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) [44].

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 xi

List of Tables xvi

Acronyms xvii

Symbols xviii

1 Introduction 1
- 1.1 Biosignals and Challenges of a Data-Driven Society . . . 1
- 1.2 Linguistic Nature of Time Series . . . 2
- 1.3 Biosignals: Context and Relevance in Occupational Health . . . 3
- 1.4 Research Paths . . . 5
- 1.5 Thesis Structure . . . 5

2 Time Series Fundamentals 8
- 2.1 Global Definitions . . . 8
- 2.2 Filtering . . . 9
  - 2.2.1 Spectral Filtering . . . 9
  - 2.2.2 Smoothing . . . 10
  - 2.2.3 Wandering Baseline . . . 10
- 2.3 Normalization . . . 10
- 2.4 Transformation . . . 11
  - 2.4.1 Spectral Transformation . . . 11
  - 2.4.2 Feature-based Representation . . . 11
  - 2.4.3 Piecewise Aggregate Approximation . . . 12
  - 2.4.4 Symbolic Aggregate Approximation . . . 13
- 2.5 Distance Measures . . . 13
  - 2.5.1 Euclidean Distance . . . 14
  - 2.5.2 Dynamic Time Warping . . . 15

2.5.3 Complexity Invariant Distance . . . 15
2.5.4 Feature-based Distance . . . 16
2.6 Applying Distance Measures . . . 17
2.6.1 Distance Profile . . . 17
2.6.2 Self-Distance Matrices . . . 18
2.6.3 Template-based Search . . . 19
2.6.4 The k-Nearest Neighbors . . . 19
2.7 Text Mining on Time Series . . . 20
2.7.1 Time Series Textual Abstraction . . . 20
2.7.2 Text Features . . . 20
2.7.3 Text Pattern Search . . . 22
2.8 Performance and Validation Measures . . . 23
2.8.1 Classification Problems . . . 23
2.8.2 Event Detection . . . 24
2.8.3 Evaluating Expressiveness . . . 25
2.8.4 Comparing Algorithms . . . 26

3 Unveiling the *Grammar* of Time Series **28**
3.1 The Problem . . . 28
3.1.1 Search Dimension . . . 28
3.1.2 Type Dimension . . . 29
3.1.3 Proposal . . . 29
3.2 Building the SSM . . . 30
3.2.1 Feature Extraction . . . 30
3.2.2 Feature-based SSM . . . 31
3.3 Information Retrieval . . . 33
3.3.1 Novelty Search . . . 33
3.3.2 Periodic Search . . . 35
3.3.3 Similarity Profiles . . . 35
3.3.4 Query Search . . . 36
3.4 Experimental Evaluation in Selected Use-cases . . . 37
3.4.1 Use-Case 1 - Human Activity . . . 37
3.4.2 Use-Case 2 - Medical domain . . . 40
3.4.3 Use-case 3 - Multidimensional . . . 43
3.5 Time Series Profiling . . . 43
3.5.1 Elements with Relevance . . . 43
3.5.2 Compact Design . . . 44
3.5.3 A step by step example . . . 44

4 Language for Time Series Data Mining **47**
4.1 Syntactic Search on Time Series . . . 48

4.1.1 Pre-Processing - Preparing the Data . . . 49
4.1.2 Connotation - The Symbolic Time Series . . . 49
4.1.3 Expressive Syntactic Search . . . 52
4.2 Towards Interpretable Time Series Classification with SSTS . . . 53
4.2.1 SSTS to Generate Time Series Documents . . . 54
4.2.2 Vectorization of Time Series Documents . . . 56
4.2.3 Towards Interpretable Results . . . 58
4.3 Towards Natural Language for Pattern Search . . . 59
4.3.1 "Googling" Time Series Patterns . . . 59
4.3.2 Mapping Features to Words . . . 61
4.3.3 Linguistic Operators . . . 63
4.3.4 Natural Language Query for Time Series . . . 65

Bibliography **69**

Annexes

I Annex 1 Lorem Ipsum **77**

List of Figures

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. 6

2.1 Discrete Fourier Transform (DFT) of a sum of sine waves. 12

2.2 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. 12

2.3 Piecewise Aggregate Approximation (PAA) representation of a Arterial Blood Pressure (ABP) signal, with window sizes of 2 and 20, repsectively. The PAA representation was computed with the Python for Time Series Classification (PYTS) module [19], based on [39]. 13

2.4 Symbolic Aggregate Approximation (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 [43]. 14

2.5 Dynamic Time Warping (DTW) and Euclidean Distance (ED) distances on two different Electrocardiogram (ECG) signals. 16

2.6 Distance profile of a query based search using the z-normalized ED. The template was a PQRS complex of an ECG. 19

2.7 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. 26

3.1 (a) Categories of search of events. In this case are shown Dimension, Window and Domain. (b) Examples of different type of events that can be considered significant in a time series. 29

3.2 Main process to reach the Self Similairty Matrix (SSM). The information needed to calculate the SSM is the record and the input parameters: the window size (w) and the overlapping percentage (o). The first stage involves the feature extraction process, based on w and o values. Features are extracted on each subsequence ($sT_1, sT_2..., sT_N$), being N the total number of windows. From the first window (sT_1), are extracted features ($f1, f2..., f_K$), being K the number of features used. The feature number is also associated with a shape (circle, triangle, etc...). The features can be extracted on multivariate records, being M the number of records used. Each feature is positioned as a row on the F_M and the SSM is computed from it. 30

3.3 Description of informative structures of the SSM of a ABP signal. We show a simplified view with highlights on the relevant structures. The record has 4 main structures: A - homogeneous segment, which corresponds to the ABP periodic signal; B - a homogeneous segment of missed data; C - homogeneous segment with a detachment of the sensor. The boxes highlight homogeneous behavior while the paths highlight periodicity in the segment. Segment C has a cross-pattern, which indicates periodicity and symmetry. nf and sf represent the novelty function and similarity function, respectively 32

3.4 Information retrieval topics explained in this section. 33

3.5 (left) Description of the matrix (kernel) used to compute the *novelty function*. The checkerboard pattern is achieved by combining kernel K_H - measure of homogeneity; and K_C - measure of cross-similarity. The resulting kernel (K_N) is combined with a Gaussian function to generate K_G. The Figure is based on the works of *Mueller et al.* [52, 53]; (right) The process to compute the novelty function is described. Kernel K_G is slided along the diagonal of the SSM to compute the *novelty function* presented as the bottom sub-plot. Positions A and B show the effect of block transitions on the *novelty function*. Figure based on the works of [18, 52, 53]. 34

3.6 Profiles computed for each segment of the example signal used in Figure 3.3. 37

3.7 Change point event detection strategy applied on the SSM to search for change point events. The sequence of activities is presented as follows: $Sitting \rightarrow Laying \rightarrow Walking \rightarrow Upstairs \rightarrow Downstairs \rightarrow Sitting \rightarrow Standing \rightarrow Laying \rightarrow Walking \rightarrow Upstairs$. The input variables used are $time_{scale}$=250 samples, $kernel_{size}$=45 samples, overlap=95% 38

3.8 ABP signal change point detection. The parameters used were a size of 5000 samples, with an overlap of 75% and a kernel size of 25 samples. 39

3.9 (a) ABP signal novelty search. The parameters used were a window size of 5000 samples, with an overlap of 95% and a kernel size of 200 samples. . . . 40

3.10 ABP signal. It represents the first 10000 samples of the signal from Figure 3.9. The parameters used were a window size of 250 samples, with an overlap of 95% and a kernel size of 200 samples. 41

3.11 ECG signal with a *pulsus paradoxus* condition starting at the 10000th. The SSM shows the two modes of the ECG. Highlights of each mode are presented with the circle zoomed-in thumbnails. In more detailed are also shown segments A and B, which highlight an area where the SSM indicates possible changes. . 42

3.12 Proposed method applied on "*Occupancy*" record of Dataset **??**. (a) A single time series of the record is used to extract events; while in (b) the SSM is computed with features extracted from the four available time series. 42

3.13 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. 44

3.14 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. 45

3.15 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. 45

3.16 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. 46

4.1 Syntactic Search on Time Series (SSTS) modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search. This Figure also provides an example with the sequence of changes that occur in each step. 48

4.2 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. 50

4.3 Step 3 of SSTS. Specific *subsequences* of the time series can be searched with a regular expressions (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 first derivative. 52

4.4 Steps of transforming time series into documents and corresponding Bag of Words (BoW) and Term Frequency - inverse Document Frequency (TF-idf) matrices for posterior classification. The steps are *sentence generation*, *bag of words model* and *tfidf model*. The human layer indicates when there can be human intervention. The steps are followed with an example. 53

4.5 Steps for the generation of sentences from a raw time series and organization as a document. Here: PP - Pre-processing, C- connotation and S - Search are each SSTS queries used to search for the patterns and attribute the matched *subsequences* the corresponding *word*. 54

4.6 (Top) Using SSTS to detect the rising stage of a time series. Each step of the process is written described as follows: (1) pre-processing: *Sm* is the function *Smooth* with a window size of 25 samples; (2) connotation: *D1*, indicates the first derivate, from which each sample is converted to z - Flat, p - rising and n falling; (3) search - regular expression $p+$ searches for all sequences with 1 or more p characters. (Bottom) Example of sentence generation. Using the other search queries ($p+$, $n+$, $z+$), we can find the derivative patterns and convert it into ordered words. 56

4.7 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 Term Frequency (tf). 57

4.8 Comparing the clustering process of the ED and the proposed symbolic method. In the proposed method, each signal has a word distribution vector, represented on the left. The signals are clustered based on these. 58

4.9 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. 59

4.10 Using our proposed query language, we can create short, intuitive natural language queries to rank the 100 exemplars in Trace, separating our sought class from the remaining data. Here $\star$ is *anything*, ! is *not*, and *square brackets* are a grouping operator (more details in Section 3). 60

4.11 QuoTS (QuoTS) steps. It shows the feature extractino process and matching each feature to words ($W_1, W_2, etc...$). These features are computed in three 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. 61

4.12 Example of a word feature vector. In this case F_{down} and F_{up} represent a negative and positive slope values. 62

4.13 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. 66

4.14 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 represents 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). 67

List of Tables

2.1 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. 22

4.1 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 50

4.2 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 "|". 51

4.3 The connotation variables, search regular expressions and corresponding words assigned to the pattern searched. The parameter m indicates the size, in samples, of the difference between a peak or a plateau, *thr* is the threshold for the derivative functions. Each word has a representation on an example of a signal. 55

4.4 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 the representative feature vectors. When a negation pair is present, it is added to the example. 64

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

In recent years, the continuous increase in accessible wearable technology has contributed to a significant amount of data available. The continuous production of data from wearable devices through the usage of mobile phones, smartwatches, hearables, wristbands, and other non-invasive wearable sensors has provided a valuable quantity of information. This data often comes in the form of time series, being one of the most common data types in nature [33]. As reported in *Tankovska et al.*, the wearable devices usage has more than doubled in the interval between 2016 and 2019, reaching 722 million [68], leading to a large volume of time series data being gathered in all possible scenarios, by monitoring patients in healthcare institutions [76, 67, 47, 11, 51, 8], tracking everyday activities of humans [14, 15, 4], recording machines in industrial processes or workers motion while performing their tasks [62, 61].

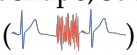
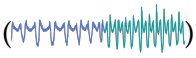
It has never been so easy to gather data about any aspect of our life, work, education, society, or industry. Of course, having relevant information about a subject is beneficial, but the overwhelming amount of data brings tremendous challenges in the ability to save, process, analyze and retrieve interpretable and meaningful information from which we can act upon[69]. Ultimately, it becomes even harder to have data 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.* 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 [59], and this is just an example of how much data can be unused. This is particularly problematic when developing machine learning applications, as data should be correctly pre-processed and labeled to be sure not to include noise, artifacts or mislabeled segments of the signal (*Garbage-in Garbage-out - GIGO*) [59].

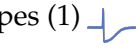
We believe that we should do more with the data we have and for that, tools should be available to support and help analysts to accelerate the process of information retrieval from time series. Both for analysts that are (not) familiar with time series data mining. Having more informative, expressive, and intuitive methods for the analysis of such data,

increases the speed of analysis for experienced analysts and promotes the democratization of this analysis for less or non-experienced analysts [42].

In this thesis, we propose novel methods that contribute to the information retrieval problem on time series. The reader will find two main domains of methods focused in (1) *unveiling the grammar of time series* and (2) *a language for time series data mining*. The proposed methods are designed to help in (1) explaining the *story* that originated the time series through a visual representation that highlights its structure and organization and (2) 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-)experience analyst is searching for specific shapes or patterns in time series. The reader may agree that scientists or other professionals often resort to describe the shape they are looking for. For instance, a physician may say "*I am searching for the T-wave, that represents the* `large peak`" () or "*I am searching for the QRS complex, that looks like a* `sharp peak followed by a sharp valley` ()". This visual intuition also happens when analysts are trying to find differences between classes of signals. For instance, the following shapes (1)  and (2)  are different because "*shape 1 has a* `peak` *where shape 2 doesn't*".

Time series are carriers of information and the presence of a change in the regimes of a time series or the presence of a specific shape in a segment of a time series may be associated with a specific occurrence in the physical world and be attributed to a meaning. This notion of structure and meaning is a good approximation of what represents the foundation of a language: grammar and meaning [16].

Grammar is generally defined as the book of rules that constitutes the structure of a language and is modeled by the morphology and syntax [16]. 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 *unveilling* 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* (Electromyogram (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 [34], accounting for the second-largest responsibility to disability worldwide [45]. These are especially prevalent as upper limb or neck disorder (with 42% of all WMSD cases reported) [28] 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 [70, 71].

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 [5, 58] 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) [54, 49, 63]. 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 [60]. 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 [13, 72]. 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 [61]. 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 [66, 41, 32, 23].

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 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

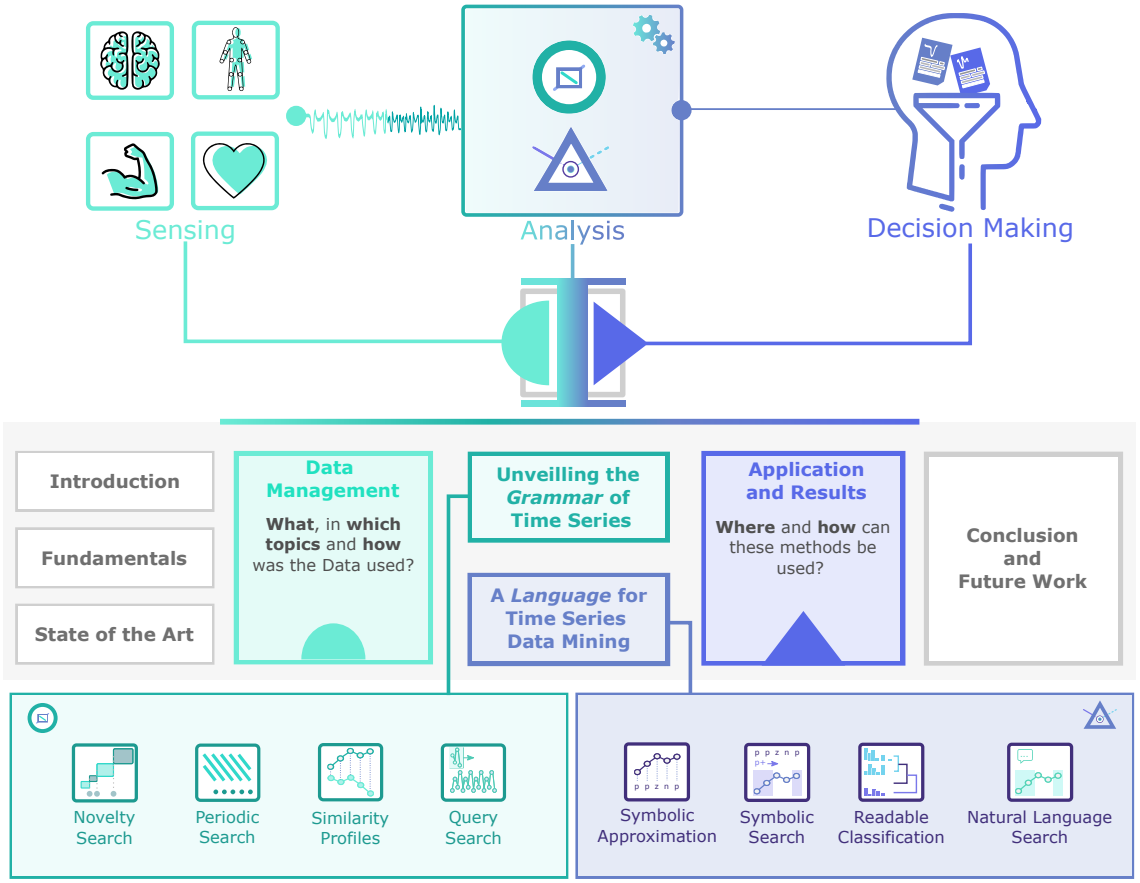


Figure 1.1: General topics contemplated on this thesis and structure of the document. It highlights the 3 layers of involvement related to time series: Sensing, Analysis, and Decision Making, focusing on the Analysis layer, which includes two major topics subdivided into 4 sections each.

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

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*, mostly 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.

2.1 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.

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

Definition 2 - multidimensional time series (MT): A MT is a set of $k \in \mathbb{N}$ time series belonging to the same acquisition: $\{T_1, T_2, ..., T_k\}$. Segments of interest are often searched inside a *time series*. A segment is called a *subsequence*:

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

event.

Definition 4 - Event (E): Following the definitions of [57, 27], which state that "*an event is a dynamic phenomenon whose behavior changes enough over time to be considered a qualitatively significant change*" and "characterized by an interval of measurements that differs significantly from some underlying baseline", we consider that an *event* is an instant in time e that indicates the presence of a relevant occurrence in the time series. Multiple *events* segment the time series into several *subsequences* of different lengths. Therefore, *event* detection is often considered time series segmentation or change point detection[12]. 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.

Definition 5 - Moving Window (MW): A *moving window* is a process of sliding along a time series T to apply a specific method on each *subsequence* it hovers. The window has, such as the *subsequence*, a predefined size $w \in \mathbb{N}$, which starts at a given position i and ends at position $i+w$. The process is iterative and can be made overlapping windows or not. The next window will start at $i+o$, being o the overlapping size and $o \in [1, w]$ (1 for total overlap and w for no overlap).

With a *MW*, 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. Before explaining these strategies, we will give examples of time series covered in this work, focusing on *biosignals*.

2.2 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.

2.2.1 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 is the *Butterworth* filter, with the following frequency response:

$$H_{j\omega} = \frac{1}{\sqrt{1 + \epsilon^2 \left(\frac{\omega}{\omega_c}\right)^2}} \tag{2.1}$$

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

2.2.2 Smoothing

Another method often used to reduce the presence of noise and represents a variation of a low-pass filter is the smoothing technique. Several variations of this technique exist, being the simplest one a moving average, which uses a moving window, 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 center samples.

$$Tm_i = \Sigma_{j=a}^{a+w} T_j H_{j-a} \tag{2.2}$$

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.

2.2.3 Wandering Baseline

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

2.3 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:

$$\overline{T} = \frac{T}{max(|T|)} \tag{2.3}$$

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

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

$$\overline{T} = \frac{T - min(T)}{max(T) - min(T)} \tag{2.4}$$

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

$$\overline{T} = \frac{T - \mu_T}{\sigma_T} \tag{2.5}$$

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.

2.4 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.

2.4.1 Spectral Transformation

One of the first and most well-known techniques suggested for time series transformation was the DFT [1]. The idea behind this concept is that any signal, of any complexity, is a decomposition of a finite number of sine waves. Each wave is represented by a complex number, known as the Fourier coefficient, transforming the signal from the time domain to the frequency domain [65]. This transformation 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 2.1 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 is a frequency series with two main peaks, at the frequencies of the sine waves.

2.4.2 Feature-based Representation

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

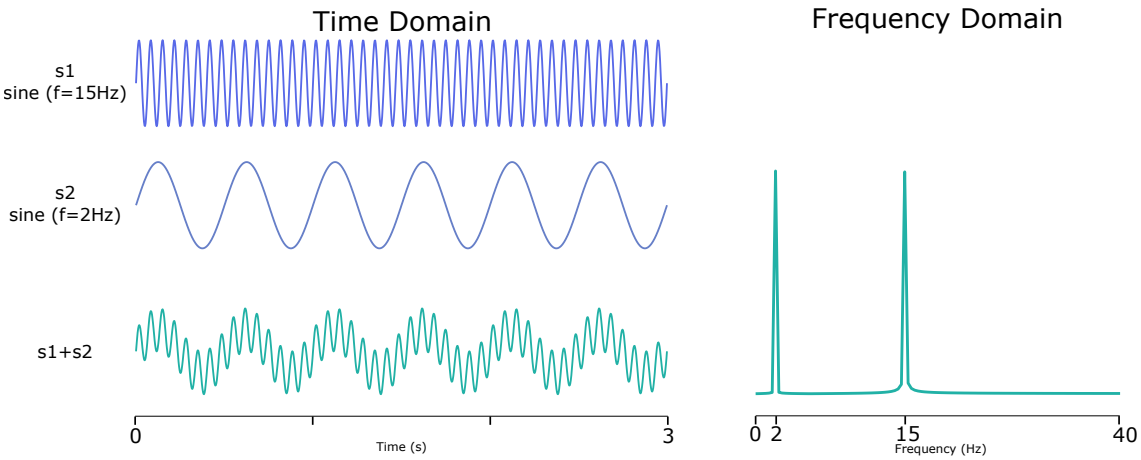


Figure 2.1: DFT of a sum of sine waves.

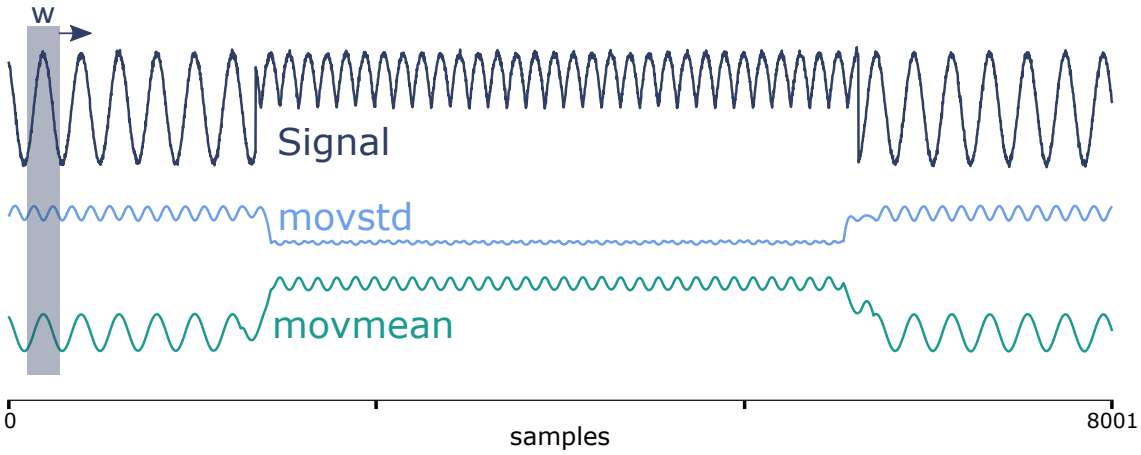


Figure 2.2: Moving window used to extract features with total overlap. The mean and standard deviation are extracted from the signal. The left shows the sine waves, while the right shows the frequency spectrum of the combination of sine waves.

Definition 6 - Feature Series (F): A *feature series*, F, is a feature representation of a time series with size m that depends on the overlap size $o \in \mathbb{N}$ of the sliding process, making the size of the resulting feature series $m = \frac{n}{w-o}$. Considering the existence of a MT, the *feature series* becomes a *multi feature series* of stacked *feature series*, with size $f_{k,m}$.

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

Definition 7 - Feature Matrix (F_M): A *feature matrix*, F_M, is the set of r features extracted for k time series, with size $r \times (k \times M)$.

On Figure 2.2 is showed a time series from which the average (moving mean) and standard deviation (moving *std*) are computed with a moving widow of size $w = 100$.

2.4.3 Piecewise Aggregate Approximation

Another common used transformation method to simplify a time series and reduce its dimension is the PAA) [39]. The new representation space will have size $1 < N \leq n$, in which N is a factor of the original size n. The searches to keep the average of the N

equi-sized subsequences in which the original signal with length n is segmented, which results in $\overline{T} = \overline{t_1}, \overline{t_2}, ..., \overline{t_N}$, such that [39]

$$\overline{t_i} = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} t_j \tag{2.6}$$

An example is showed in Figure 2.3, where a ABP is converted to PAA with sizes 2 and 20, respectively.

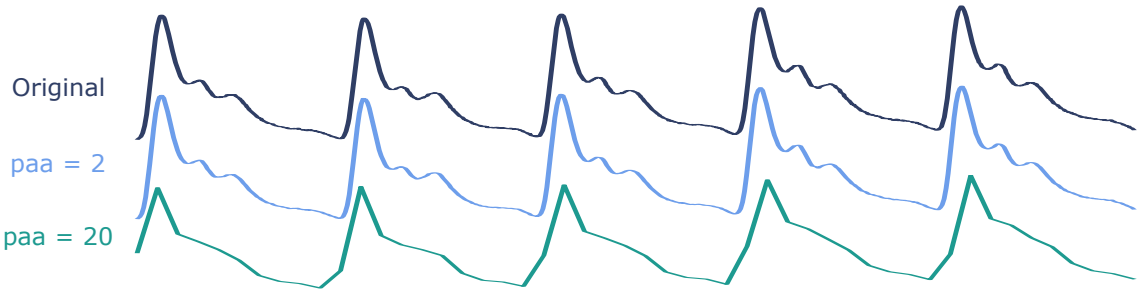


Figure 2.3: PAA representation of a ABP signal, with window sizes of 2 and 20, repsectively. The PAA representation was computed with the PYTS module [19], based on [39].

2.4.4 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 [43]. This method applies PAA to a z-normalized time series and indexes a *character* to each sample of the simplified signal based on the distribution of its amplitude values. The signal's amplitude values are separated in bins with equal probability. The number of bins is equal to the size of the *alphabet* chosen. Figure 2.4 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.

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

2.5 Distance Measures

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

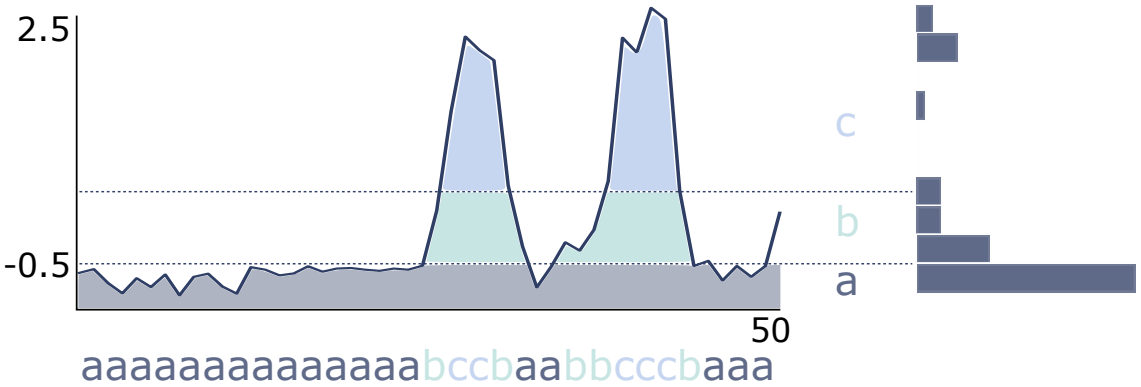


Figure 2.4: SAX representation of a power consumption signal from a Dutch Company, with window bin size of 3. The SAX representation was computed with PYTS based on [43].

2.5.1 Euclidean Distance

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

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

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

$$ED(Q, C) = \sqrt{\Sigma_{i=1}^{n}(q_i - c_i)^2} \tag{2.7}$$

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 distortion, and local scaling ("warping") distortion. The first can be compensated by the z-normalized ED [7]:

$$z\_ED(Q, C) = \sqrt{2m(1 - \frac{\Sigma_{i=1}^{m} Q_i C_i - m\mu_Q\mu_C}{m\sigma_Q\sigma_C})} \tag{2.8}$$

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.

2.5.2 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 $d = (q_i - c_j)^2$ [38]. Figure 2.5 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 [38].

$$W = w_1, w_2, ..., w_k, ..., w_K; \qquad max(m, n) \leq K < m + n + 1 \tag{2.9}$$

$$DTW(Q, C) = min\sqrt{\Sigma_{k=1}^{K} w_k} \tag{2.10}$$

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:

$$\gamma(i, j) = d(q_i, c_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \tag{2.11}$$

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

2.5.3 Complexity Invariant Distance

A different type of distance measure is also used to cope with complexity invariance. This distance uses a complexity correction factor (CF) with an existing distance measure, such as ED [7]:

$$CD(Q, C) = ED(Q, C) \times CF(Q, C) \tag{2.12}$$

The CF is defined as [7]:

$$CF = \frac{max\{CE(Q), CE(C)\}}{min\{CE(Q), CE(C)\}} \tag{2.13}$$

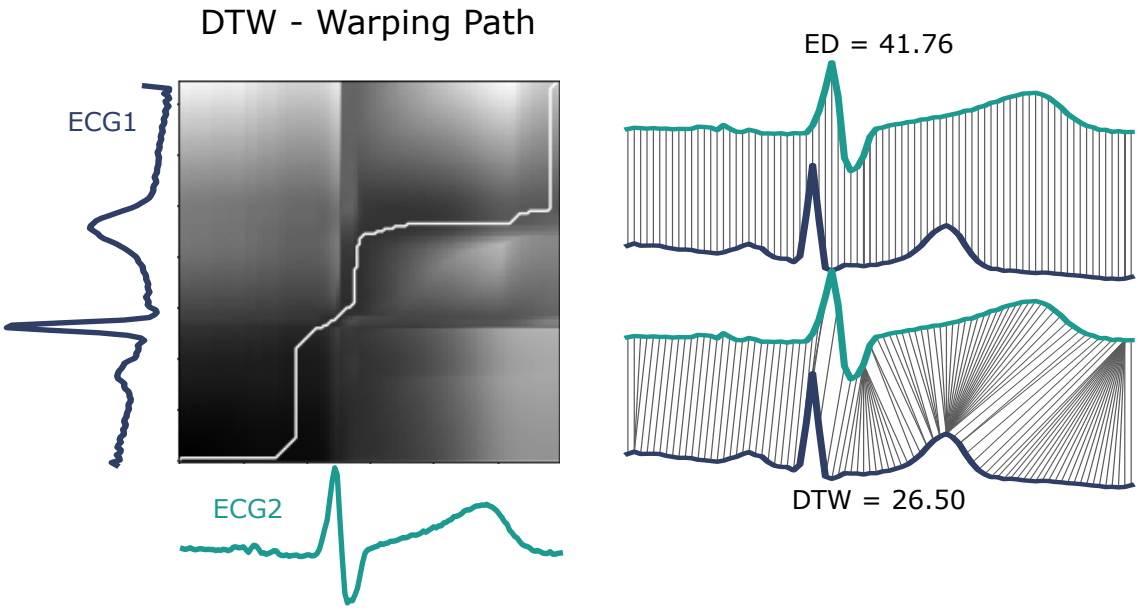


Figure 2.5: DTW and ED distances on two different ECG signals.

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[7]:

$$CE(Q) = \sqrt{\Sigma_{i=1}^{n-1}(q_i - q_{i+1})^2} \tag{2.14}$$

These distance measures are performed on the original representation domain of time series. As we showed above, other representation techniques can be employed, creating opportunities for other types of approaches. In this work, we 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.

2.5.4 Feature-based Distance

As mentioned, a feature series F can be computed from the original time series to represent it based on a specific feature. If the size of the *moving window* is equal to the size of the time series then F is represented by a single value. Otherwise, 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 [30] (equation 2.15).

$$CS = \frac{\vec{f_A} \cdot \vec{f_B}}{||\vec{f_A}||||\vec{f_B}||} \tag{2.15}$$

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 [30].

2.6 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 and find if it occurs in the time series. Another relevant application of distance measures is its usefulness to retrieve relevant structural information of a time series by comparing each of its *subsequences* to all other *subsequences*. In this subsection, relevant methods applied with the help of the presented distance measures are explained to retrieve information from a time series. We will start with distance/similarity profiles.

2.6.1 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 *profile* for a time series based on a z-normalized ED matrix. By keeping the lowest value of each *distance profile* (nearest neighbor), we retrieve the *matrix profile* [25]. The result gives the minimal distance pair of each *subsequence*, meaning that minimum values are *motifs* and maximum values are *discords*.

Definition 8 - Nearest Neighbor (NNbr): The NNbr is the *subsequence* with lowest distance from the *subsequence* being compared with all the other *subsequences*. The NNbr form a pair.

Definition 9 - Motif (mtf): The NNbr 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*.

Definition 10 - Discord (drd): The NNbr pair that has the highest distance forms a *discord*. That means that on the entire time series, these two *subsequences* are the furthest apart.

2.6.2 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 [52]. Each are relevant assets for segmentation and summarization tasks.

Let X be a sequence with size N that can be a time series or a representation of a time series in the PAA or feature space, such that $X = (x_1, x_2, ..., x_i, ..., x_j, ..., x_N)$. Each element of X can either be a single value or a vector with r features. Independently of that, a matrix SDM with size $N \times N$ can be computed, such that:

$$SDM(i, j) = d(x_i, x_j) \tag{2.16}$$

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* [52, 53].

Areas with lower distance are highlighted as *homogeneous* structures. These give an indication of *homogeneity* and *novelty*. *Homogeneity* because a *block* along the diagonal means that the time series has a constant behavior during the segment delimited by the *block*. *Novelty* because when 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.

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

It is relevant to highlight that as distance matrices can be computed, similarity matrices can as well, following the same equation 2.16, but using a similarity measure (s) instead of d. Further, we will mention the similarity matrix, which will be called SSM.

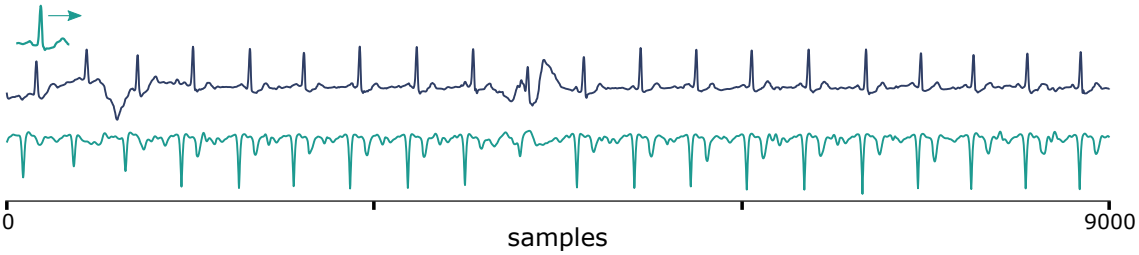


Figure 2.6: Distance profile of a query based search using the z-normalized ED. The template was a PQRS complex of an ECG.

2.6.3 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 2.6, where a PQRS complex of an ECG is used as a template to search for all the other complexes. The method applied a simple 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* [48] or text [2]. In this work, we will introduce novel ways of performing a query-based search with regex and natural language.

2.6.4 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-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 [17].

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 is the k 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 [17].

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. The reader will appreciate that an in-depth explanation of the relationship between text and time series is made.

2.7 Text Mining on Time Series

2.7.1 Time Series Textual Abstraction

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

Definition 11 - Character (Char): A *character* is an unit symbolic element that represents a sample or *subsequence* of a time series. Each sample of a time series is transformed into a *character* to form a *symbolic time series*.

Definition 12 - Symbolic Time Series (ST): A *symbolic time series* is a sequence of *characters* ordered in time with length $n \in \mathbb{N}$: $ST = (st_1, st_2, ..., st_n)$. A specific sequence of *characters* of a ST can form a *word*.

Definition 13 - Word (W): A *word* is the concatenation of a sequence of *characters*, giving a textual representation of a *subsequence*. Putting *words* together forms a *sentence*.

Definition 14 - Sentence (S): A *sentence* represents a group of *subsequences*. It is formed by joining sequences of symbolic *words*.

Definition 15 - Document (D): The set of *sentences* in a time series are called a *document*. It represents the entire time series.

Definition 16 - Corpus (GD): The *corpus* is a collection of text material (group of documents). It represents a higher level of textual information. This collection is typically annotated and used for machine learning tasks. In this case, a corpus will be represented by the set of documents that describe a time series dataset.

Definition 17 - Vocabulary (V): The *vocabulary* comprehends the set of all different words present in all time series.

2.7.2 Text Features

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

Definition 18 - Bag of Words (Bow): A BoW is a feature matrix representation of a

corpus, being the feature the number of occurrences of each *term*, called the tf:

$$bow(t,d) = tf_{t,d} = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{2.17}$$

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 *word* or an *n-gram*.

Definition 19 - N-gram: It is a span of followed *words* that are counted in the BoW/TF-idf. Possible *2-gram* from Figure 2.4 would be `aa`, `ca` or `ac`. This strategy resembles a *moving window* with total overlap on time series, but for text. It makes the BoW/TF-idf model more robust since it makes it rely in more than single *word* statistics. Regarding time series, this method is relevant because it takes into account time dependencies between *words*, which reflects the time dependency seen in time series between *subsequences* (e.g. it is more meaningful to say that a signal has a `peak` next to a `valley` that 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 limitation, the TF-idf matrix is used.

Definition 20 - Term Frequency Inverse Document Frequency (TF-idf): The TF-idf matrix increases the relevance of t by means of the t_f, while reducing its importance in proportion to the number of *documents*, d that contain the term t. The model is defined by being a ratio between the t_f and the *inverse document frequency* (idf), which is calculated as follows:

$$idf(t,D) = \log \frac{L}{|\{d \in D : t \in d|\}} \tag{2.18}$$

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

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \tag{2.19}$$

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 Support Vector Machines (SVM) classifiers [56].

2.7.3 Text Pattern Search

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

The last method is a parsing technique that is convenient to write text patterns, being more flexible than direct matches. It is based on regular languages, 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 [22]:

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 2.1: 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.

2.8 Performance and Validation Measures

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

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

$$P = \frac{TP}{TP + FP} \tag{2.20}$$

$$R = \frac{TP}{TP + FN} \tag{2.21}$$

$$F1 = 2 \times \frac{P * F}{P + R} \tag{2.22}$$

$$F1 = \frac{TN + TP}{TN + TP + FP + FN} \tag{2.23}$$

These measures were initially performed in binary classification problems, but can be adapted for multi-classification ones. For this, each class (the target class) is compared

to all the other classes, being the target class the *positive* and all the other classes *negative*. All measures are calculated for each class. Finally, a macro and micro average of these metrics can be calculated.

$$macroP = \Sigma_{i=0}^{c} \frac{P_c}{c} \tag{2.24}$$

$$macroR = \Sigma_{i=0}^{c} \frac{R_c}{c} \tag{2.25}$$

$$microP = \frac{\Sigma_{i=0}^{c} TP_c}{\Sigma_{i=0}^{c} TP_c + \Sigma_{i=0}^{c} FP_c} \tag{2.26}$$

$$microR = \frac{\Sigma_{i=0}^{c} TP_c}{\Sigma_{i=0}^{c} TP_c + \Sigma_{i=0}^{c} FN_c} \tag{2.27}$$

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

2.8.2 Event Detection

Regarding event detection problems, the process involves finding the sample that corresponds to the ground truth event. Considering that it would not be fair to calculate the performance of an event detection algorithm by searching if the ground truth sample is found, we calculate the TP, FP and FN based on an error margin. From these measures, metrics from Equations 2.20, 2.21 and 2.22 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):

$$MAE = \sum_{i=1}^{k} \frac{|g_i - e_i|}{k} \tag{2.28}$$

$$MSE = \frac{1}{k} \sum_{i=1}^{k} (g_i - e_i)^2 \tag{2.29}$$

$$ME = \frac{1}{k}\sum_{i=1}^{k}(g_i - e_i) \tag{2.30}$$

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).

2.8.3 Evaluating Expressiveness

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 [36]:

Vocabulary

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

$$Voc = oprt + oprd \tag{2.31}$$

Length

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

$$Lgth = Toprt + Toprd \tag{2.32}$$

Calculated Length

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

$$CL = oprt * \log_2(oprt) + oprd * \log_2(oprd) \tag{2.33}$$

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:

$$Vol = Lgth * \log_2(Voc) \tag{2.34}$$

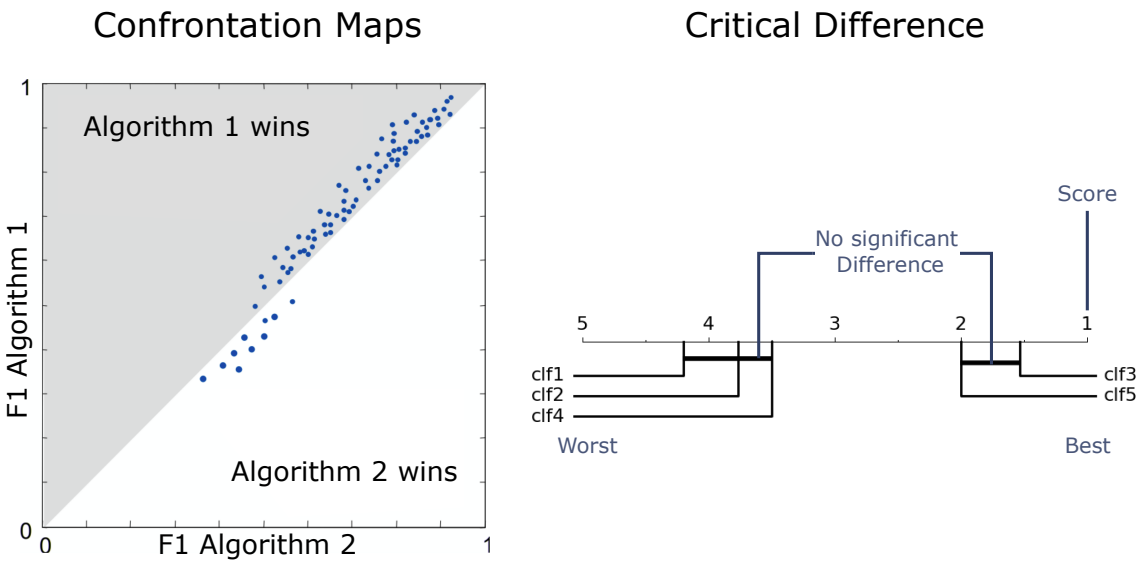


Figure 2.7: On the left are confrontation maps, used to compare classification algorithm performances. On the right is a critical difference map, which shows a statistical difference in the performance of algorithms for general purposes.

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:

$$Dif = (\frac{oprt}{2} + \frac{Toprd}{oprd}) \tag{2.35}$$

Effort

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

$$Efrt = Vol * Dif \tag{2.36}$$

2.8.4 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 2.7.left and 2.7.right, respectively.

2.8.4.1 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 *VS* algorithm 2. Figure 2.7.left displays an example of it taken from [37]. 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.

2.8.4.2 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 2.7.right, the plot compares 5 different classifiers (*clf1* to *clf5*) and highlights that the difference in performance is not significant between *clf1, 2* and *4*, neither between *clf3 and 5*. However, *clf3* and *5* have a much better performance than the other classifiers. The closer the classifiers are from the right (1), the better they are. The bold bar connects the classifier with performances that are not significantly different.

In this work, we will use an implemented critical difference method from [35].

3

Unveiling the *Grammar* of Time Series

In this chapter is described the process to unveil the structure of a time series. The next sections will start by giving an introduction to the variables that have to be retrieved and demonstrates how to perform them. The main method is inspired by the audio-processing domain for *music structure analysis*, namely segmentation and audio-thumbnailing. While having already been extensively studied in this domain [52, 55, 9, 10], this knowledge has not yet been extended to other types of *time series*, which could greatly benefit from it [3].

The process follows the steps of building a similarity matrix using a feature-based representation of a time series. These will be explained and examples will be provided for (1) novelty segmentation, (2) periodic segmentation, (3) similarity profiles, and (4) query-based search. Additionally, we will show how this process can be used for time series summarization and automatic annotation.

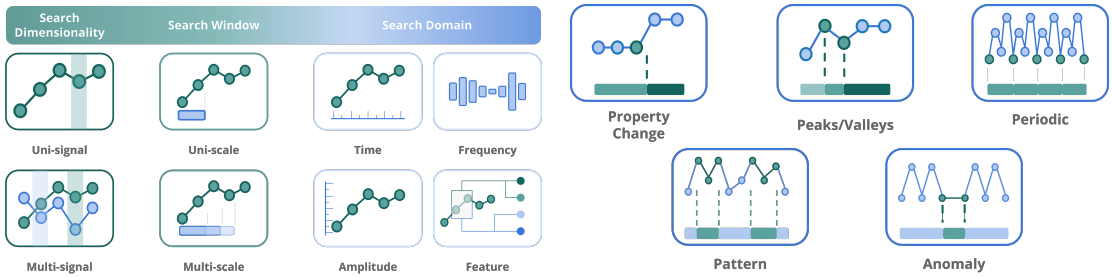
3.1 The Problem

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, especially for tasks related to data annotation/labeling. The structure of a time series is built of *segments* delimited by *events*. The problem is the search for *events* that are significant.

From definition 4 of Chapter 2.1, 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.

3.1.1 Search Dimension

Figures **??** and **??** are an illustrated summary of the two dimensions of the problem. Regarding the *search* dimension, it is formed by three layers: *(1) dimensionality*: the search

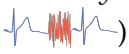


(a) (b)

Figure 3.1: (a) Categories of search of events. In this case are shown Dimension, Window and Domain. (b) Examples of different type of events that can be considered significant in a time series.

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

3.1.2 Type Dimension

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

3.1.3 Proposal

In order to fill as much ground as possible in this problem, we started by defining the search space considering that if the time series would be transformed in the feature space, any change in any of the features would be relevant, for instance, we might be searching for changes in the mean, standard deviation, mean frequency or other property. By characterizing the signal into the feature space, we can explore changes in all feature representations. Additionally, an event should separate two different behaviors. The

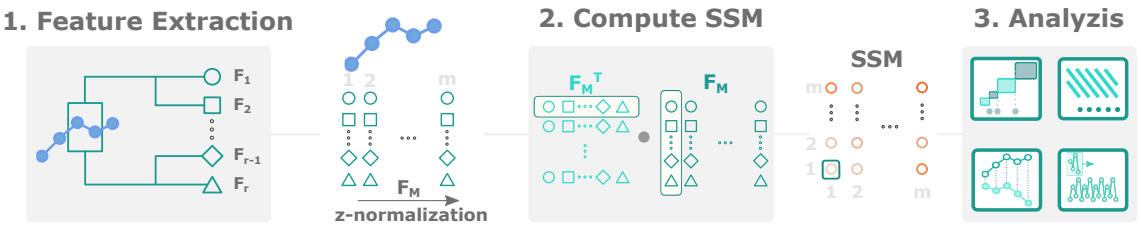


Figure 3.2: Main process to reach the SSM. The information needed to calculate the SSM is the record and the input parameters: the window size (w) and the overlapping percentage (o). The first stage involves the feature extraction process, based on w and o values. Features are extracted on each subsequence ($sT_1, sT_2..., sT_N$), being N the total number of windows. From the first window (sT_1), are extracted features ($f1, f2..., f_K$), being K the number of features used. The feature number is also associated with a shape (circle, triangle, etc...). The features can be extracted on multivariate records, being M the number of records used. Each feature is positioned as a row on the F_M and the SSM is computed from it.

notion of *difference* in time series can be associated with *distance/similarity*. This would enable us to find change points, recurrent patterns, anomalies, and periodic shapes.

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

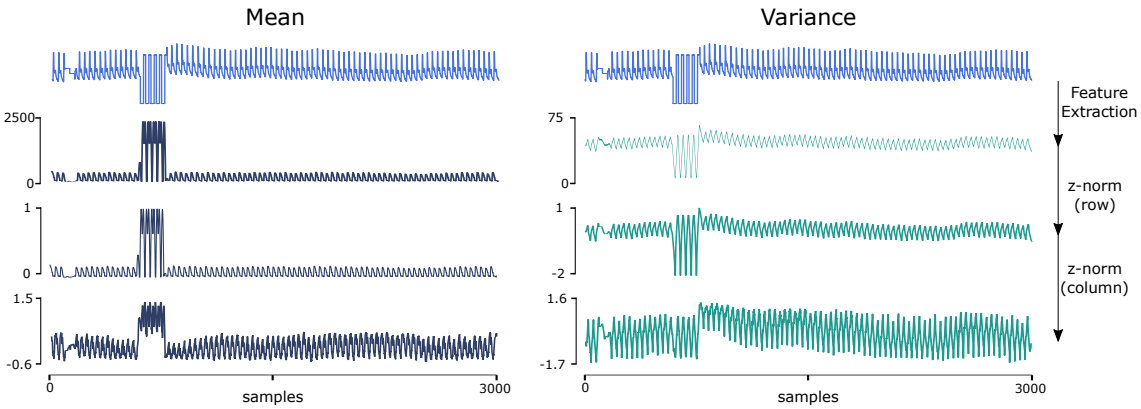
We provide evidence that the proposed method is reliable for the detection of the mentioned events, supporting our claims with several examples in multiple time series domains (it is type agnostic) and comparing the results with state-of-the-art methods. Besides, we highlight that these events are all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

3.2 Building the SSM

In this section, we explain the steps of the proposed method. The extraction of relevant events from time series starts by computing the SSM. As explained in Section 2.6.2, this matrix has relevant structural information to retrieve *events*, namely *blocks*, *paths* and *similarity profiles*. Figure 3.2 summarizes the steps involved in calculating the SSM.

3.2.1 Feature Extraction

The structural information present on the SSM depends on the richness of the set of features into translating the changes and disruptions of the signal. Behavioral changes might be related to a variate set of features. As a feature may be sensitive to a type of change, the set of features should be diverse to identify a multivariate set of events and



be agnostic to all types of signals. For this purpose, we used the available features from the *TSFEL* [6] Python library presented in the Feature Table **??** from Appendix **??**.

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

The extracted features are grouped into a feature matrix (F_M), where the rows represent a feature series and the columns the corresponding *subsequence*, described by all features. Features extracted from a multidimensional record are ordered in the F_M as rows as well. The total number of rows can be, at maximum: $r \times k$, being k the number of time series being analyzed and r the number of features extracted, as illustrated in Figure 3.2.

Each feature extracted is z-normalized to guarantee that each feature series (rows of the feature matrix) has a more equal contribution in the description of the signal. Additionally, a second normalization is applied to the feature vector (columns of the feature matrix), which optimizes the calculation of the cosine distance 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 **??**.

3.2.2 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* within the time series record. Since each column of the F_M is the feature characterization of each *subsequence* by the entire set of features, the comparison between segments is achieved by calculating the dot product between the z-normalized transposed F_M and itself:

$$SSM = F_M^T F_M \tag{3.1}$$

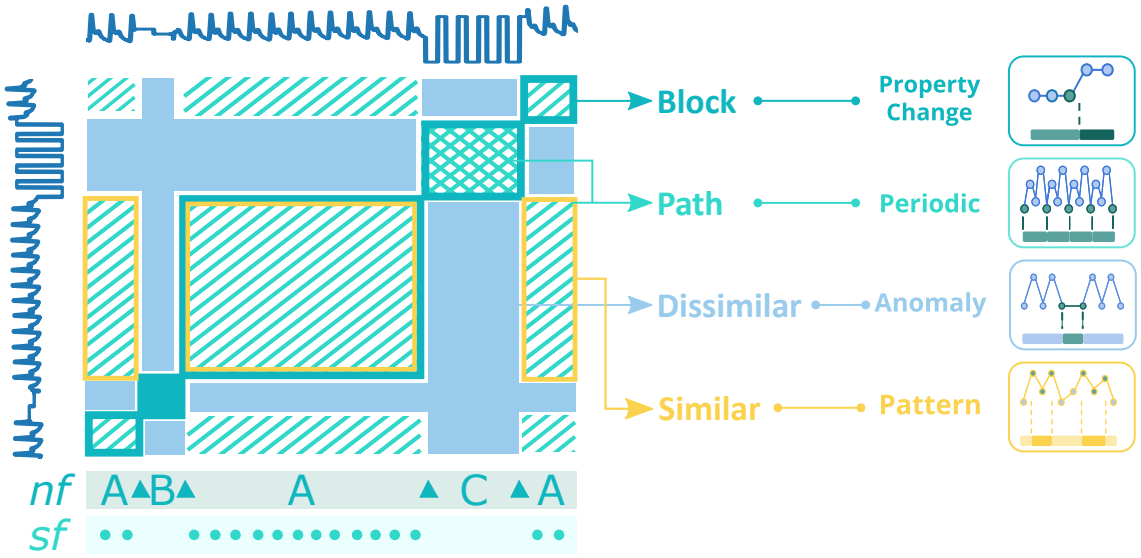


Figure 3.3: Description of informative structures of the SSM of a ABP signal. We show a simplified view with highlights on the relevant structures. The record has 4 main structures: A - homogeneous segment, which corresponds to the ABP periodic signal; B - a homogeneous segment of missed data; C - homogeneous segment with a detachment of the sensor. The boxes highlight homogeneous behavior while the paths highlight periodicity in the segment. Segment C has a cross-pattern, which indicates periodicity and symmetry. nf and sf represent the novelty function and similarity function, respectively

The dot product gives a similarity score based on the feature values of each *subsequence*. Cells of the SSM with higher similarity scores indicate that the corresponding *subsequences* have similar feature values [55, 9]. As a result, the SSM provides rich visual information, highlighting structures, such as blocks and paths, that describe the signal's morphological behavior over time and its structure.

In Figure 3.3, the main structures are illustrated and highlighted in an example of an *SSM* [55] computed from an ABP signal. As mentioned in Chapter 3, the main structures are *blocks* and *paths*. Our proposed method takes advantage of these main structures to extract the desired information.

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

Differently, *blocks* are square-shaped structures that indicate homogeneous areas of the SSM, which translate as constant behavior in the time series. The change between block structures along the main diagonal indicates a relevant change in morphology and behavior in the time series. In Figure 3.3, the SSM is segmented into several blocks on layer *nf*. The triangular shapes indicate 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 highlight (dis)similar segments, such as anomalies or highlight very

Figure 3.4: Information retrieval topics explained in this section.

similar *subsequences*, such as motifs or cycles.

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

3.3 Information Retrieval

The SSM is a powerful visual tool *per se*, highlighting relevant information that could be missed if looking at the raw time series. However, being the information on the SSM, it should be possible to retrieve it automatically. As presented in Figure 3.4, here are explained 4 approaches for information retrieval on the SSM, namely (1) novelty search (*block* transitions), (2) periodic search of patterns (*paths*), (3) similarity profiles (how similar are the *subsequences*) and (4) how to use a query from the SSM to search for specific *subsequences*.

3.3.1 Novelty Search

The search for *novelty* is inspired by a method used in musical structure analysis and presented by *Foote et al.* [20]. The process involves searching for transitions between *blocks* using a moving checkerboard square matrix. The result is a 1 dimensional function designated *novelty function - nova*.

As shown in Figure 3.5, block transitions along the diagonal are represented by a checkerboard pattern. Detecting such patterns can be made by correlating a standard checkerboard matrix with the diagonal of the SSM. For this, a sliding squared matrix, designated *kernel*, is used. As illustrated in Figure **??**, the kernel has a checkerboard pattern and is combined with a Gaussian function to add a smoothing factor. The kernel, K_N, is a combination of two different square matrices: K_H and K_C. The first is responsible for identifying the homogeneity of the SSM on each side of the center point along the diagonal. The higher the homogeneity, the higher will be the values in these sections. The latter measures the level of cross-similarity, returning higher values in cases of high cross-similarity. Therefore, when sliding the kernel K_N along the diagonal, a higher correlation value will be returned when it reaches a segment of the SSM with a similar checkerboard pattern. The result is the mentioned $nova$ [18, 52, 53].

As shown in Figure 3.5 (left), the kernel in position **A**, which is placed in an area of high homogeneity, returns a value close to 0 when summing the product between

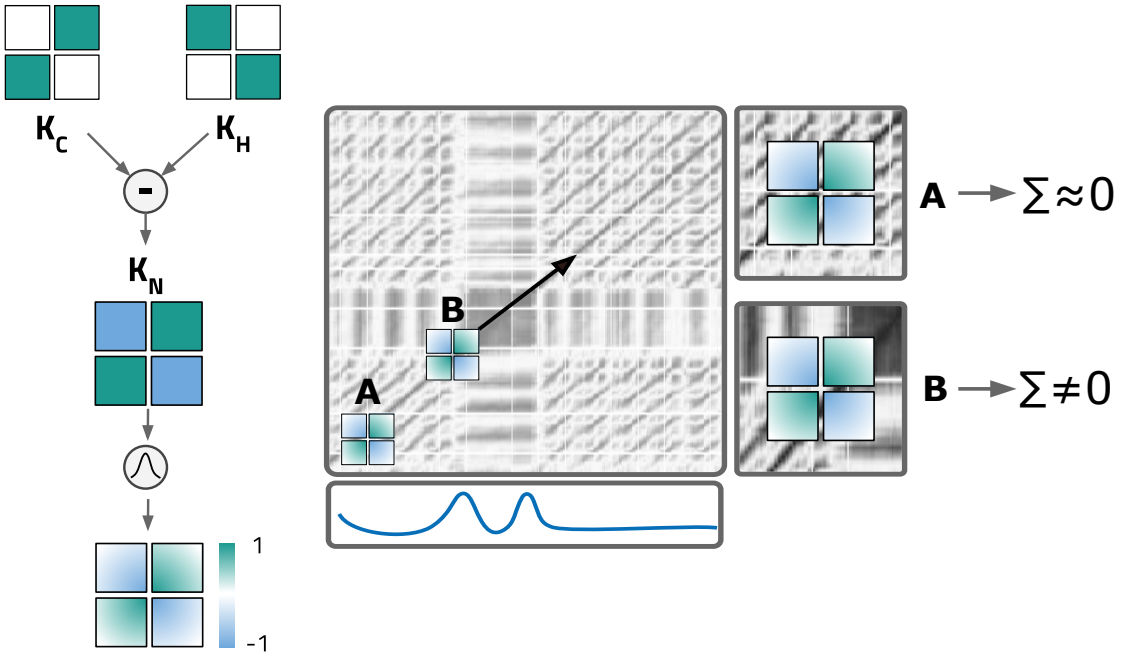


Figure 3.5: (left) Description of the matrix (kernel) used to compute the *novelty function*. The checkerboard pattern is achieved by combining kernel K_H - measure of homogeneity; and K_C - measure of cross-similarity. The resulting kernel (K_N) is combined with a Gaussian function to generate K_G. The Figure is based on the works of *Mueller et al.* [52, 53]; (right) The process to compute the novelty function is described. Kernel K_G is slided along the diagonal of the SSM to compute the *novelty function* presented as the bottom sub-plot. Positions A and B show the effect of block transitions on the *novelty function*. Figure based on the works of [18, 52, 53].

it and the section of the SSM it overlaps. On the other end, in position **B**, the kernel reaches a segment with low cross-similarity and high diagonal similarity, which results in high correlation values with a checkerboard pattern. The *nova* is high in these transition segments [18, 52, 53].

Each section of the kernel has the same size, L, being the total kernel size configured by $D = 2 \times L + 1$, with $L \in \mathbb{N}$. The kernel has an odd size to adapt zero values in centered points. It also has total size D × D, being K_N defined by the following function [52, 53]:

$$K_N(i, j) = \text{sign}(a_i) \cdot \text{sign}(b_j) \tag{3.2}$$

being $a, b \in [-L : L]$ and sign representing the sign function, which indicates the sign of the value (1, 0 or -1). A radially symmetric Gaussian function is used to smooth the Kernel, with the following equation [52, 53]:

$$\phi(p, u) = \exp(-\frac{1}{2L\sigma^2}(p^2 + u^2)) \tag{3.3}$$

being σ the standard deviation, equal for both x and y dimensions of the matrix, L the size of each kernel's section, and p and u the position in the x and y dimensions,

respectively. The final kernel is computed by point-wise multiplication with the Gaussian function:

$$K_G = \phi \cdot K_N \tag{3.4}$$

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

$$nova(m) = \sum_{i,j=0}^{2L+1} K_G(a_i, b_j) SSM(m + a_i, m + b_j) \tag{3.5}$$

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

3.3.2 Periodic Search

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

$$sf(x) = \sum_{i=0}^{m} SSM_{ix} \tag{3.6}$$

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

Although not validated in this work, we highlight that the similarity function can have an additional purpose. Considering that each sample of the sf is an average similarity of a subsequence to all other subsequences, it is possible to find *anomalies*. If we define *anomaly* as a subsequence highly unique and different from all the rest of the time series, then the average similarity to all the other subsequences should be low, meaning that a sample of the sf that represents an anomaly should have a lower value.

3.3.3 Similarity Profiles

The main elements, *blocks* and *paths*, are essential sources of information for the segmentation of the time series. Besides these, the SSM also provides the pairwise similarity values

between all *subsequences* of the time series. This is an important measure that gives an understanding of how close together are each *subsequence* and can be used to cluster them or find either *motifs* or *discords*. In order to use the similarity values of the SSM to compare *subsequences* we can use the *similarity profiles*.

A similarity profile, such as a distance profile explained in Section 2.6.1, represents the similarity values of a *subsequence* to all the other *subsequences*. This simply represents one column/row of the SSM. An example of a *similarity profile* from the ABP signal is presented on Figure **??**. We can find maximum values where the signal is more similar to the *subsequence* from which the profile is being computed. We find that this profile can be used to compare *subsequences* but also entire segments of the signal. Take for instance all three segments A highlighted on Figure 3.2. Although having different size, their profile is highly similar.

Although the comparison between segments could be made by directly using the region of the SSM delimited by both *subsequences*, we find that a stronger measure is to compare how much each of the two segments is similar/different based on their similarity/distance to all the other *subsequences* of the time series. For this, a *similarity profile* ($P_s(c)$) of a segment is computed as the column(row)-wise average similarity values of the region of the SSM delimited by the segment being profiled, with size l, and all the other *subsequences* of the time series, with size m:

$$P_s(c) = \frac{\Sigma_{i=0}^{l} SSM(i, c)}{l} \tag{3.7}$$

The *similarity profile* is computed column(row)-wise, being each column(row) $c(r)$ 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, these can then be compared with the ED and clustered based on these distance values. This process can be specially valuable to cluster the segments previously extracted with the *nova* and *similarity* functions.

A general example of applying this process to the segmented time series based on the *nova* function is showed in Figure **??**. Each segment category (A, B and C) extracted from the SSM of Figure 3.3 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.

3.3.4 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

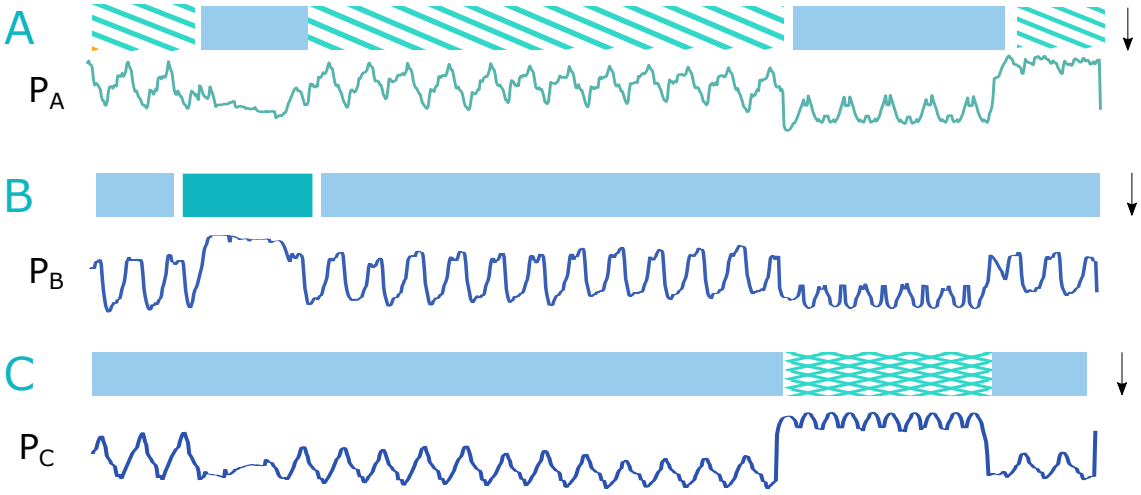


Figure 3.6: Profiles computed for each segment of the example signal used in Figure 3.3.

$$D(i) = \sum_{i=0}^{i=m} \sqrt{(SSM(i) - SSM_t)^2} \tag{3.8}$$

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

3.4 Experimental Evaluation in Selected Use-cases

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

3.4.1 Use-Case 1 - Human Activity

The example presented in Figure **??** shows the usage of the SSM on a record of Dataset 2. In this example, the method was applied to all the 3-axis of the accelerometer data. All are shown and described with the sequence of activities as captioned in Figure 3.7.

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

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

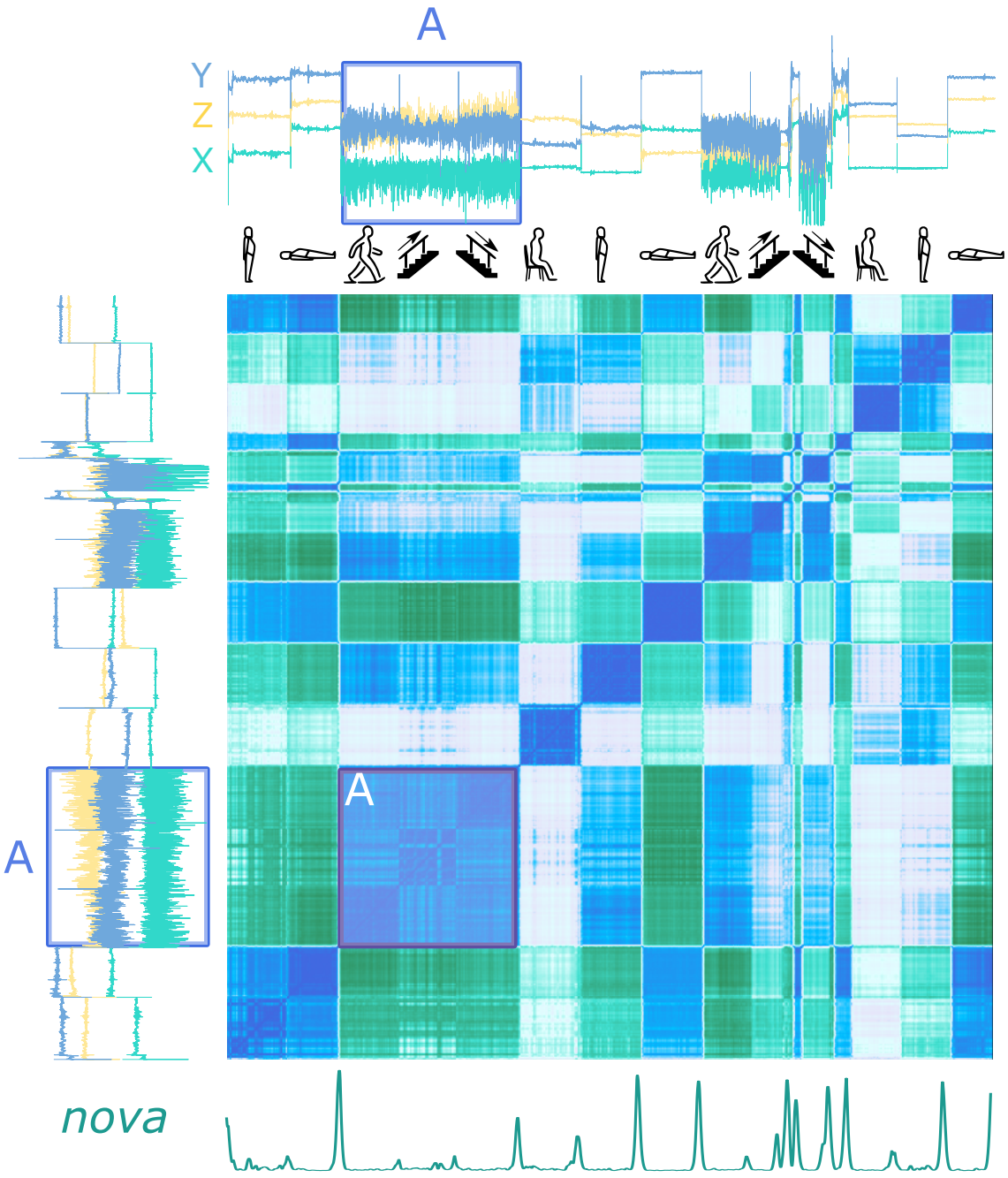


Figure 3.7: Change point event detection strategy applied on the SSM to search for change point events. The sequence of activities is presented as follows: $Sitting \rightarrow Laying \rightarrow Walking \rightarrow Upstairs \rightarrow Downstairs \rightarrow Sitting \rightarrow Standing \rightarrow Laying \rightarrow Walking \rightarrow Upstairs$. The input variables used are $time_{scale}$=250 samples, $kernel_{size}$=45 samples, overlap=95%

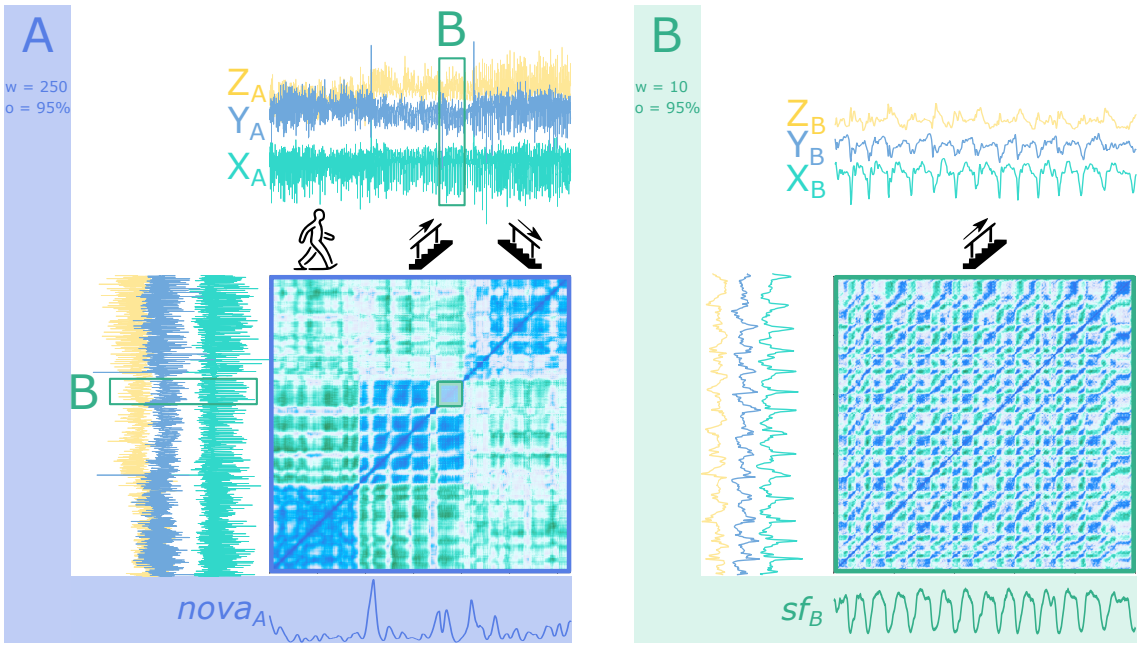


Figure 3.8: ABP signal change point detection. The parameters used were a size of 5000 samples, with an overlap of 75% and a kernel size of 25 samples.

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

When *zooming-in* the SSM into segment A, which shows transitions between walking behaviors, the checkerboard pattern that highlights change points are clearer and the three different walking patterns are easily segmented. The two major peaks in the corresponding *nova* function are from these transitions, as presented in Figure 3.8 (left). In addition, the reader might notice that the segments of the matrix related with *walking in stairs* are also segmented into smaller *blocks*. Although the information is not available in the dataset description, we strongly believe these are a flight of stairs.

Considering that the signal is a walking behavior, the reader might question the fact that the periodicity of the walking pattern is not exhibited on the matrix. The reason is that the window size used to compute the SSM of Figure 3.7 is too large. If features are extracted with smaller window size, closer to the walking period, the *paths* that indicate the recurrence of shapes are visible. Figure 3.8 (right) shows the SSM built from the segment B of the original time series, with a window size of 10 samples and an overlap of 95 %. The matrix shows the *paths*, from which it is possible to extract the periods with the similarity function (sf_B).

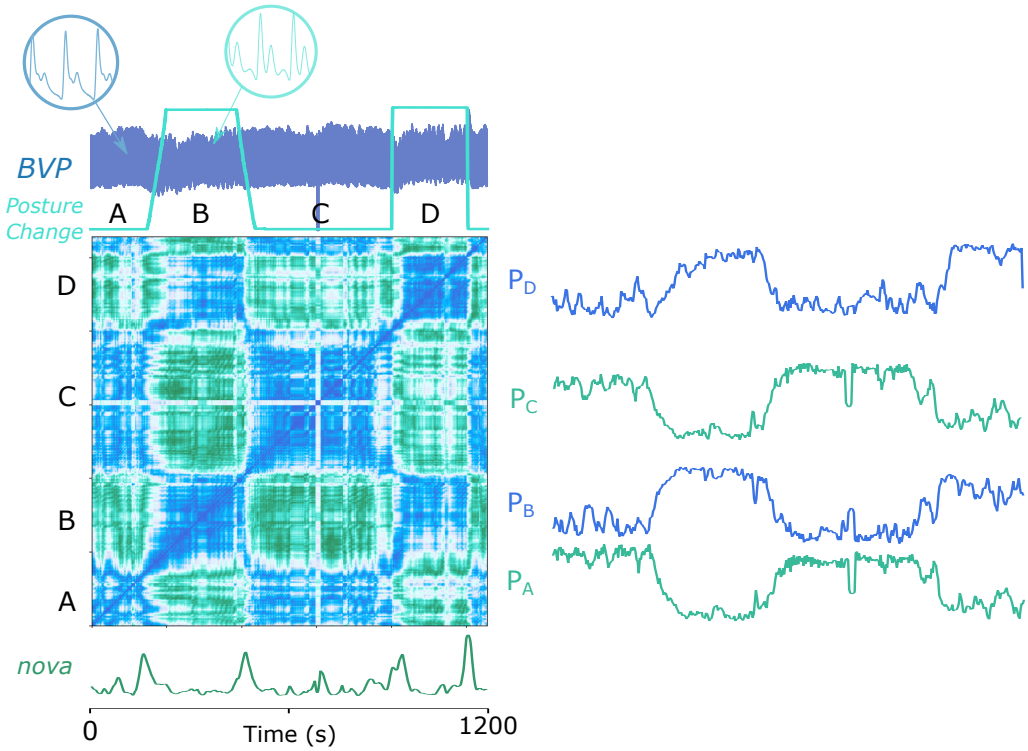


Figure 3.9: (a) ABP signal novelty search. The parameters used were a window size of 5000 samples, with an overlap of 95% and a kernel size of 200 samples.

3.4.2 Use-Case 2 - Medical domain

In the medical domain, there are several examples of structural information to retrieve. Some signals are periodic, such as the ECG, the ABP or the Respiratory Inductance Pletismography (RESP) signals. When acquiring this type of data, several instances might reflect unexpected changes, either because of physiological responses, medical disorders or to the sensing process (such as noise from sensors, motion artifacts, sensor detachment, etc...). Here we show two examples of physiological changes in two periodic signals.

The ABP signal can change due to postural changes. An experiment was conducted to study this effect and is available at Physionet [31, 26]. Figure 3.9 shows the process of segmenting the ABP signal based on postural changes, signaled with the ground truth (as the square signal). The change points are well perceived by the proposed strategy. The reader can notice that the shape of the ABP signal in each regime is very similar, being hard to notice by the human eye where it happened. This is an interesting result considering that it solely relies on the ABP signal for this detection. It is also 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 5000 samples, which is much larger than the period length. However, we can perform this periodic segmentation if using a smaller window size (in this case of 250 samples). This process is illustrated on Figure 3.10 where a segment of the original signal of Figure 3.9 (the first 10000 samples) is computed into the SSM. The resulting *sf* shows the periods of the ABP signal quiet well.

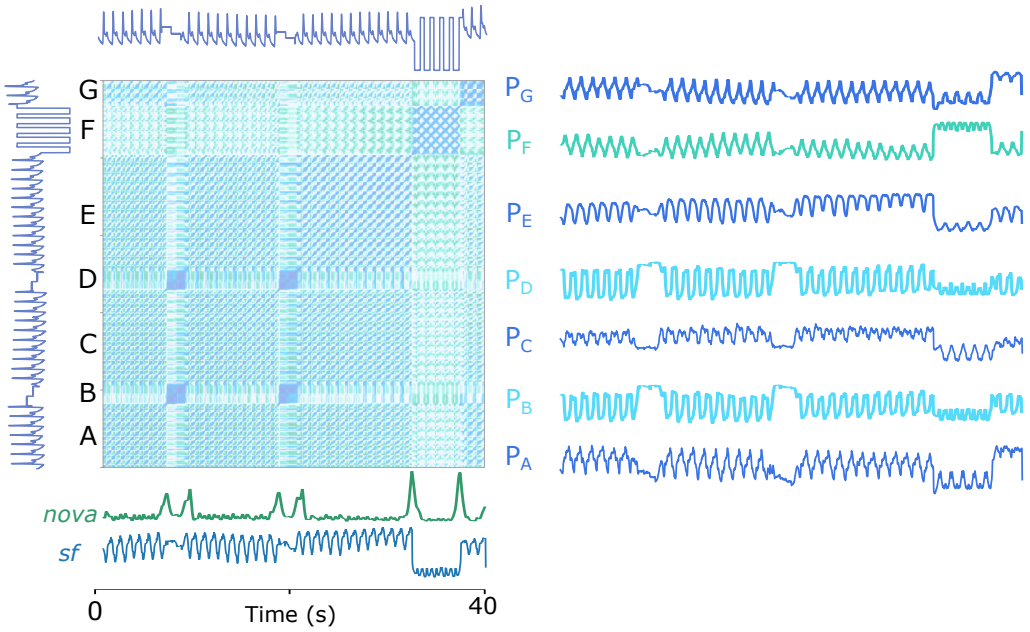


Figure 3.10: ABP signal. It represents the first 10000 samples of the signal from Figure 3.9. The parameters used were a window size of 250 samples, with an overlap of 95% and a kernel size of 200 samples.

The SSM of Figure 3.9.a also shows which segments are similar to each other. The blue colors of the matrix indicate high similarity and it is presenting that segments from the same posture are more similar. To show this we computed the similarity profiles of each segment (as if segmented by the *nova* function) and show that the corresponding sections would be well clustered based on these profiles ($P_A = P_C$ and $P_B = P_D$). In the same way, the similarity profiles of the signal of Figure 3.10 highlight also the similarity between segmented *subsequences*. Profiles with a similar shape can be grouped together such that we can understand that $P_A = P_C = P_E = P_G$ and $P_B = P_D$.

The same happens on the ECG signal from Figure 3.11.right. It displays the presence of a condition called *pulsus paradoxus*, which is an exaggerated fall (>10 mmHg) in the patient's blood pressure during inspiration [50]. This also can occur when the patient's changes sleeping posture after a heart surgery[24]. Detecting when these cases occur is an important task. The present example is a case of an ECG that exhibits this condition. Again, the reader can notice that the change point is hardly perceivable by the human eye, but the proposed strategy can clearly show the difference between both regimes.

In addition to the novelty detection, it is visible on the first segment of the signal previous to the *pulsus paradoxus* occurrence, that there are smaller segmentation points. Zooming into the signal (segment A), it is not clear what this can be, but zooming further into segment B makes it clear that there are minor changes on the ECG due to additional noise on some segments. This method would be able to segment it to.

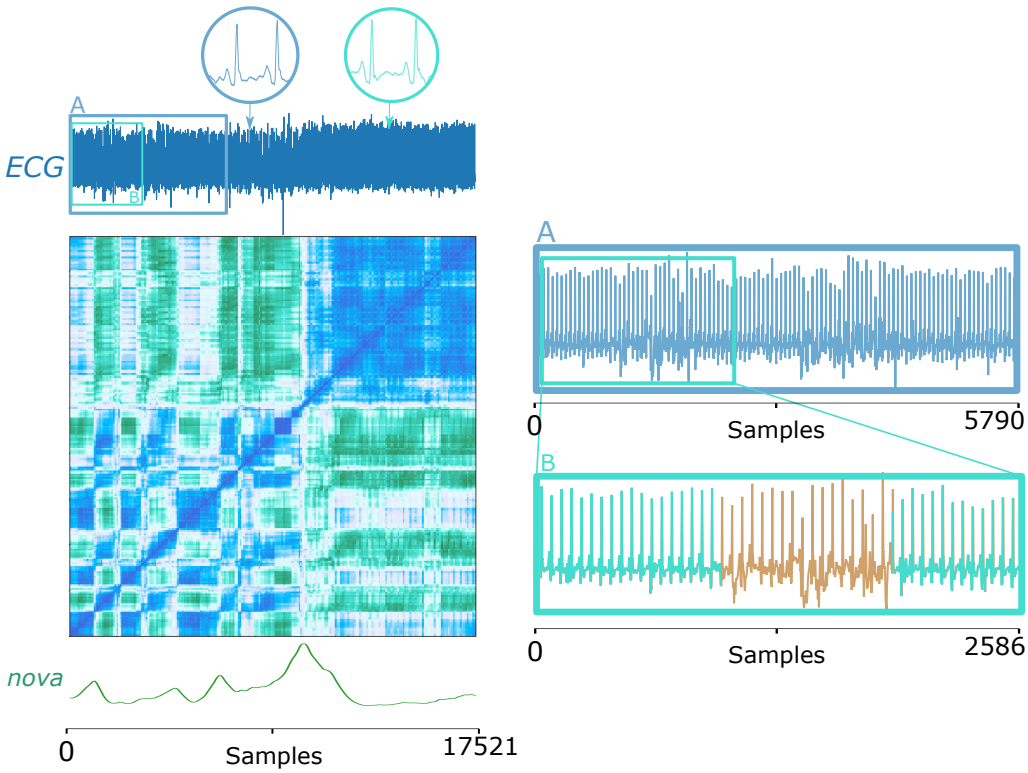


Figure 3.11: ECG signal with a *pulsus paradoxus* condition starting at the 10000th. The SSM shows the two modes of the ECG. Highlights of each mode are presented with the circle zoomed-in thumbnails. In more detailed are also shown segments A and B, which highlight an area where the SSM indicates possible changes.

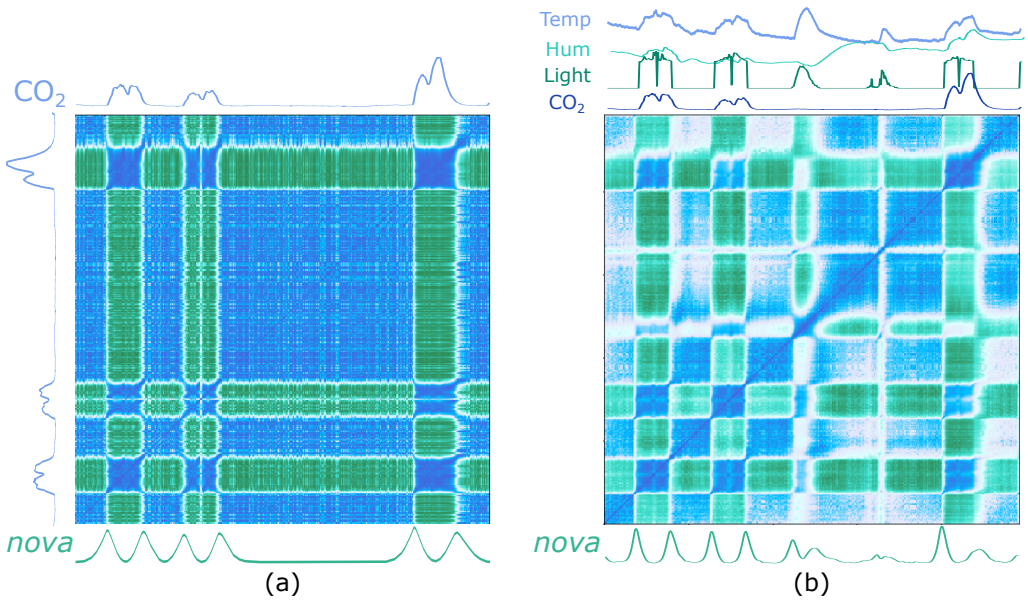


Figure 3.12: Proposed method applied on "*Occupancy*" record of Dataset **??**. (a) A single time series of the record is used to extract events; while in (b) the SSM is computed with features extracted from the four available time series.

3.4.3 Use-case 3 - Multidimensional

The proposed method accepts both single and multidimensional records. The difference regards the number of features extracted. As presented in Figure 3.2, the same set of features is extracted for each time series of the record and combined in the F_M.

Using a single time series of a multivariate record is optional and depends on the detection's purpose. In some cases, using a single time series from a multidimensional record can lead to missing relevant events undetected. An example of this can be seen on Figure 3.12.a with record *"Occupancy"* from Dataset **??**.

The record is a multi-dimensional time series that measures room occupancy based on temperature, humidity, light, and CO_2. All events can only be detected if using several time series of the record [12]. In Figure 3.12.a, a single time series was analyzed by the proposed method to detect relevant events, while Figure 3.12.b is the result of using all the time series of the record.

3.5 Time Series Profiling

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

Following the presented work, we studied how to use it for a meaningful summary of time series. This process is inspired by methodologies that exist in other scenarios for data summarization techniques with statistical analysis, such as the available methods from the *pandas python library*: *pandas.profile()* and *pandas.describe()*. These methods 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.

3.5.1 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. Several examples from other domains can be used as inspiration on how to join all these elements in a compact, expressive and intuitive way.

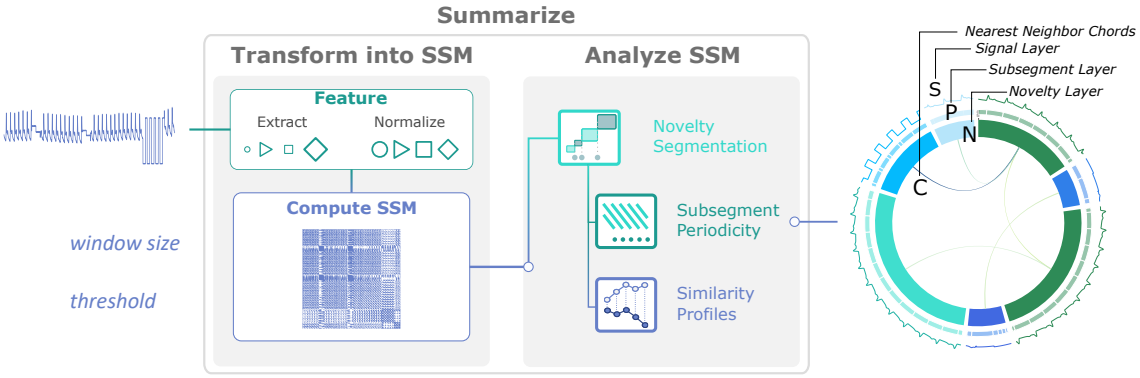


Figure 3.13: Steps to use the aforementioned methods on the SSM to summarize the time series into a circular plot with C - chords that connect nearest neighbors, S - the signal layer, P - the subseqment layer and N - the color coded novelty layer. Each of these elements of the summarized plots are built based on novelty segmentation, subsegment periodicity check and similarity profiles.

3.5.2 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 [40, 74]. 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.

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

3.5.3 A step by step example

The steps are indicated in Figure 3.13 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 3.14, 3.15 and 3.16 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 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 3.14.

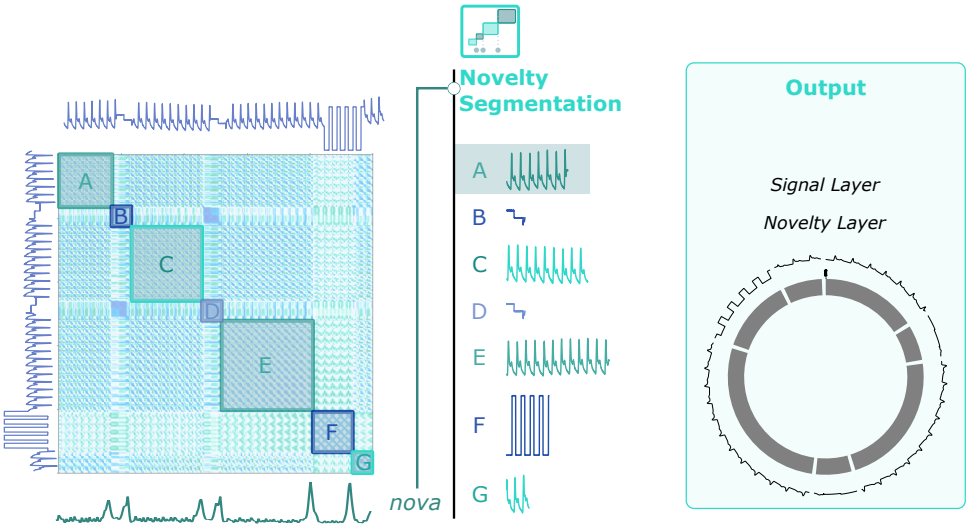


Figure 3.14: How to reach a standard format. This step applies the novelty search method to find the segmentation points. The signal is broken into A to G segments.

Further, the segments are compared with the *similarity profiles* of each segment. Figure 3.15 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 3.15. The dendrogram shows that

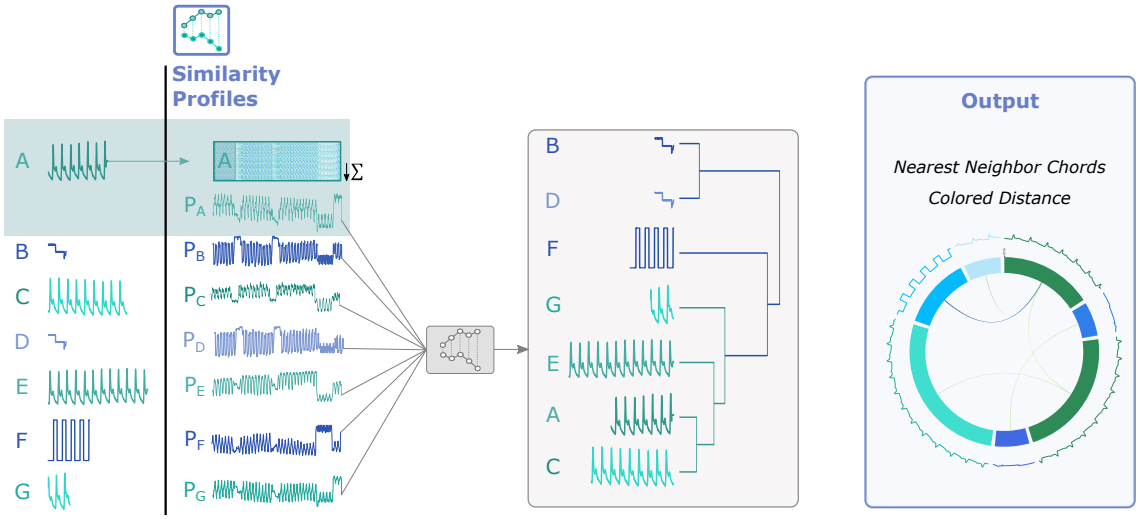


Figure 3.15: From each segment, a similarity profile can be computed. The pairwise euclidean distance is applied to these profiles, from which a dendogram is created. From this association, layer C can be drawn and the colors of the novelty layer can be added. Segment A is highlighted for the next step.

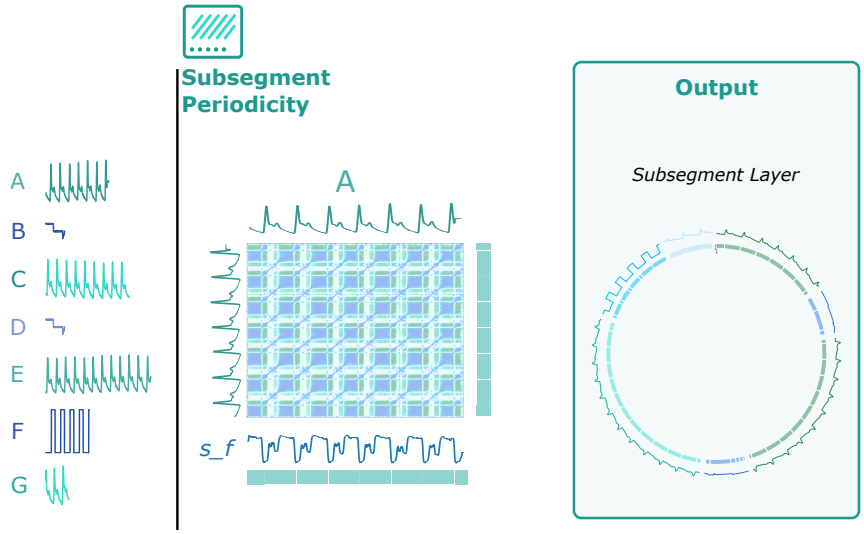


Figure 3.16: After finding the first layer of the segmentation points, the periodicity of each segment can be checked and added as a sublayer of segments, adding layer P. In this case, segment A is the example.

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.

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

4

Language for Time Series Data Mining

In "Pattern Recognition: Human and Mechanical", Satosi Watanabe defines a pattern as a vaguely defined entity that is the opposite of chaos, to which a name can be given [73]. The recognition of these entities immersed in chaos is well performed by the human brain by distinguishing or finding similarities in features that characterize a certain pattern, being an innate capability for decision making. This capacity is also revealed in the search for patterns in time series, as data scientists can visually understand where these patterns occur with previous training and express it linguistically.

Human language is foremost a means of communication, in which the information is represented by sentences, composed of words that can be broken into sequences of symbols. Time series are, in their turn, carriers of information about a certain measure, as sequences of ordered real domain numerical data observed during a given temporal interval. For instance, analysts have a visual perception of the morphological behavior of time series and can describe it in such a way that another subject understands which portion of the time series he/she is referring to. As we mentioned in Chapter 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, the reader can associate words with specific properties, such as `peak` can be related with *slope* and `high` related with *amplitude*.

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

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

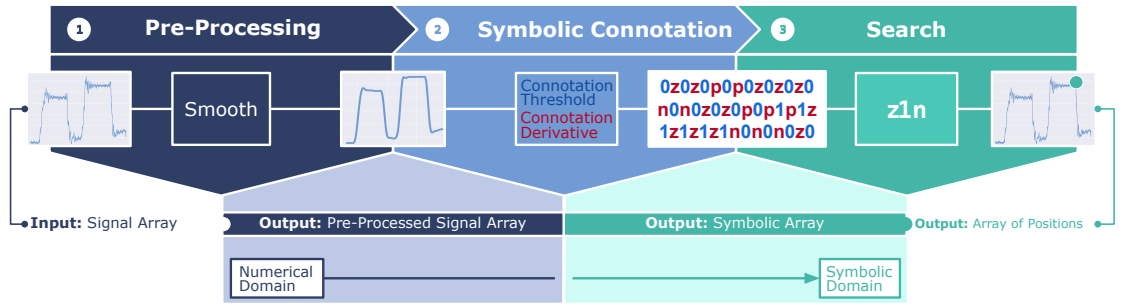


Figure 4.1: SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search. This Figure also provides an example with the sequence of changes that occur in each step.

word, which can be used with additional *operators* to search for the desired patterns with natural language.

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

4.1 Syntactic Search on Time Series

In Chapter 2 was presented the well-known SAX, which is a symbolic representation of the time series based on the 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... 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 regex for searching the desired pattern. The proposed tool focuses on the knowledge retrieved from the visual interpretation of the signal that enables the search for the desired patterns by writing a regular expression that parses the syntactic representation of the signal.

The proposed tool, SSTS, uses symbolic sequences to perform each of the three steps that compose it: *pre-processing* - prepare the signal to highlight the desired patterns and ease the search process; *connotation* - 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; and *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 accelerometer 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 **??** 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.

4.1.1 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 [46]. 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 [29], 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 4.1.

In the example of Figure 4.1, the pre-processing step uses the following string: "Sm 500", which indicates that a smooth filter using a window with a size of 500 samples is applied to the signal. The resulting signal is shown under the original one. The area sectioned in the plots represents the segment of the signal that will be represented in subsection 4.1.2.

4.1.2 Connotation - The Symbolic Time Series

In semiotics, it is described that the connotative aspect of an image or a sign corresponds to the extraction of meaning (sometimes called the *second meaning*), by the personal interpretation of its traits and characteristics [75]. The *connotation*step is ehe same connotative aspect but of a time series. In this, the interpreter (the analyst) has made his personal analysis of the time series and retrieved the necessary information to search for the desired patterns.

The symbolic *connotation* step generates a sequence of *characters* by extracting properties of the signal that are based on a conversion rule the user defines. Each of these conversion rules are designated a *connotation* method. These are responsible for converting each sample of the signal into a *character* or group of *characters* that represent the state of the sample for that conversion rule. Each of the *connotation* methods is related to specific attributes of the time series that are considered relevant for the search procedure. The

Table 4.1: 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

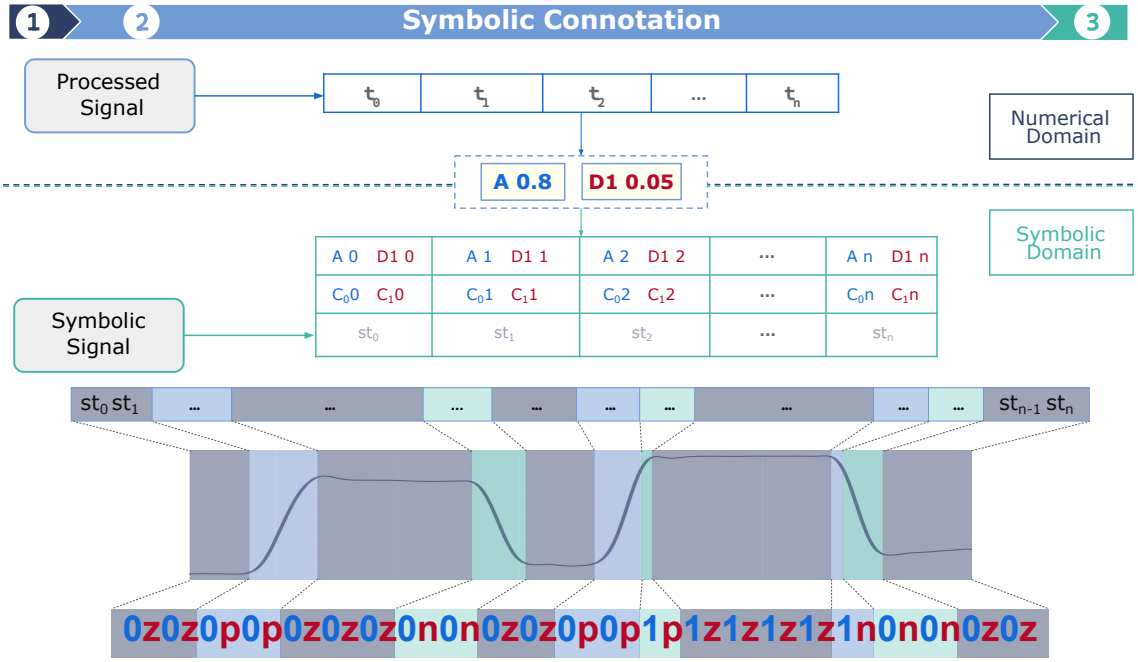


Figure 4.2: Here are highlighted the second stage of the process. It is the moment the signal is transformed from the numerical domain to the symbolic domain. A represents the amplitude method (blue) while $D1$ is the first derivative (red). The exemplified signal plus the symbolic translation are presented.

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 signals, in which cases the string is a combination of *connotation* methods corresponding to each time series, returning a group of *characters* for each sample of the time series.

The set of connotation methods can be defined by the user and added as needed. A base list of connotation methods and the corresponding symbols used for the examples presented in the paper are listed in Table 4.2.

Table 4.2: 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 is'n'; else, t'_i is "z".
D2	(T, *thr*)	["D", "C", "z"]	Second derivative of signal T (T''). If t''_i > *thr*, t''_i is 'D'; elif t''_i < - *thr*, t''_i is 'C'; else, t''_i is "z".
AD	(s, *thr*)	["1", "0"]	Determinates if amplitude from a minimum to a maximum and vice-versa is superior to *thr*. If so, t_i is "1", else, t_i is "0".
SA	(s, *thr*)	["r", "R", "f", "F"]	Amplitude of a slope segment. The characters are case sensitive, being ***r*** for a low rise and ***R*** for a high rise. The same for falling (***F*** or ***f***).
SS	(s, *thr*)	["r", "R", "f", "F"]	Speed of the signal measured by the amplitude on the first derivative. When the sample is quicker than the threshold it is converted to "R" - quick rise or "F" - quick fall, whereas when slower it is converted to "r" - slow rise or "f" - slow fall.

The symbolic connotation step of the exercise of Figure **??** demonstrates the latter

Figure 4.3: 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 first derivative.

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

4.1.3 Expressive Syntactic Search

The last step of the process involves searching for the desired pattern in the generated symbolic time series with a regular expression. This string is governed by the rules of regular expressions from the *alternative regular expression Python module* [21] and benefits from all the functionalities and meta-characters typically used with this tool, which can be recalled from Chapter 2. The search procedure returns the intervals at which positive matches occurred. It is relevant to note that it has been decided to use regex, although any other parser, conveniently combined with the previous steps, might be used for the same purpose.

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

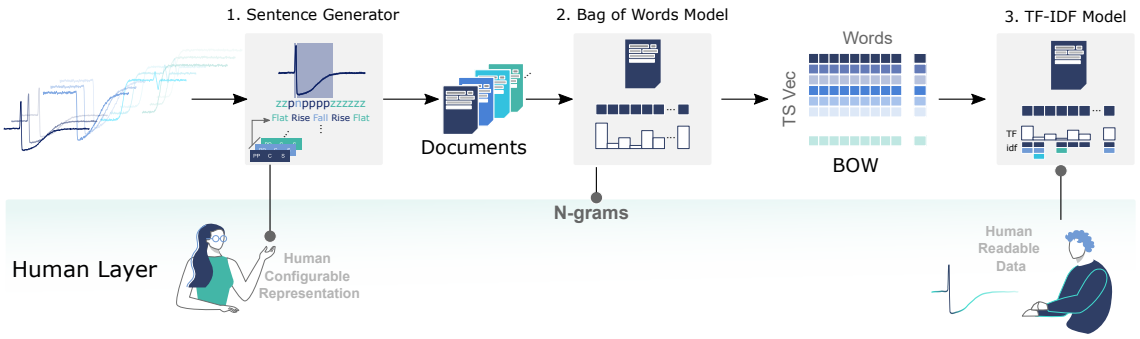


Figure 4.4: Steps of transforming time series into documents and corresponding BoW and TF-idf matrices for posterior classification. The steps are *sentence generation*, *bag of words model* and *tfidf model*. The human layer indicates when there can be human intervention. The steps are followed with an example.

that the signal, at some point superior to the threshold will start to decrease.

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

4.2 Towards Interpretable Time Series Classification with SSTS

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

In text mining, the difference between text *documents* is traditionally performed with a feature extraction process on text, such as BoW or TF-idf, returning a statistical representation of the words or ngram. *Documents* with different words and sequences of words will have different weights on these models, which is used to perform a class separation on the *documents*. In this work, we take inspiration from this process and apply it to time series. 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.

4.2.1 SSTS to Generate Time Series Documents

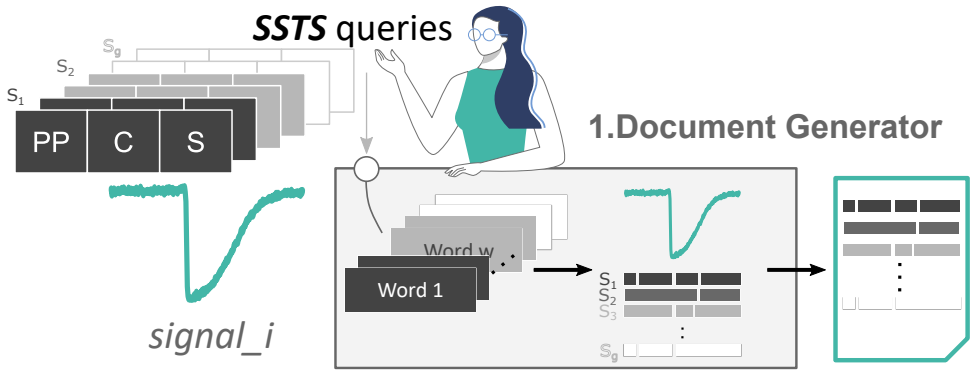


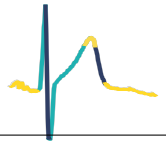
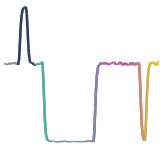
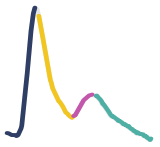
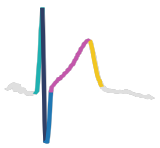
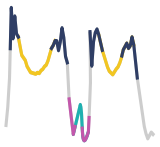
Figure 4.5: Steps for the generation of sentences from a raw time series and organization as a document. Here: PP - Pre-processing, C- connotation and S - Search are each SSTS queries used to search for the patterns and attribute the matched *subsequences* the corresponding *word*.

Samples of the signal are converted to characters, patterns are searched to form words, and these are ordered by their time index. With this, the time series can be translated into sentences and be described just as a human would describe it. For instance, this signal can be very broadly translated into `Flat Rise Fall Rise Flat` or `Flat Peak Valley Flat`, while this signal would be translated into `Flat Fall Rise Flat` or `Flat Valley Flat`. This type of description is easily performed with SSTS, as showed on Figure 4.6.

In order to perform this description, the words have to be associated with a specific *SSTS query*. Each query is a regex pattern that searches the match on the time series and attributes it to its corresponding *word*. For instance, the query {PP: Sm 25, C: D1 0.05, S: p+} searches for moments where the time series is increasing and attributes to these *subsequences* the word *Rise*. A specific set of SSTS queries are used to build a full description of the signal dynamics in higher leveled structures, mostly with amplitude and derivative *connotations*. The selection of 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 is the customizable step of the process, in which the analyst can include his/her intuition.

The ability to distinguish time series with text will be as good as the descriptive richness of the generated sentences. Depending on the differences between the time series being classified, a simple description based on the sign of the signal's derivative might be enough. However, in some cases, the overall dynamic of signals might be dissimilar 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 4.3, 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 event new connotation methods and queries can be created

Table 4.3: The connotation variables, search regular expressions and corresponding words assigned to the pattern searched. The parameter *m* indicates the size, in samples, of the difference between a peak or a plateau, *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	

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 4.6. 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.

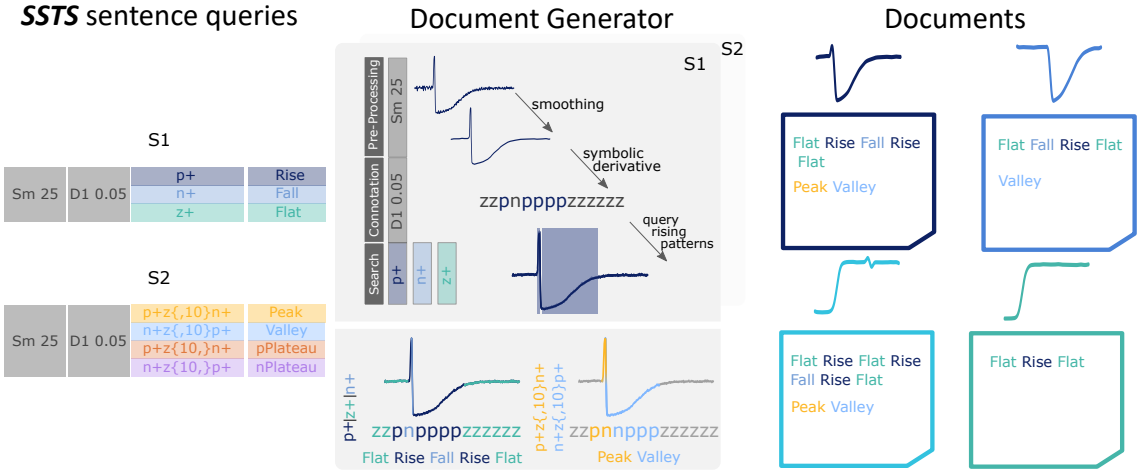


Figure 4.6: (Top) Using SSTS to detect the rising stage of a time series. Each step of the process is written described as follows: (1) pre-processing: *Sm* is the function *Smooth* with a window size of 25 samples; (2) connotation: *D1*, indicates the first derivate, from which each sample is converted to *z* - Flat, *p* - rising and *n* falling; (3) search - regular expression *p+* searches for all sequences with 1 or more *p* characters. (Bottom) Example of sentence generation. Using the other search queries (*p+*, *n+*, *z+*), we can find the derivative patterns and convert it into ordered words.

Having now a method to translate time series into text, we can use text mining methods directly on the *documents*. Typically, text data is vectorized with the BoW or TF-idf models. This brings the reader to the second step of the process.

4.2.2 Vectorization of Time Series Documents

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

After converting the time series into *documents*, the text is analyzed to build a matrix of *word* or *n-gram* frequencies, the 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 **??** shows the vectorization of a 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 tasks as well as supervised ones, for topic modeling and keyword extraction. The BoW model can also be trained with Naive Bayes Classifiers, Support Vector Machine or Linear Regressors [56]. In addition, the BoW model feature matrix can be converted into the TF-idf feature matrix, which can then be

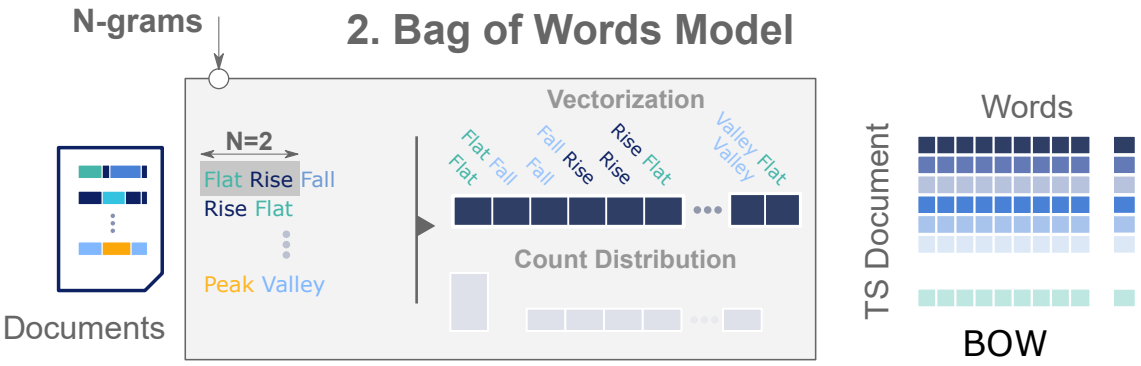


Figure 4.7: Process to transform a time series document into a vectorized representation of words and n-grams. The user can set the size of the n-gram. The documents are transformed into the BoW having a count distribution, which is the tf.

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 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 2.19).

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 time series document can be used as a distance measure to compare them. In Figure 4.8 is showed the distribution of *word's* weights for four different time series documents of the *Trace* dataset, 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 (RSVP) 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),

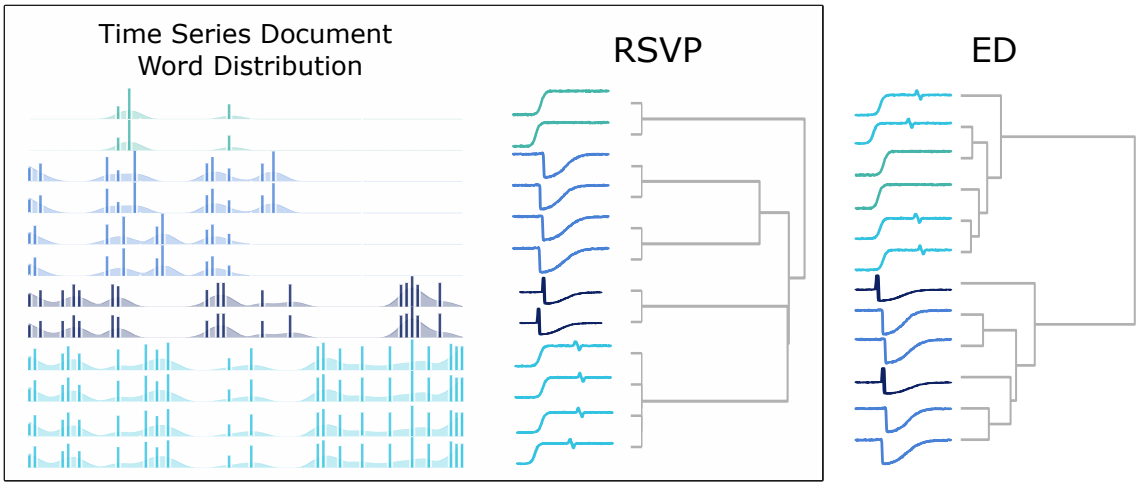


Figure 4.8: Comparing the clustering process of the ED and the proposed symbolic method. In the proposed method, each signal has a word distribution vector, represented on the left. The signals are clustered based on these.

increases the distance between signals that apparently have the same shape. On the other end, the proposed strategy follows a distance measure based on the presence(absence) of specific shapes, as well as how these are ordered (if *n-grams* are used). This means that classes 1 and 2 will not be mismatched, because signals from class 1 have a peak whereas the ones from class 2 don't.

4.2.3 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 [64].

The process to highlight the areas of the signal that are more relevant and able to explain the difference between classes is illustrated on Figure 4.9. From the BoW model, the TF-idf model is computed, following equations 2.17, 2.18 and 2.19. 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:

$$(a, b) = ssts(t_j, w_i) \tag{4.1}$$

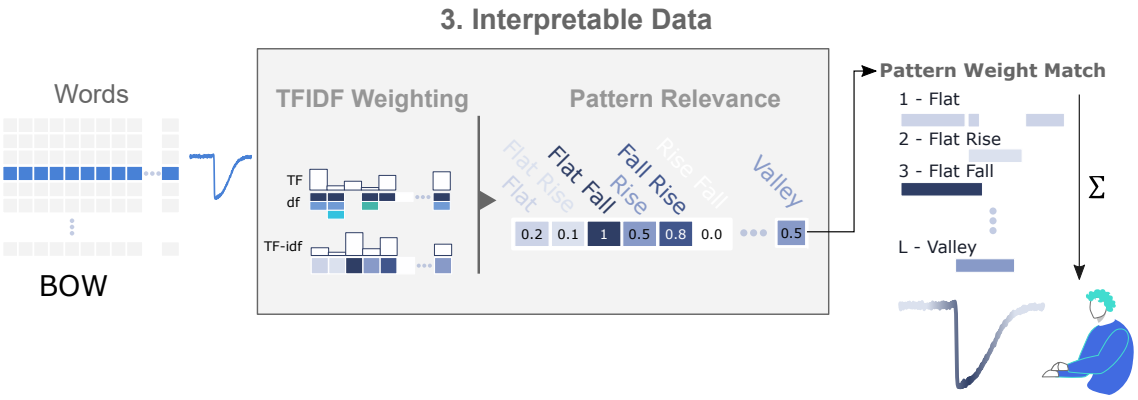


Figure 4.9: From the TF-idf matrix has weights for each word or n-gram (pattern relevance). The search of these words and cumulative attribution of weights on the segments matched can represent a feedback tool. In this case, the valley of the signal is highlighted.

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

$$WTS_i(a,b) = WTS_i(a,b) + h_{ji} \tag{4.2}$$

In Figure 4.9, the exemplified signal has a higher relevance on the `valley`, being the most characteristic element of that time series.

4.3 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. Nevertheless, this process requires that the analyst has some background knowledge on regex, which is fine for a computer scientist but might be harder for an analyst outside of the computer science domain. In order to contribute to the democratization of pattern search on time series, we propose another method for pattern search, but in this case, it is based on natural language and linguistic operators, from a feature-based representation of the signal instead of a symbolic representation of it. The process to find a pattern on a time series is very similar to how a user searches for *web-pages* on the *Google Search Toolbar*, using *keywords* and *operators*.

4.3.1 "Googling" Time Series Patterns

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

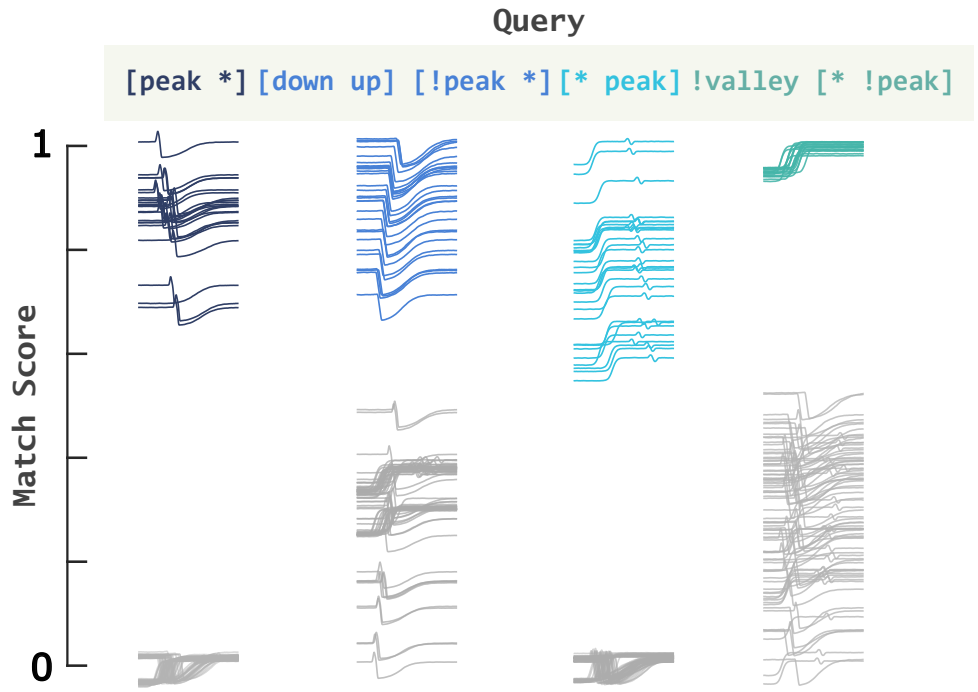


Figure 4.10: Using our proposed query language, we can create short, intuitive natural language queries to rank the 100 exemplars in Trace, separating our sought class from the remaining data. Here ⋆ is *anything*, ! is *not*, and *square brackets* are a grouping operator (more details in Section 3).

process. However, let us take into account this simple *keyword* based search process and apply it to pattern search on time series.

As we mentioned above, patterns can be described with *words*, which is done because we associate this text words with specific properties or the presence of specific shapes on the pattern. In that sense, we propose to create a search paradigm that relies on words represented by features extracted from the signal. Informally, if a user wishes to find examples of a particular behavior, he/she can simply *Google it*, by describing the data he/she expects to see, given that behavior. For example, we considered the four-class Trace dataset, which has been studied in more than 1,000 papers. As Figure **??** shows, it is possible to use our system to create queries that can separate each of the four classes from the rest of the data. This separation is possible simply by describing with *words* the presence/absence of structures. Further, we will explain in detail how the process is done, which are the operators that we designed, and how queries can be written. Still, for now, we will explain the meaning of each query of this example. For instance, signals from class 1 () match well with query `[peak *]`, which translates into *has a peak on the first half and anything on the second half*. The method sorts the signals based on this query and gives a very low score for all signals that do not have a peak 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

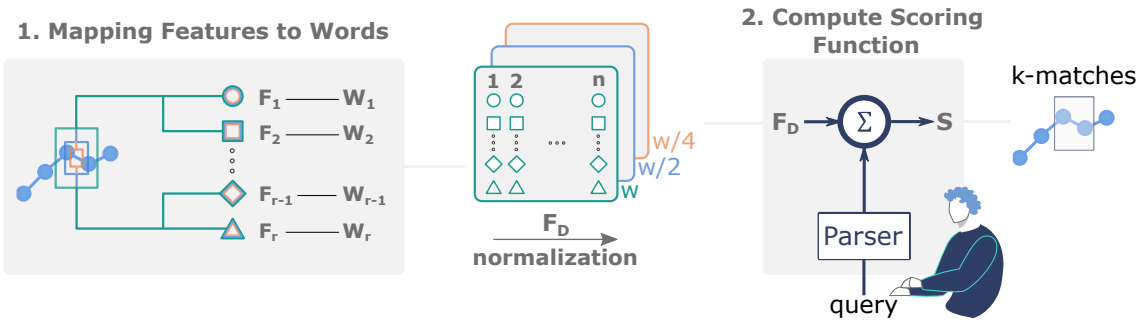


Figure 4.11: QuoTS steps. It shows the feature extractino process and matching each feature to words ($W_1, W_2, etc...$). These features are computed in three dimensions (w, w/2 and w/4). The user can input a query to search for a specific pattern. This query will be scored based on the parser and features. It then returns the top *k*-matches.

has not a valley and has no peak on the second half.

The overall steps of the process are illustrated in Figure 4.11. In order to perform the search with natural language 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. 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 representation is expressive enough to discriminate specific behaviors on a time series and that it moves towards a democratized time series analysis process, where analysts from other scientific domains, not familiar with programming languages, can also perform high-level pattern search.

4.3.2 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

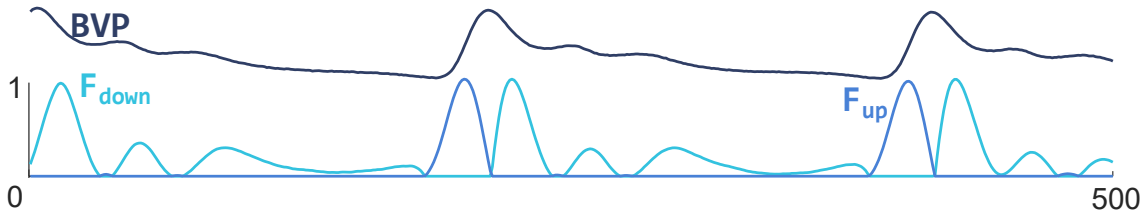


Figure 4.12: Example of a word feature vector. In this case F_{down} and F_{up} represent a negative and positive slope values.

feature vector with different window lengths, based on w: $W_1 \rightarrow w$; $W_2 \rightarrow \frac{w}{2}$; $W_3 \rightarrow \frac{w}{4}$; with the intent of matching ordered sequences of words inside a subsequence with the grouped followed by an 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:

$$!W = 1 - W \tag{4.3}$$

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 4.3. 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 4.4).

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.

4.3.3 Linguistic Operators

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

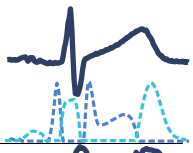
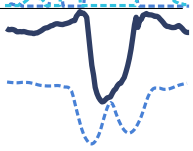
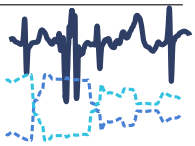
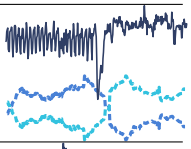
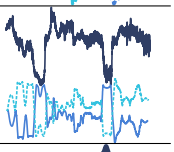
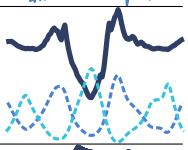
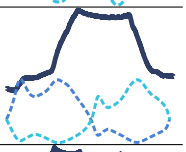
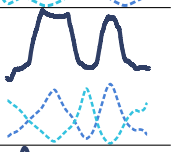
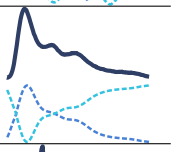
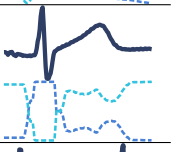
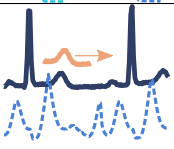
Web search engines have many operators at the user's disposal, but since a list of words is usually powerful enough to retrieve and correctly sort most of the desired results, very few (or none!) are often used. We believe that this is the case for this application as well but acknowledge that simple operators can make the query more natural and come in handy to perform conjunctions between features and multiple dimensions, such as *temporal logic* or *negation*. For instance, we often rely on *temporal logic* to express the presence of a shape in regards to another: `the peak that comes after the valley`, or even use the absence of a property: `it does not have a peak`. We also typically express the order 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 4.3. 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 4.3 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

Table 4.4: 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 the 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 2.14. (`simple = !complex`)	
Symmetric (**Asymmetric**)	The normalized ED (2.8) to the *subsequence*'s horizontally flipped self.	
Peak (**Valley**)	The logarithmic normalized ED (2.8) to the template of a peak (valley), modulated by a gaussian function.	
StepUp (**StepDown**)	The logarithmic normalized ED (2.8) to the template of a step-up(down) function.	
PlateauUp (**PlateauDown**)	The logarithmic normalized ED (2.8)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 on the query language.	

can be single words, multiple words, or even queries for different dimensions of the time series.

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

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

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

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

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

4.3.4 Natural Language Query for Time Series

As illustrated on Figure 4.11, 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.

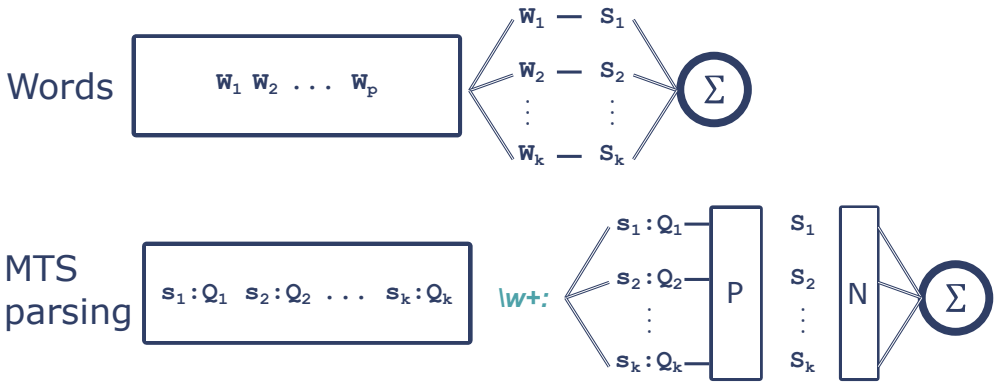


Figure 4.13: Process to parse the input text query. When having simple space separated words, each corresponding word feature vector is summed together. When in multidimensional mode, there is the *signal id* first.

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

When the `followed by` operator is used, the query is parsed in the *elements* that come before and after it (this is applicable either if the operator is used for intra-signal or inter-signal search). For this, the following 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 **??**).

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 4.13 and **??** give a visual aid to understand the parsing process for each case.

The simplest case for a query would be a sequence of *words*. In this case, each *word feature vector*, associated 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

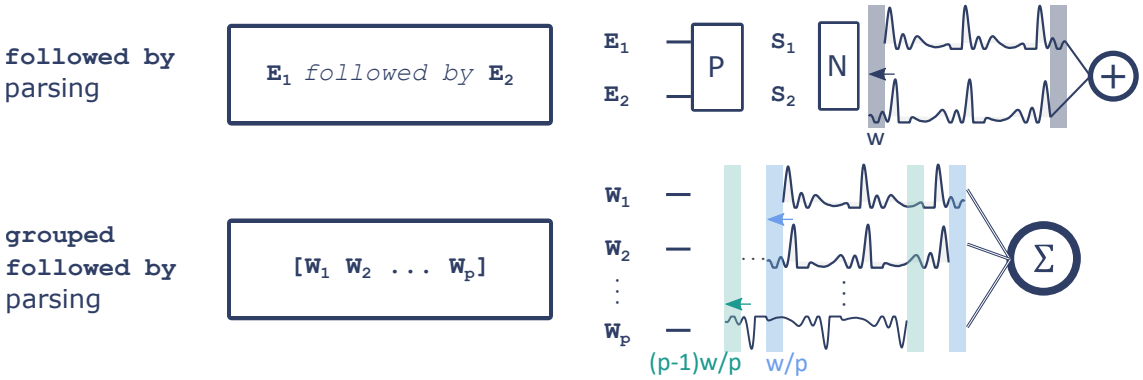


Figure 4.14: Second order of the parser. In case the `followed by` operator is used, the elements (E_1 and E_2) are parsed (P) and word feature vectors are summed, but the E_1 has to be delayed to count as the "previous" instance (w represents the selected window size). When using the more specific `grouped followed by` operator, square brackets are used and represents 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).

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 4.13), 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 4.13) 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 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.

Bibliography

[1] 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. 11).

[2] 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. 19).

[3] 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. 28).

[4] 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 p. 1).

[5] 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 p. 3).

[6] M. Barandas et al. "TSFEL: Time Series Feature Extraction Library". en. In: *SoftwareX* 11 (2020-01), p. 100456. ISSN: 23527110. DOI: 10.1016/j.softx.2020.100456. URL: https://linkinghub.elsevier.com/retrieve/pii/S2352711020300017 (visited on 2020-04-03) (cit. on p. 31).

[7] 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. 14–16).

[8] 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 p. 1).

[9] 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. 28, 32).

[10] 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. 28).

[11] 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).

[12] 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. 9, 43).

[13] 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).

[14] 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).

[15] 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).

[16] D. Crystal. *Making Sense of Grammar*. Pearson Education, 2004-01. ISBN: 9780582848634 (cit. on p. 2).

[17] 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. 19).

[18] 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. 33, 34).

[19] 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. 13).

[20] 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 p. 33).

[21] P. S. Foundation. *regex 2017.09.23*. 2017. URL: https://pypi.python.org/pypi/regex/ (cit. on p. 52).

[22] J. Friedl. *Mastering Regular Expressions*. 3^{rd}. O'Rilley Media, Inc, 2006 (cit. on p. 22).

[23] 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).

[24] 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 p. 41).

[25] 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 p. 17).

[26] 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 p. 40).

[27] V. Guralnik and J. Srivastava. "Event Detection from Time Series Data." In: 1999-01, pp. 33–42. DOI: 10.1145/312129.312190 (cit. on p. 9).

[28] 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).

[29] 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. 49).

[30] 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. 17).

[31] 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. 40).

[32] 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).

[33] 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).

[34] 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).

[35] 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 p. 27).

[36] 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. 25).

[37] 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. 27).

[38] 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. 15).

[39] 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 pp. 12, 13).

[40] 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 p. 44).

[41] 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).

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

[43] 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. 13, 14).

[44] 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).

[45] 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).

[46] R. G. Lyons. *Understanding Digital Signal Processing*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1996. ISBN: 0201634678 (cit. on p. 49).

[47] 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).

[48] 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. 19).

[49] 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).

[50] V. MN. Dam and B. Fitzgerald. *Pulsus Paradoxus*. 2022-01. URL: https://www.ncbi.nlm.nih.gov/books/NBK482292/ (cit. on p. 41).

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

[52] M. Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015. ISBN: 978-3-319-21944-8 (cit. on pp. 18, 28, 33, 34).

[53] 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. 18, 33, 34).

[54] 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).

[55] 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. 28, 32).

[56] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 21, 56).

[57] D. Preston, P. Protopapas, and C. Brodley. "Discovering arbitrary event types in time series". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 2.5-6 (2009), pp. 396–411. DOI: 10.1002/sam.10060. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sam.10060. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.10060 (cit. on p. 9).

[58] 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. 3).

[59] 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 p. 1).

[60] 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).

[61] 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).

[62] 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).

[63] 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).

[64] 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 p. 58).

[65] H. Shatkay. "The Fourier Transform - a Primer". In: *Technical Report CS-95-37*. Brown University, 1995 (cit. on p. 11).

[66] 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).

[67] 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).

[68] 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).

[69] 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).

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

[71] 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 p. 3).

[72] 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).

[73] S. Watanabe. *Pattern Recognition: Human and Mechanical*. New York, NY, USA: John Wiley and Sons, Inc., 1985. ISBN: 0-471-80815-6 (cit. on p. 47).

[74] 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 p. 44).

[75] A. Wilden. "The Rules Are No Game: The Strategy of Communication by Anthony Wilden". In: (1987) (cit. on p. 49).

[76] 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).

I

Annex 1 Lorem Ipsum

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

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

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

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

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

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