00	2500	0.77	1.00
108	1000	2.50	0.16	0.95	0.93	2000	0.62	0.62
109	2000	0.25	0.36	0.95	0.93	2000	0.80	0.62
110	2500	0.13	0.26	0.95	1.00	2000	0.88	0.75
111	2000	1.25	0.01	0.95	0.88	2000	0.88	0.59
112	2000	0.62	0.46	0.95	0.88	2000	0.86	0.47
113	2000	0.50	0.41	0.95	0.89	2000	0.77	0.62
114	2000	0.62	0.16	0.95	0.94	2000	0.86	0.75
115	1000	2.50	0.26	0.95	0.88	2000	0.67	0.88
116	2000	1.25	0.01	0.95	0.88	2000	0.80	0.75
117	1000	1.87	0.26	0.95	0.88	2500	0.57	0.50
118	2000	0.62	0.16	0.95	0.89	2000	0.71	0.75
119	2500	0.75	0.21	0.95	0.88	2500	0.77	0.62
120	2000	1.25	0.26	0.95	1.00	2000	0.75	0.62
121	2500	0.63	0.16	0.95	0.94	2000	0.75	0.62
122	2500	0.63	0.16	0.95	0.94	2000	0.88	0.59
123	2500	1.26	0.01	0.95	0.86	2000	0.82	0.56
124	2500	0.25	0.31	0.95	1.00	2000	0.86	0.88
125	2000	1.25	0.16	0.95	0.88	2000	0.88	0.75
126	2000	1.25	0.26	0.95	0.93	2000	0.86	0.75
127	2000	1.25	0.26	0.95	0.93	2000	0.80	0.62
128	2000	0.75	0.21	0.95	0.94	2000	0.88	0.75
129	2500	0.75	0.01	0.95	0.89	2000	0.88	0.75
130	2500	1.26	0.01	0.95	0.86	2500	0.86	0.75
131	2000	1.87	0.01	0.95	0.80	2500	0.86	0.50
132	2000	1.87	0.01	0.95	0.86	2000	0.88	0.62
133	2500	0.38	0.31	0.95	0.88	2000	0.71	0.75
134	2500	0.63	0.26	0.95	0.88	2000	0.77	0.59
135	2500	0.25	0.26	0.95	0.93	2000	0.77	0.88
136	2500	0.50	0.21	0.95	0.89	2000	0.67	0.59
137	2000	0.62	0.31	0.95	0.82	2000	0.86	0.62

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
138	2500	0.63	0.16	0.95	0.94	2000	0.93	0.62
139	2500	0.63	0.26	0.95	0.94	2000	0.82	0.59
140	2500	0.75	0.41	0.95	0.94	2000	0.94	0.82
141	2000	1.25	0.31	0.95	0.86	2000	0.80	0.62
142	2500	0.50	0.16	0.95	0.89	2000	0.80	0.59
143	1000	2.50	0.26	0.95	0.88	2000	0.57	0.50
144	2500	0.63	0.16	0.95	0.94	2000	0.67	0.75
145	2500	0.63	0.11	0.95	0.84	2000	0.93	0.62
146	2500	0.75	0.21	0.95	0.80	2000	0.75	0.62
147	2500	2.51	0.01	0.95	0.93	2000	0.88	0.62
148	2500	1.88	0.01	0.95	0.93	2000	0.75	0.62
149	2000	0.62	0.46	0.95	0.88	2000	0.88	0.62
150	2500	0.13	0.31	0.95	0.86	2500	0.88	0.62
151	2000	0.50	0.46	0.95	0.88	2000	0.88	0.62
152	2500	1.26	0.46	0.95	0.93	2500	0.93	0.62
153	2000	1.87	0.11	0.95	0.88	2000	0.67	0.75
154	2500	1.88	0.01	0.95	0.93	2000	0.75	0.50
155	2500	0.75	0.01	0.95	0.84	2500	0.86	0.62
156	2500	1.25	0.01	0.95	0.94	2500	0.77	0.62
157	2500	1.88	0.01	0.95	0.86	2500	0.93	0.62
158	2500	0.63	0.26	0.95	0.84	2000	0.88	0.62
159	2500	1.88	0.01	0.95	0.80	2500	0.67	0.62
160	2000	1.87	0.01	0.95	0.88	2500	0.93	0.62
161	2500	2.51	0.01	0.95	0.86	2500	0.80	0.62
162	2000	1.25	0.16	0.95	0.94	2000	0.88	0.88
163	2000	0.62	0.46	0.95	0.89	2000	1.00	0.75
164	2000	1.25	0.21	0.95	0.89	2500	0.93	0.62
165	2500	0.50	0.16	0.95	0.80	2500	0.80	0.62
166	2500	1.26	0.41	0.95	0.93	2500	0.86	0.62
167	2500	0.75	0.01	0.95	0.82	2500	0.93	0.62
168	2000	1.87	0.01	0.95	0.88	2000	0.88	0.62
169	2500	1.88	0.01	0.95	0.86	2000	0.80	0.59
170	2500	1.26	0.36	0.95	0.86	2000	0.88	0.59
171	2000	1.25	0.36	0.95	0.82	2000	0.93	0.62
172	2500	1.26	0.01	0.95	0.80	2500	0.93	0.75
173	2000	1.25	0.41	0.95	0.88	2000	0.88	0.75

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	$K\%$	$O\%$	$T\%$	$F1$	W_{size}	$F1$	$F1$
174	2000	1.25	0.11	0.95	0.82	2000	0.75	0.75
175	2500	0.75	0.11	0.95	0.82	2500	0.86	0.62
176	2000	1.87	0.01	0.95	0.82	2500	0.88	0.62
177	2000	1.87	0.01	0.95	0.74	1000	0.62	0.62
178	2500	0.13	0.21	0.95	0.88	2500	0.93	0.75
179	2500	0.75	0.01	0.95	0.94	2000	0.88	0.75
180	2500	1.26	0.01	0.95	0.88	2000	0.86	0.71
181	2000	1.25	0.46	0.95	0.88	2500	0.80	0.62
182	2500	1.26	0.01	0.95	1.00	2000	0.86	0.59
183	2500	0.63	0.55	0.95	0.86	2000	0.86	0.62
184	2000	1.25	0.01	0.95	0.82	2500	0.93	0.75
185	2500	0.75	0.01	0.95	0.82	2000	0.88	0.75
186	2000	1.25	0.11	0.95	0.82	2500	0.82	0.78
187	2500	1.26	0.11	0.95	0.82	2000	0.88	0.67
188	2000	1.25	0.11	0.95	0.88	2500	0.93	0.75
189	2000	1.25	0.41	0.95	0.82	2500	0.67	0.75
190	2000	1.87	0.01	0.95	1.00	2000	0.88	0.75
191	2500	1.26	0.01	0.95	0.80	2000	0.86	0.62
192	2500	1.25	0.31	0.95	0.82	2000	0.80	0.67
193	2500	0.38	0.31	0.95	1.00	2000	0.80	0.82
194	2000	1.87	0.16	0.95	0.93	2000	0.93	0.62
195	2500	0.75	0.41	0.95	1.00	2500	0.86	0.71
196	2000	0.62	0.31	0.95	0.89	2500	0.71	0.62
197	2500	1.26	0.01	0.95	0.93	2500	0.86	0.75
198	2000	1.25	0.31	0.95	0.88	2500	0.86	0.62
199	2500	0.38	0.31	0.95	0.94	2000	0.88	0.75
200	2000	1.25	0.46	0.95	0.80	2000	0.88	0.75
201	2500	1.25	0.01	0.95	0.93	2000	0.88	0.71
202	2500	1.26	0.01	0.95	0.93	2000	0.75	0.75
203	2500	0.50	0.36	0.95	0.93	2000	0.82	0.71
204	2000	1.25	0.26	0.95	0.84	2500	0.86	0.62
205	2500	0.75	0.50	0.95	0.93	2000	0.88	0.78
206	2000	1.25	0.50	0.95	0.82	2000	0.86	0.75
207	2500	0.50	0.41	0.95	0.93	2000	0.86	0.82
208	2000	1.25	0.11	0.95	0.82	2500	0.86	0.62
209	2500	0.38	0.31	0.95	1.00	2000	0.88	0.71

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
210	2500	0.63	0.31	0.95	0.88	2000	0.86	0.71
211	2500	0.75	0.41	0.95	1.00	2000	0.88	0.71
212	2000	0.62	0.31	0.95	0.80	2000	0.80	0.62
213	2500	0.75	0.41	0.95	1.00	2000	0.88	0.88
214	2000	1.25	0.01	0.95	0.82	2000	0.86	0.62
215	2500	0.75	0.16	0.95	1.00	2500	0.86	0.71
216	2000	0.50	0.36	0.95	0.89	2000	0.88	0.62
217	2000	0.62	0.50	0.95	0.89	2000	1.00	0.75
218	2000	1.87	0.41	0.95	0.86	2000	0.86	0.62
219	2500	1.25	0.16	0.95	0.93	2500	0.86	0.71
220	2000	0.62	0.31	0.95	0.82	2000	0.88	0.62
221	2500	0.75	0.21	0.95	1.00	2500	0.88	0.78
222	2000	0.50	0.50	0.95	0.82	2000	0.80	0.71
223	2500	0.38	0.41	0.95	0.93	2500	0.86	1.00
224	2500	1.26	0.01	0.95	0.93	2000	0.86	0.75
225	2000	0.75	0.36	0.95	0.94	2000	0.80	0.62
226	2500	1.26	0.01	0.95	0.84	2500	0.86	0.62
227	2500	0.75	0.50	0.95	1.00	2000	0.88	0.82
228	2500	0.25	0.21	0.95	0.82	2500	0.71	0.50
229	2000	1.87	0.01	0.95	1.00	2000	0.86	0.75
230	2500	1.25	0.46	0.95	0.86	2000	0.93	0.88
231	2500	0.63	0.36	0.95	1.00	2000	0.80	0.71
232	2000	0.50	0.41	0.95	0.82	2000	0.71	0.62
233	2500	0.50	0.16	0.95	1.00	2000	0.88	0.71
234	2000	1.25	0.21	0.95	0.82	2000	0.88	0.71
235	2500	1.26	0.11	0.95	0.88	2000	0.88	0.50
236	2000	1.25	0.46	0.95	0.93	2500	0.80	0.75
237	2500	0.63	0.21	0.95	1.00	2000	0.93	0.62
238	2000	1.25	0.31	0.95	0.93	2500	0.86	0.71
239	2500	0.50	0.41	0.95	0.94	2000	0.86	0.88
240	1000	2.50	0.55	0.95	0.94	2000	0.62	0.62
241	2500	0.25	0.31	0.95	1.00	2000	0.88	0.59
242	2500	0.63	0.46	0.95	0.93	2000	0.80	0.56
243	2500	0.25	0.21	0.95	0.89	2000	0.88	0.75
244	2000	2.49	0.01	0.95	0.93	2500	0.80	0.62
245	2500	0.63	0.46	0.95	1.00	2500	0.86	0.67

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
246	2000	0.25	0.36	0.95	0.89	2000	0.71	0.71
247	2000	0.25	0.46	0.95	0.80	2500	1.00	0.75
248	2000	0.25	0.31	0.95	0.67	2500	0.67	0.50
249	2000	0.50	0.36	0.95	0.89	2000	0.88	0.62
250	2000	1.25	0.46	0.95	0.88	2000	0.80	0.62
251	2000	0.75	0.41	0.95	0.94	2000	0.62	0.38
252	2500	0.25	0.31	0.95	0.89	2000	0.93	0.59
253	2000	0.37	0.41	0.95	0.89	2000	0.93	0.75
254	2000	1.25	0.11	0.95	0.88	2500	0.88	0.62
255	2500	0.63	0.46	0.95	1.00	2000	0.71	0.62
256	2000	0.50	0.31	0.95	0.88	2000	0.93	0.62
257	2500	0.75	0.11	0.95	1.00	2000	0.88	0.59
258	2000	0.50	0.26	0.95	0.89	2500	0.86	0.62
259	2000	0.37	0.46	0.95	0.88	2000	0.93	0.88
260	2000	0.25	0.36	0.95	0.88	2000	0.88	0.62
261	2500	0.63	0.50	0.95	1.00	1000	0.88	0.75
262	2000	0.75	0.26	0.95	0.88	2000	0.88	0.75
263	2500	0.38	0.41	0.95	1.00	2000	0.93	0.50
264	2500	0.63	0.01	0.95	0.88	2000	0.80	0.62
265	2000	0.75	0.46	0.95	1.00	2000	0.86	0.50
266	2000	1.25	0.21	0.95	0.89	2000	0.93	0.62
267	2000	0.50	0.46	0.95	1.00	2000	0.86	0.75
268	2000	1.25	0.31	0.95	0.88	2000	0.93	0.62
269	2500	0.38	0.41	0.95	1.00	2500	0.67	0.62
270	2000	1.25	0.21	0.95	0.88	2000	0.86	0.82
271	2500	0.75	0.11	0.95	0.94	2000	0.86	0.75
272	2000	0.75	0.16	0.95	0.82	2000	0.88	0.67
273	2500	0.50	0.21	0.95	0.89	2000	0.88	0.71
274	2000	1.87	0.01	0.95	0.93	2000	0.86	0.62
275	2500	0.63	0.41	0.95	1.00	2000	0.93	0.62
276	2000	1.87	0.41	0.95	0.86	2000	0.86	0.62
277	2000	1.25	0.36	0.95	0.88	2500	0.67	0.50
278	2500	0.75	0.21	0.95	0.91	2000	0.80	0.71
279	2000	1.87	0.01	0.95	0.86	2500	0.86	0.71
280	2500	0.25	0.31	0.95	1.00	2000	0.86	0.62
281	2000	0.50	0.41	0.95	0.82	2000	0.88	0.62

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
282	2500	0.38	0.36	0.95	1.00	2000	0.80	0.62
283	2500	1.25	0.01	0.95	0.93	2000	0.88	0.62
284	2000	0.37	0.26	0.95	0.82	1000	0.88	0.50
285	2500	1.25	0.01	0.95	0.91	2500	0.80	0.67
286	2500	0.63	0.11	0.95	1.00	2000	0.82	0.63
287	2000	1.25	0.01	0.95	0.82	2500	0.71	0.56
288	2000	1.87	0.01	0.95	0.88	2500	0.93	0.50
289	2500	0.38	0.26	0.95	0.94	1000	1.00	0.62
290	2000	0.75	0.16	0.95	0.82	2000	0.88	0.62
291	2000	1.25	0.31	0.95	0.88	2000	0.93	0.62
292	2500	0.63	0.26	0.95	0.94	2000	0.80	0.75
293	2000	1.24	0.01	0.95	0.88	2500	0.86	0.62
294	2500	0.38	0.16	0.95	0.94	2500	0.80	0.75
295	2500	1.26	0.01	0.95	0.93	2500	0.86	0.62
296	2500	0.50	0.26	0.95	0.94	2000	0.82	0.62
297	2000	1.25	0.01	0.95	0.82	2500	0.80	0.62
298	2500	0.38	0.16	0.95	0.94	2000	0.88	0.62
299	2000	1.87	0.26	0.95	0.86	2500	0.93	0.62
300	2000	1.25	0.01	0.95	0.88	2000	0.93	0.62
301	2500	0.63	0.16	0.95	0.89	2500	0.86	0.71
302	2000	0.62	0.46	0.95	0.88	2000	0.80	0.75
303	2500	0.75	0.11	0.95	0.89	2500	0.86	0.75
304	2500	0.75	0.46	0.95	0.93	2000	0.86	0.75
305	2000	1.87	0.21	0.95	0.88	2000	0.86	0.62
306	2000	1.25	0.31	0.95	0.88	2000	0.88	0.62
307	2000	1.87	0.01	0.95	0.93	2000	0.93	0.62
308	2500	0.75	0.41	0.95	1.00	2000	0.86	0.88
309	2000	1.25	0.36	0.95	0.93	2000	0.75	0.62
310	1000	2.50	0.11	0.95	0.88	2500	0.86	0.50
311	1000	1.87	0.21	0.95	0.89	2500	0.71	0.62
312	2000	1.25	0.26	0.95	1.00	2000	0.93	0.62
313	2000	1.25	0.50	0.95	0.93	2000	0.93	0.62
314	2000	1.25	0.46	0.95	0.93	2000	0.86	0.62
315	2500	1.26	0.21	0.95	1.00	2500	0.93	0.62
316	2000	1.25	0.01	0.95	0.88	2000	0.86	0.62
317	2500	0.75	0.11	0.95	0.94	2000	0.86	0.75

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
318	2500	0.75	0.11	0.95	0.94	2500	0.88	0.75
319	1000	1.87	0.16	0.95	0.89	2000	0.86	0.62
320	2500	0.63	0.31	0.95	0.94	2000	0.88	0.75
321	2000	0.75	0.36	0.95	0.89	2000	0.93	0.62
322	2500	0.50	0.21	0.95	0.94	2000	0.88	0.62
323	2000	0.75	0.26	0.95	0.88	2000	0.88	0.62
324	2500	0.75	0.01	0.95	0.89	1000	0.82	0.62
325	2000	1.25	0.26	0.95	0.88	2000	0.93	0.62
326	2500	0.50	0.41	0.95	1.00	1000	0.88	0.75
327	2500	0.75	0.11	0.95	0.88	2000	0.86	0.62
328	2500	0.50	0.41	0.95	0.74	2000	0.75	0.75
329	2500	1.26	0.26	0.95	0.82	2000	0.75	0.62
330	2500	0.50	0.26	0.95	0.94	2000	0.75	0.62
331	2000	0.75	0.26	0.95	1.00	2000	0.93	0.62
332	2000	0.37	0.36	0.95	0.88	2500	0.93	0.59
333	2500	0.63	0.41	0.95	0.89	2000	0.86	0.75
334	2000	1.25	0.11	0.95	0.88	2500	0.93	0.59
335	2000	0.50	0.41	0.95	0.94	2500	0.86	0.75
336	2500	1.88	0.01	0.95	0.86	2000	0.88	0.71
337	2000	0.37	0.41	0.95	0.88	1000	0.88	0.62
338	2000	1.25	0.31	0.95	0.88	2500	0.93	0.62
339	2500	0.63	0.31	0.95	0.88	2500	0.80	0.62
340	2000	1.25	0.31	0.95	0.80	2500	0.86	0.62
341	2500	1.26	0.01	0.95	0.88	2500	0.86	0.62
342	2500	0.38	0.36	0.95	0.80	2000	0.93	0.59
343	2000	0.62	0.36	0.95	0.84	2500	0.80	0.62
344	2000	0.62	0.31	0.95	0.73	2000	0.88	0.78
345	2000	1.25	0.01	0.95	0.82	2500	0.93	0.75
346	2500	0.75	0.46	0.95	0.82	2000	0.93	0.71
347	2000	1.25	0.01	0.95	0.88	2000	0.88	0.62
348	2000	1.25	0.11	0.95	0.88	2000	0.88	0.75
349	2000	0.37	0.41	0.95	0.89	2000	0.88	0.71
350	2000	0.62	0.31	0.95	0.89	2000	0.93	0.75
351	2500	0.50	0.26	0.95	0.82	2000	0.80	0.71
352	2500	0.63	0.31	0.95	0.89	2000	0.86	0.67
353	2000	0.37	0.36	0.95	0.74	2500	0.80	0.62

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
354	2000	0.62	0.21	0.95	0.82	2000	0.82	0.59
355	2000	1.87	0.01	0.95	0.88	2500	0.93	0.62
356	2500	1.26	0.01	0.95	0.93	2000	0.88	0.62
357	2500	1.26	0.01	0.95	0.88	2500	0.88	0.62
358	2500	0.63	0.41	0.95	1.00	2000	0.93	0.62
359	2000	0.75	0.31	0.95	0.82	2000	0.88	0.62
360	2500	0.75	0.41	0.95	1.00	2000	0.93	0.59
361	2000	1.25	0.41	0.95	0.88	2000	0.88	0.71
362	2000	1.87	0.01	0.95	0.82	2000	0.89	0.67
363	2500	0.75	0.16	0.95	0.82	2000	0.93	0.62
364	2500	0.38	0.36	0.95	1.00	2500	0.86	0.62
365	2000	0.75	0.11	0.95	0.82	2000	0.75	0.62
366	2000	1.25	0.16	0.95	0.88	2000	0.88	0.50
367	2000	1.25	0.01	0.95	0.88	2500	0.93	0.62
368	2000	1.25	0.11	0.95	0.82	2000	0.88	0.62
369	2000	0.75	0.16	0.95	0.80	2000	0.80	0.71
370	2500	1.25	0.01	0.95	0.93	2500	0.93	0.62
371	2500	1.26	0.01	0.95	0.86	2000	0.88	0.56
372	2500	0.50	0.21	0.95	0.94	2500	0.93	0.62
373	2000	1.25	0.41	0.95	0.88	2500	0.93	0.62
374	2000	1.25	0.21	0.95	0.88	2000	0.88	0.62
375	2500	0.63	0.01	0.95	0.80	2000	0.88	0.62
376	2500	1.26	0.01	0.95	0.82	2500	0.80	0.62
377	2500	0.63	0.26	0.95	0.84	2000	0.89	0.63
378	2500	0.75	0.16	0.95	0.94	2000	0.86	0.71
379	2000	1.25	0.26	0.95	0.78	2500	0.80	0.62
380	2500	1.25	0.11	0.95	0.93	2000	0.88	0.62
381	2000	1.25	0.26	0.95	0.82	2000	0.88	0.62
382	2000	0.62	0.26	0.95	0.84	2000	0.86	0.71
383	2500	0.75	0.26	0.95	0.84	2000	0.94	0.62
384	2500	0.25	0.26	0.95	0.94	2500	0.86	0.75
385	2000	0.75	0.26	0.95	0.78	2500	0.93	0.62
386	2000	1.87	0.01	0.95	0.80	2000	0.93	0.62
387	2000	1.25	0.01	0.95	0.82	2000	0.86	0.47
388	2500	0.75	0.46	0.95	0.88	2000	0.93	0.62
389	2000	1.87	0.26	0.95	0.88	2500	0.88	0.62

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
390	1000	0.75	0.65	0.95	0.80	2500	0.53	0.50
391	2000	1.25	0.16	0.95	0.82	2500	0.93	0.59
392	2500	0.63	0.16	0.95	0.89	2000	0.88	0.88
393	2500	0.75	0.46	0.95	0.88	2500	0.93	0.62
394	2500	0.38	0.36	0.95	0.93	2000	0.93	0.62
395	2500	1.25	0.16	0.95	0.80	2000	0.86	0.59
396	2500	0.38	0.36	0.95	0.82	2000	0.80	0.62
397	2000	1.25	0.01	0.95	0.89	2500	0.93	0.62
398	2000	1.25	0.11	0.95	0.88	2500	0.80	0.62
399	2500	1.25	0.01	0.95	0.80	2500	0.80	0.62
400	2000	1.25	0.11	0.95	1.00	2500	0.88	0.62
401	2000	1.25	0.26	0.95	0.88	2000	0.80	0.62
402	2000	1.87	0.01	0.95	0.93	2000	0.88	0.62
403	2000	0.25	0.36	0.95	0.75	2000	0.86	0.62
404	2500	0.63	0.16	0.95	0.94	2500	0.88	0.75
405	2000	1.87	0.01	0.95	0.88	2000	0.80	0.62
406	2500	1.25	0.01	0.95	1.00	2500	0.80	0.75
407	2500	0.75	0.31	0.95	0.82	2500	0.86	0.62
408	2500	0.50	0.36	0.95	0.94	2000	0.88	0.75
409	2000	1.25	0.21	0.95	0.74	2500	0.86	0.62
410	2500	0.75	0.11	0.95	0.94	2500	0.86	0.75
411	2500	0.38	0.26	0.95	0.74	2000	0.86	0.62
412	2000	1.25	0.16	0.95	0.88	2500	0.88	0.75
413	2000	1.25	0.16	0.95	0.88	2000	0.80	0.62
414	2500	0.75	0.21	0.95	0.94	2000	0.93	0.88
415	2000	1.25	0.31	0.95	0.88	2000	0.86	0.62
416	2500	1.26	0.01	0.95	0.93	2500	0.86	0.62
417	2000	1.25	0.26	0.95	0.82	2500	0.86	0.62
418	2500	0.50	0.26	0.95	0.88	2000	0.88	0.59
419	2000	0.75	0.26	0.95	0.73	2000	0.86	0.50
420	2500	1.26	0.01	0.95	0.93	2000	0.88	0.75
421	2500	0.75	0.31	0.95	0.89	2000	0.80	0.62
422	2500	0.63	0.36	0.95	0.94	2000	0.88	0.88
423	2000	1.25	0.21	0.95	0.88	2000	0.88	0.62
424	2000	1.25	0.11	0.95	0.88	2500	0.80	0.62
425	2000	1.25	0.31	0.95	0.82	2000	0.88	0.71

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
426	2500	1.26	0.41	0.95	0.82	2000	0.62	0.75
427	2500	0.75	0.41	0.95	0.89	2500	0.77	0.82
428	2500	0.75	0.21	0.95	1.00	2000	0.88	0.75
429	2500	0.75	0.16	0.95	0.88	2000	0.88	0.71
430	2000	1.25	0.16	0.95	1.00	2000	0.80	0.75
431	2500	0.75	0.01	0.95	0.88	2000	0.88	0.62
432	2500	0.50	0.36	0.95	0.94	2000	0.93	0.75
433	2000	1.25	0.11	0.95	0.93	2000	0.80	0.62
434	2000	0.62	0.36	0.95	0.93	2500	0.86	0.88
435	2000	0.37	0.36	0.95	0.89	2000	0.86	0.62
436	2500	0.50	0.21	0.95	0.93	2000	0.88	0.88
437	2000	1.25	0.31	0.95	0.93	2000	0.88	0.75
438	2000	1.25	0.16	0.95	0.93	2000	0.82	0.75
439	2500	0.63	0.36	0.95	0.94	2000	0.88	0.71
440	2000	0.62	0.21	0.95	0.93	2500	0.86	0.88
441	2000	0.75	0.31	0.95	0.94	2000	0.86	0.67
442	2500	1.26	0.11	0.95	0.86	2000	0.93	0.75
443	2500	0.63	0.26	0.95	0.94	2500	0.77	0.71
444	2000	0.62	0.31	0.95	0.94	2000	0.88	0.75
445	2500	0.50	0.21	0.95	1.00	2000	0.71	0.75
446	2500	0.50	0.31	0.95	0.94	2000	0.88	0.78
447	2500	0.50	0.31	0.95	0.82	2500	0.77	0.75
448	2500	0.63	0.50	0.95	0.88	2000	0.75	0.71
449	2500	0.63	0.41	0.95	0.89	2000	0.86	0.75
450	2500	0.63	0.41	0.95	0.94	2000	0.86	0.75
451	2000	2.49	0.01	0.95	0.86	2000	0.88	0.62
452	2500	0.75	0.21	0.95	1.00	2500	0.86	0.75
453	2000	2.50	0.21	0.95	0.88	2000	0.75	0.62
454	2000	0.62	0.26	0.95	0.89	2000	0.75	0.75
455	2500	1.26	0.36	0.95	0.88	2500	0.93	0.82
456	2500	0.75	0.26	0.95	1.00	2000	0.88	0.88
457	2500	0.63	0.36	0.95	0.89	2000	0.71	0.71
458	2500	0.63	0.41	0.95	0.94	2000	0.88	0.75
459	2000	1.25	0.21	0.95	0.88	2500	0.77	0.62
460	2500	0.25	0.21	0.95	0.94	2000	0.75	0.88
461	2000	1.25	0.31	0.95	0.94	2000	0.93	0.62

Continuation of Table B.5

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
462	2000	1.25	0.11	0.95	0.94	2000	0.75	0.75
463	2000	0.62	0.36	0.95	0.89	2000	0.88	0.62
464	2000	0.37	0.36	0.95	0.88	2500	0.86	0.75
465	2000	1.87	0.36	0.95	0.88	2000	0.88	0.62
466	2500	0.63	0.26	0.95	0.94	2000	0.88	0.75
467	2500	1.26	0.21	0.95	0.86	2000	0.80	0.62
468	2000	0.75	0.21	0.95	0.94	2000	0.88	0.88
469	2000	1.25	0.26	0.95	0.88	2500	0.77	0.88
470	2000	2.49	0.01	0.95	0.86	2000	0.67	0.75
471	2500	0.75	0.31	0.95	0.94	2000	0.88	0.62
472	2500	1.25	0.11	0.95	0.93	2000	0.71	0.75
473	2500	0.63	0.16	0.95	1.00	2000	0.80	0.62
474	2000	1.87	0.01	0.95	0.93	2000	0.93	0.75
475	2000	1.87	0.16	0.95	0.86	2000	0.75	0.62

End of Table

Table B.6: Parameter configuration and experimental results of each signal in Dataset 5 (see Section 5.5). W_{size}: window size; K%: kernel size in percentage of the window size; O%: overlap percentage of the window size. T%: amplitude threshold for event detection; Novelty: our proposed method with the novelty function; WL: window-based segmentation; BS: binary segmentation. The tolerance used to calculate the F1-score ($F1$) equals the window size for novelty function computation.

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
0	1500	1.90	0.10	0.95	0.97	2500	0.88	0.80
1	2500	1.26	0.10	0.95	1.00	2500	0.88	0.86
2	2500	1.89	0.14	0.95	0.69	2500	0.71	0.57
3	1000	3.12	0.10	0.95	0.97	2500	0.76	0.69
4	1000	3.12	0.10	0.95	0.97	2500	0.88	0.80
5	1500	1.90	0.10	0.95	0.97	2500	0.82	0.86
6	1000	2.50	0.10	0.95	1.00	2500	0.88	0.80
7	2500	1.89	0.10	0.95	0.91	2500	0.75	0.67
8	2500	1.89	0.10	0.95	0.76	2500	0.71	0.57

Continuation of Table **??**

	Novelty					WL		BS
Signal	W_{size}	K%	O%	T%	$F1$	W_{size}	$F1$	$F1$
9	2500	1.26	0.28	0.95	0.88	2500	0.88	0.73
10	2500	3.15	0.10	0.95	0.75	1500	0.71	0.57
11	1000	2.50	0.10	0.95	0.97	2500	0.88	0.86
12	1000	2.50	0.10	0.95	1.00	2500	0.85	0.80
13	1000	1.87	0.10	0.95	0.94	1500	0.82	0.80
14	2500	3.15	0.14	0.95	0.94	1500	0.82	0.91
15	1500	1.27	0.10	0.95	0.94	2500	0.88	0.86
16	2500	1.26	0.32	0.95	0.64	2500	0.53	0.45
17	2500	1.89	0.10	0.95	0.91	2500	0.94	0.86
18	1500	2.53	0.10	0.95	1.00	2500	1.00	0.91
19	2500	1.89	0.10	0.95	1.00	2500	0.82	0.86
20	2500	1.26	0.37	0.95	0.60	2500	0.65	0.57
21	2500	1.26	0.28	0.95	0.86	2500	0.82	0.74
22	2500	1.26	0.14	0.95	1.00	1500	0.88	0.74
23	2500	1.89	0.10	0.95	0.74	2500	0.53	0.46
24	2500	1.89	0.10	0.95	0.88	2500	0.82	0.69
25	2500	1.26	0.10	0.95	0.64	2500	0.65	0.46
26	2500	2.52	0.19	0.95	0.90	2500	0.88	0.91
27	2500	1.89	0.10	0.95	0.58	2500	0.47	0.51
28	2500	3.15	0.23	0.95	0.65	2500	0.71	0.57
29	2500	1.89	0.14	0.95	0.74	2500	0.78	0.68
30	1500	2.53	0.14	0.95	0.97	2500	0.88	0.86
31	1000	1.87	0.14	0.95	0.97	2500	0.94	0.86
32	1000	3.12	0.10	0.95	1.00	2500	0.82	0.86
33	2500	1.26	0.10	0.95	0.78	2500	0.88	0.80
34	1500	2.53	0.10	0.95	1.00	2500	1.00	0.86
35	2500	1.26	0.10	0.95	0.62	2500	0.65	0.46
36	2500	0.63	0.14	0.95	0.97	2500	0.91	0.74
37	2500	1.89	0.23	0.95	0.76	1500	0.76	0.74
38	2500	1.89	0.10	0.95	0.94	2500	0.88	0.74
39	2500	1.89	0.10	0.95	0.89	2500	0.88	0.69
40	2500	1.89	0.10	0.95	0.94	2500	0.94	0.91
41	1000	1.87	0.14	0.95	0.88	1500	0.41	0.57
42	2500	3.15	0.10	0.95	0.67	1500	0.67	0.58
43	2500	0.63	0.19	0.95	0.67	2500	0.56	0.67
44	2500	1.89	0.14	0.95	0.52	2500	0.35	0.46

Continuation of Table **??**

	Novelty					WL		BS
Signal	W_{size}	$K\%$	$O\%$	$T\%$	$F1$	W_{size}	$F1$	$F1$
45	2500	1.26	0.10	0.95	0.50	1000	0.53	0.63
46	2500	2.52	0.14	0.95	0.91	2500	0.65	0.80
47	2500	1.89	0.10	0.95	0.94	2500	0.76	0.69
48	1000	6.25	0.10	0.95	1.00	1000	0.65	0.74
49	1000	3.12	0.10	0.95	0.94	1500	0.76	0.74
50	2500	3.15	0.10	0.95	0.71	1500	0.65	0.63

End of Table

Table B.7: Parameter configuration and novelty function-based experimental results of each signal in Dataset 10 (see Section **??**). W_{size}: window size; K%: kernel size in percentage of the window size; T%: amplitude threshold for event detection; The tolerance corresponded to five samples, specified by the benchmark specifications [25].

Signal	$F1$	W_{size}	K%	T%
apple	0.95	10	50	0.65
bank	0.67	100	100	0.90
bee_waggle_6	0.66	100	250	0.80
bitcoin	0.69	10	65	0.15
brent_spot	0.86	10	30	0.75
businv	0.93	20	15	0.70
centralia	0.98	6	2	0.70
children_per_woman	0.88	10	10	0.70
co2_canada	0.85	10	20	0.35
construction	0.93	20	50	0.70
debt_ireland	0.97	6	2	0.70
gdp_argentina	0.97	20	10	0.70
gdp_croatia	1.00	20	10	0.70
gdp_iran	0.92	10	10	0.70
gdp_japan	1.00	10	10	0.70
global_co2	0.62	100	50	0.70
homeruns	0.93	15	25	0.50
iceland_tourism	0.65	150	150	0.99
jfk_passengers	0.98	20	30	0.80
lga_passengers	0.89	20	30	0.80
measles	0.17	20	30	0.80
nile	1.00	20	30	0.80
occupancy	0.95	10	20	0.40
ozone	0.86	6	4	0.60
quality_control_1	1.00	6	20	0.60
quality_control_2	1.00	6	20	0.60
quality_control_3	1.00	20	30	0.80
quality_control_4	0.97	10	50	0.75
rail_lines	0.91	6	2	0.75
ratner_stock	0.93	6	50	0.75
robocalls	0.98	6	10	0.50
scanline_126007	0.89	6	10	0.30
scanline_42049	0.98	6	20	0.50
seatbelts	0.66	6	20	0.30
shanghai_license	0.98	20	10	0.80
unemployment_nl	0.82	4	6	0.27
usd_isk	0.91	6	25	0.30
us_population	0.93	4	6	0.27
well_log	0.81	10	15	0.40

B.2 Novelty Segmentation of Occupational Data

Table B.8: Results of type event 1 (work period transition), discriminated per time serie samples. Measurements of P, R, F1 of each according sample from Dataset 14 (see Section **??**). Signals without changes in activity were not considered (N.A).

Signal	P	R	F1
Operator 1 Workstation 1	0,78	1	0,88
Operator 1 Workstation 2	1	1	1
Operator 2 Workstation 1&2	0,86	1	0,92
Operator 3 Workstation 1	0,80	1	0,89
Operator 3 Workstation 2	N.A	N.A	N.A
Operator 4 Workstation 1	1	1	1
Operator 4 Workstation 2	N.A	N.A	N.A
Operator 5 Workstation 1	1	1	1
Operator 5 Workstation 2	1	1	1
Operator 6 Workstation 1	N.A	N.A	N.A
Operator 6 Workstation 2	N.A	N.A	N.A
Average	0.92	1	0.96

Table B.9: Detected cycles and duration error results of the detection of cycling events, on Dataset 14 (see Section **??**). When *1&2* appears on the Table, it means the collaborator performed both workstations on the same signal.

Signal	Detected Cycles
Operator 1 Workstation 1	11/11
Operator 1 Workstation 2	14/15
Operator 2 Workstation 1&2	14/14
Operator 2 Workstation 2	11/11
Operator 3 Workstation 1&2	16/16
Operator 3 Workstation 2	13/13
Operator 4 Workstation 1	14/14
Operator 4 Workstation 2	11/11
Operator 5 Workstation 1	12/12
Operator 5 Workstation 2	10/11
Operator 6 Workstation 1	14/15
Operator 6 Workstation 2	14/15
Total	154/157