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) [**novathesis-manual**].

Dedicatory lorem ipsum.

Acknowledgements

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

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

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

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

Abstract

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

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

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

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

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

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

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

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

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

Resumo

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

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

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

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

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

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

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

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

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

Contents

List of Figures xiii

List of Tables xx

Acronyms xxii

Symbols xxiii

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 . . . 6

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 Complexity . . . 25
2.8.4 Comparing Algorithms . . . 26

3 State of the Art **28**
3.1 Information Retrieval from Time Series . . . 28
3.1.1 Segmentation and Event Detection . . . 28
3.1.2 Self Similarity Matrix . . . 29
3.1.3 Summarization . . . 31
3.1.4 Text based Query Search . . . 31
3.1.5 Summarization . . . 32
3.1.6 Classification . . . 33
3.1.7 Search by Query . . . 35
3.2 Symbolic Representation . . . 36
3.3 Dictionnary-based Classification . . . 36
3.4 Search by Query . . . 36

4 Data Description and Management **37**
4.0.1 Classification Benchmark - UCR . . . 37
4.0.2 Kaggle . . . 37
4.0.3 UCI Machine Learning Repository . . . 38
4.0.4 Physionet . . . 41
4.0.5 Alan Turing CPD Benchmark . . . 42
4.0.6 HCILab Bhevioral Driving Dataset . . . 43
4.0.7 CSL-Share Dataset . . . 43
4.0.8 Industrial Job Dataset . . . 44
4.1 Notes for Ground Truth Annotations . . . 45

5 Unveiling the *Grammar* of Time Series **46**

5.1 The Problem . . . 46
5.1.1 Search Dimension . . . 47
5.1.2 Type Dimension . . . 47
5.1.3 Proposal . . . 47
5.2 Building the SSM . . . 48
5.2.1 Feature Extraction . . . 49
5.2.2 Feature-based SSM . . . 49
5.3 Information Retrieval . . . 51
5.3.1 Novelty Search . . . 51
5.3.2 Periodic Search . . . 53
5.3.3 Similarity Profiles . . . 54
5.3.4 Query Search . . . 55
5.4 Experimental Evaluation in Selected Use-cases . . . 55
5.4.1 Use-Case 1 - Human Activity . . . 55
5.4.2 Use-Case 2 - Medical domain . . . 58
5.4.3 Use-case 3 - Multidimensional . . . 60
5.5 Time Series Summarization . . . 61
5.5.1 Elements with Relevance . . . 61
5.5.2 Compact Design . . . 62
5.5.3 A step by step example . . . 62

6 Language for Time Series Data Mining **65**
6.1 Syntactic Search on Time Series . . . 66
6.1.1 Pre-Processing - Preparing the Data . . . 67
6.1.2 Connotation - The Symbolic Time Series . . . 67
6.1.3 Expressive Syntactic Search . . . 70
6.2 Towards Interpretable Time Series Classification with SSTS . . . 71
6.2.1 SSTS to Generate Time Series Documents . . . 72
6.2.2 Vectorization of Time Series Documents . . . 74
6.2.3 Towards Interpretable Results . . . 76
6.3 Towards Natural Language for Pattern Search . . . 77
6.3.1 "Googling" Time Series Patterns . . . 77
6.3.2 Mapping Features to Words . . . 79
6.3.3 Linguistic Operators . . . 81
6.3.4 Natural Language Query for Time Series . . . 83

7 Results and Validation **87**
7.1 Segmentation . . . 87
7.1.1 Novelty Segmentation . . . 87
7.1.2 Cyclic Segmentation . . . 93
7.2 Pattern Search with SSTS . . . 94

7.2.1 SSTS on selected Use-Cases . . . 95
7.2.2 Measure of Expressiveness . . . 99
7.3 Time Series Classification with HeaRTS . . . 101
7.3.1 Classification Performance . . . 102
7.3.2 Interpretable Data Outputs . . . 104
7.3.3 Discussion of Results . . . 106
7.4 Pattern Search with QuoTS . . . 108
7.4.1 *QuoTS* Matches Gestures . . . 108
7.4.2 *QuoTS* in Selected Use Cases . . . 110
7.4.3 Matching *Known* Shapes with Words . . . 113
7.5 Application to Occupational Scenario . . . 115

8 Conclusion **116**
8.1 Main Contributions on General Topics . . . 116
8.2 Scientific Contributions . . . 118
8.2.1 Unveiling the *Grammar* of Time Series . . . 118
8.2.2 Using Language for Time Series Data Mining . . . 119
8.3 Other Contributions . . . 120
8.3.1 Managing Rotation Plans with Exposure, Diversity and Team Homogeneity . . . 120
8.3.2 MicroErgo - Concept for Personal Assessment of Occupational Risk in Desk/Office Jobs . . . 121
8.3.3 In using Direct Measures for Occupational Health Assessment . . . 121
8.3.4 Volatile Organic Compounds Classification . . . 122
8.4 Scientific Production . . . 122
8.4.1 Journal Publications . . . 123
8.4.2 Book Chapters . . . 123
8.4.3 Conference Proceedings . . . 123
8.4.4 Methods . . . 124
8.4.5 Projects . . . 124
8.4.6 Awards . . . 125

9 Future Work **126**
9.1 Overall Improvements to compute the ?? and Segmentation Process . . . 126
9.2 Unsupervised Automatic Segmentation and Labeling of Time Series . . . 127
9.3 Hierarchical Segmentation of Time Series . . . 127
9.4 Periodic Segmentation . . . 127
9.5 Online Unsupervised Segmentation . . . 127
9.6 Tool for Time Series Profiling . . . 128
9.7 Syntactic Search on Time Series (SSTS) Improvements and Further Applications . . . 128

9.8 Further Developments for QuoTS 129

Appendices

A Appendix 2 - Detailed Results **131**
A.1 Novelty Segmentation 131
A.1.1 UCI3 131
A.1.2 UCI4 133
A.1.3 HAR5 134
A.1.4 Physionet 1 134
A.1.5 Physionet 2 134
A.1.6 Kaggle 134
A.1.7 Alan Turing Dataset 134
A.2 HeaRTS Classification 134

Annexes

I Annex 1 Lorem Ipsum **136**

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 [**pyts**], based on [**paa**]. 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 [**sax**]. 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 Strategies for time series summary found on the literature. These images are taken from the works from [**snippets**, **bitmap**] 32

3.2 A - Diagram for string association. This image is taken from the works from [**arcplots**]; B - Circular plot by OmicCircos. Several layers (Circular tracks) identify genome position, expression heatmaps, correlation between expression and CNV, among other features. The image is taken from the works from *Ying Hu, et al.* [**genomics**]. 33

4.1 Example of the signal for this dataset. It shows the 3 axis of the accelerometer signal and the corresponding labels in each section [**dataset2**, **dataset2_2**]. Yellow areas indicate moments where there was a change on the signal not related with the labeled activity. 38

4.2 Example of the signal for this dataset. It shows one axis for the Accelerometer (ACC) and ?? signals and the corresponding labels in each section [**dataset3**]. In this datasets, labels also highlight posture transitions, such as *Standing to Sitting*. Yellow areas indicate moments where there is activity but the signal was not labeled. 39

4.3 Example of the signal for this dataset. It shows one axis of each type of data acquired, for both smartphone and smartwatch. The activities are highlighted as well and labeled as: A - walking, B - running, C - walking on stairs, D - sitting, E - standing, F - typing, G - brushing, H - eating soup, I - eating chips, J - eating pasta, K - drinking, L - eating a sandwich, M - kicking, N - catching a ball, P - dribbling, Q - writing, R - clapping, O - iron [**dataset4**]. 40

4.4 Example of the Electromyogram (EMG) data and the corresponding hand postures. 41

4.5 • 41

4.6 • 42

5.1 (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. 47

5.2 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. 48

5.3 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 50

5.4 Information retrieval topics explained in this section. 51

5.5 (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.* [**fmp1**, **fmp2**]; (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 [**Dannenberg2008**, **fmp1**, **fmp2**]. 52

5.6 Profiles computed for each segment of the example signal used in Figure 5.3. 55

5.7 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% 56

5.8 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. 57

5.9 (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. . . . 58

5.10 ABP signal. It represents the first 10000 samples of the signal from Figure 5.9. The parameters used were a window size of 250 samples, with an overlap of 95% and a kernel size of 200 samples. 59

5.11 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. . 60

5.12 Proposed method applied on "*Occupancy*" record of Dataset 4.0.5. (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. 61

5.13 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. 62

5.14 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. 63

5.15 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. 63

5.16 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. 64

6.1 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. . 66

6.2 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. 68

6.3 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. 70

6.4 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. 71

6.5 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*. 72

6.6 (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. 74

6.7 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). 75

6.8 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. 76

6.9 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. 77

6.10 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). 78

6.11 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. 79

6.12 Example of a word feature vector. In this case F_{down} and F_{up} represent a negative and positive slope values. 80

6.13 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. 84

6.14 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). 85

7.1 Critical distance diagram comparing the methods used in **[cpd_alan]** (except *RBOCPDMS*) and the *novelty function* (*nova* - highlighted in SeaGreen). The performance measure corresponds to the F1-score for all single-dimension datasets of the benchmark (except the ones highlighted in gray in Figure 7.3). 90

7.2 Segmentation of a walking pattern on a 1D signal with the similarity function (s_f). The window size was 5000 samples and the overlap percentage of 75%. 94

7.3 Example 2 - Solution pipeline of Step Detection example. At the bottom of the figure are summarized the operators and methods used in each step. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - A - Amplitude Comparison, Red - D1 - First Derivative). A dotted line is shown to represent the threshold level. Each color band in the signal corresponds to a specific primitive. A positive match is highlighted with green in the search step. 96

7.4 Example 3 - Solution pipeline of Dicrotic notch detection example. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between methods is represented with colors (Blue - AD, Red - D1). For each color band in the signal, there is a corresponding primitive. The match is highlighted with green in the search step. 98

7.5 Example 6 - Solution pipeline of Stable Lifting Detection. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - AD, Red - D1). The threshold level is identified with a white dotted line. Each color band in the signal corresponds to a specific primitive. The match is highlighted with green in the "Search" step. 99

7.7 right) Classification accuracies for the Bag of Synthatic Patterns in comparison with the euclidean distance combined with 1-NN classifier (1-NN ED) with standard train and test; left) with a 10 fold cross validation. 103

7.8 Classification accuracies for all experimented methods versus the accuracies of the TF-idf method with an Support Vector Machines (SVM) classifier. . . 104

7.9 Interpretable results with highlighted shapes for the *UMD* and *BME* datasets. The *UMD* dataset was classified with with an accuracy of 99.3%, computed with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$. The *BME* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$ 105
7.10 Interpretable results with highlighted shapes for the *Trace* and *TwoPatterns* datasets. The *Trace* dataset was classified with with an accuracy of 100.0%, computed with a $w_s = 25$, $thr = 0.05$ and a $1 - gram$. The *TwoPatterns* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.1$ and a $5 - gram$ 106
7.11 Interpretable results with highlighted shapes for the *Gunpoint* and *ShapeletSim* datasets. The GunPoint dataset was classified with with an accuracy of 99.0%, computed with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$. The *ShapeletSim* was classified with an accuracy of 99.4% with a $w_s = 1$, $thr = 0.05$ and a $1 - gram$ 106
7.12 *UWaveGestureLibrary* subsequence matches with QuoTS. Column-wise are showed the gesture class, the corresponding mean wave for all subsequences, the query used to match the subsequence class and the corresponding match score for all subsequences, highlighting with the color of the specific class. . 109
7.13 ECG use case to identify noisy sections, as well as specific segments of the ECG pattern. The queries are written in text boxes. On the side, an example of a match is showed as a larger pattern. 110
7.14 Examples of the detection of three specific cases of arrhythmia. The ground truth is selected from the annotations of specialists in the area. The queries are presented in text boxes and the found patterns are highlighted. On the side, an example of a match in a larger size is shown. 112
7.15 Example of multivariate patterns on a dataset from auto telemetry with physiological signals of the driver. The queries are written in text boxes. Several matches are highlighted based on the queries written. These events are associated with specific occurrences during the driving session, symbolized by traffic signs. These events were matched by searching the coordinates on the map. **[hcilab]** 114
7.16 Example of searching for a 3 point turn event with QuoTS. 115
7.17 Creating query prompt data by puppeteering a car model. The puppet data for a 3-point turn (smoothed with 10 samples moving average) is presented on the left, while the real data from a real car is presented on the right. 115

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

6.1 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 68

6.2 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 "|". 69

6.3 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. 73

6.4 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. 82

7.1 Overall results for the performance of the method on novelty segmentation. The dimension of the records is presented on the column # *Ch*, as well as the types of signals used and the task in which applied (HACP - Human Activity Segmentation; Act/Rel - Activation/Relaxation of the EMG and Noise detection). The overall measures of Recall and Precision were micro-averaged. 88

7.2 Distance error as a ratio of the time scale (T_s) for the detected TP. 88
7.3 Comparison of performance between the proposed method (*Nova*) and other algorithms. The colors indicate if the other algorithms were better (Green) or worse (Blue) in performance than the *Nova* method for a specific dataset. Time series from *apple*, *bee_waggle_6*, *occupancy* and *run_log* are multivariate. The average results are grouped on the last row. Averages did not considered the gray columns, since these would imply that no change point should be detected, or an error on the signal was present. *T* appears when the method timedout, *M* and *F* when the method failed in compiling 89
7.4 Detected cycles and Duration Error (DE) results of the detection of type 2 events, over the *Human Activity Recognition (HAR)* database 93
7.5 Evaluation of the complexity in the resolution of examples with the *syntactic* approach and with the *classical* approach. Voc - Vocabulary, Lgth - Length, CL - Calculated Length, V - volume, D - difficulty, E - effort. 101
7.6 Results for the top 10 and top 100 sorted gestures classes (G) when using the queries from Figure 7.12. 109

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 [**puttinghuman**]. As reported in *Tankovska et al.*, the wearable devices usage has more than doubled in the interval between 2016 and 2019, reaching 722 million [**tankovska_23_2020**], leading to a large volume of time series data being gathered in all possible scenarios, by monitoring patients in healthcare institutions [**cpd_medical_1**, **cpd_medical_2**, **cpd_medical_3**, **cpd_medical_4**, **dataset6**, **dataset7**], tracking everyday activities of humans [**cpd_har_1**, **cpd_har_2**, **review_1**], recording machines in industrial processes or workers motion while performing their tasks [**antonio**, **sara**].

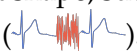
It has never been so easy to gather data about any aspect of our life, work, education, society, or industry. Of course, having relevant information about a subject is beneficial, but the overwhelming amount of data brings tremendous challenges in the ability to save, process, analyze and retrieve interpretable and meaningful information from which we can act upon[**bigdata**]. Ultimately, it becomes even harder to have data well structured and labeled, considering that it is a sensitive and time consuming process, 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 [**roh2019survey**], 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*) [**roh2019survey**].

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 [**democratize**].

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

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

Grammar is generally defined as the book of rules that constitutes the structure of a

language and is modeled by the morphology and syntax [**grammar**]. The first is the structure of words, how these are built or morphed based on context, while the latter consists in organizing words in sequences to form larger linguistic units, such as sentences. Just as a language has morphological and syntax rules that represent its structural information, time series are also organized by a formal structure of ordered subsegments with specific morphological characteristics, organized to build larger segments. Our first topic is related to *unveiling* this structure with methods that can parse it.

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

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

1.3 Biosignals: Context and Relevance in Occupational Health

This thesis was developed in a strong partnership with the *Ergonomy* team from *Volkswagen Autoeuropa*. Therefore, the central domain of the application regarded the analysis of *biosignals* from workers to retrieve meaningful information about their occupational risk status and prevent Work Related Musculoskeletal Disorder (WMSD)s. WMSDs prevail as the most common occupational disease in the European Union. These have a global impact on the well-being of individuals and their quality of life in a range of working sectors [**Irastorza2010**], accounting for the second-largest responsibility to disability worldwide [**Luttmann2003**]. These are especially prevalent as upper limb or neck disorder (with 42% of all WMSD cases reported) [**Seidel2019**] in several industry sectors, such as textile and automotive, where production processes with pre-defined motions and actions have a repetitive/cyclic nature. This has a negative impact on the risk to develop musculoskeletal disorders, with tremendous consequences to both workers and companies, leading to absenteeism, early retirement, and loss of productivity [**Trabalhadores**, **Varandas19**].

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 [**jobrotation1**,

jobrotation2] 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) [**ocra**, **rula**, **eaws**]. 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 [**romero**]. 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 [**Caputo2019**, **Hang19**]. 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 [**sara**]. 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 [**silva_rip**, **cardiovascular_load**, **rythm_cyclic_work**, **rui_varandas**].

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.

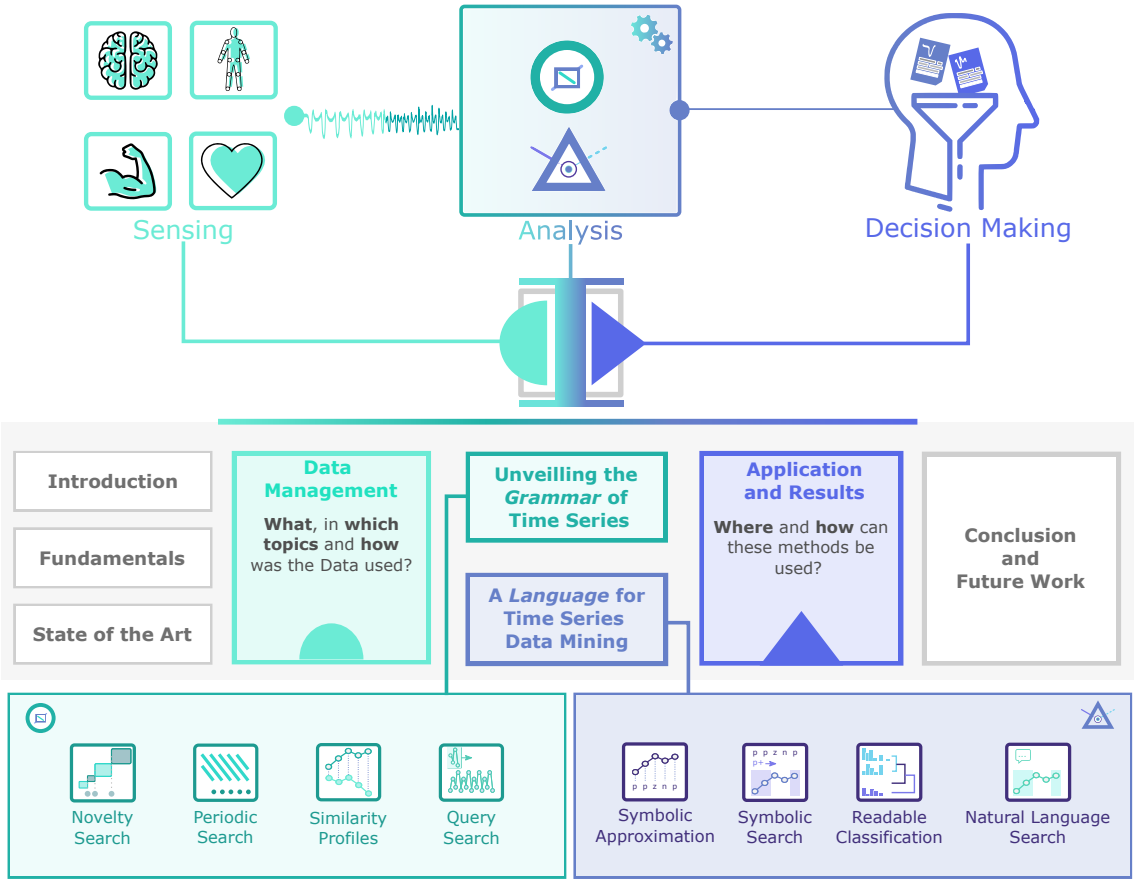


Figure 1.1: General topics contemplated on this thesis and structure of the document. It highlights the 3 layers of involvement related to time series: Sensing, Analysis, and Decision Making, focusing on the Analysis layer, which includes two major topics subdivided into 4 sections each.

1.5 Thesis Structure

This thesis provides a detailed description and explanation of the research work developed during the Ph.D. program. It is organized into nine Chapters, each contributing to telling the story of this thesis. The reader may appreciate Figure 1.1, which illustrates a guideline of the structure of this work, with a short description of each Chapter's topics and content. **Chapter 1** introduced the main motivations, goals, and context for the development of the proposed methods. **Chapter 2** provides the reader with the fundamental definitions and knowledge needed to have a clear picture of what is developed in this work.**Chapter 3** depicts the state-of-the-art works related to what we developed, namely in the topics of segmentation, summarization, pattern/event search and classification.**Chapter 4** describes the data we used, explaining its source for both private data and publicly available data, for which purposes it was used and how it was used in this work.**Chapter 5** explains the algorithm developed for time series structural information retrieval (Unveiling the *Grammar* of Time Series) and provides examples of its usage for novelty segmentation, periodic segmentation, similarity profiles, and query search. *Chapter 6* covers the usage

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

2

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 [**event_def1**, **event_def2**], which state that "*an event is a dynamic phenomenon whose behavior changes enough over time to be considered a qualitatively significant change*" and "characterized by an interval of measurements that differs significantly from some underlying baseline", we consider that an *event* is an instant in time e that indicates the presence of a relevant occurrence in the time series. Multiple *events* segment the time series into several *subsequences* of different lengths. Therefore, *event* detection is often considered time series segmentation or change point detection[**cpd_alan**]. 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 [**fourier**]. The idea behind this concept is that any signal, of any complexity, is a decomposition of a finite number of sine waves. Each wave is represented by a complex number, known as the Fourier coefficient, transforming the signal from the time domain to the frequency domain [**fourier2**]. This transformation allows 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.

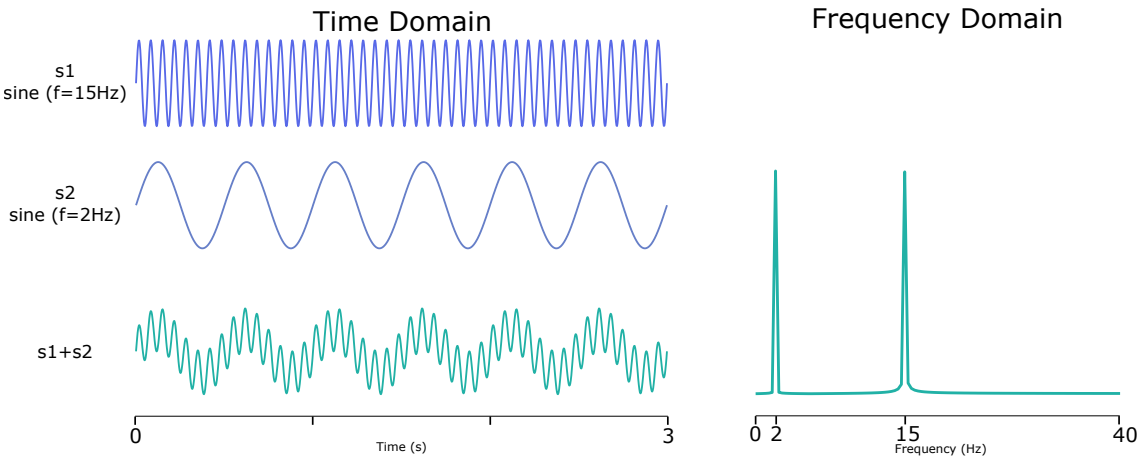


Figure 2.1: DFT of a sum of sine waves.

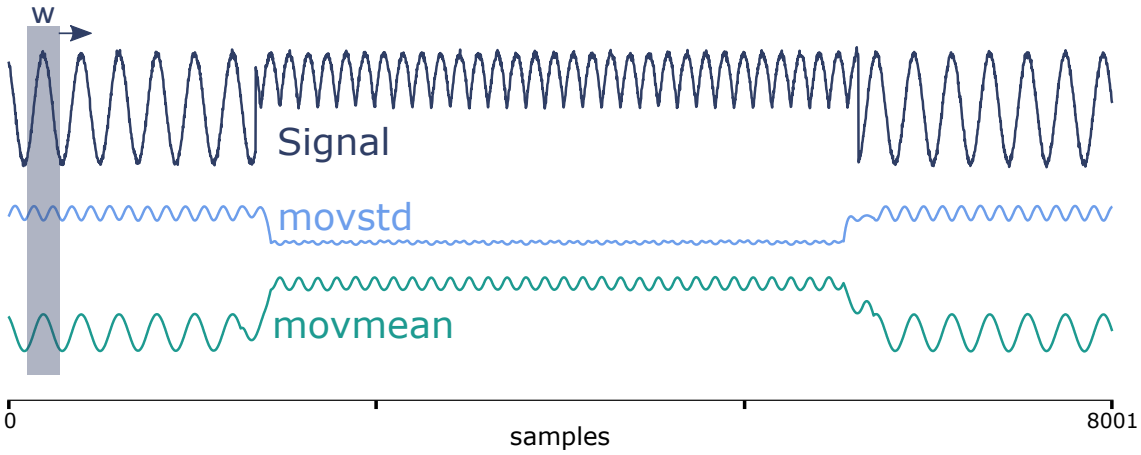


Figure 2.2: Moving window used to extract features with total overlap. The mean and standard deviation are extracted from the signal. The left shows the sine waves, while the right shows the frequency spectrum of the combination of sine waves.

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) [**paa**]. The new representation space will have size $1 < N \leq n$, in which N is a factor of the original size n. The searches to keep the average of the N

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

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

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

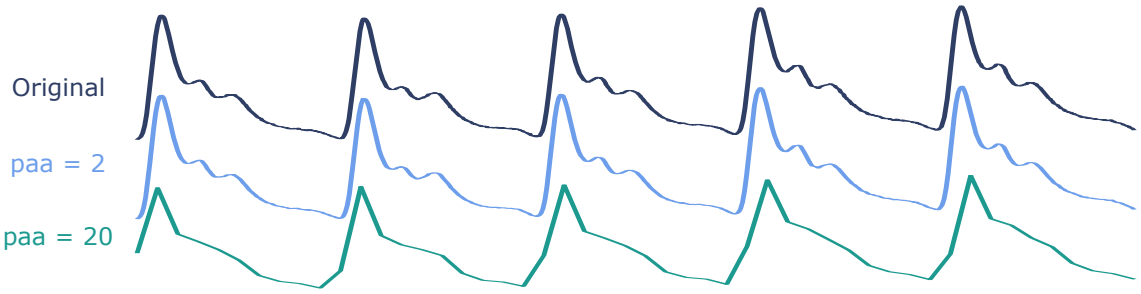


Figure 2.3: PAA representation of a ABP signal, with window sizes of 2 and 20, repsectively. The PAA representation was computed with the PYTS module [**pyts**], based on [**paa**].

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

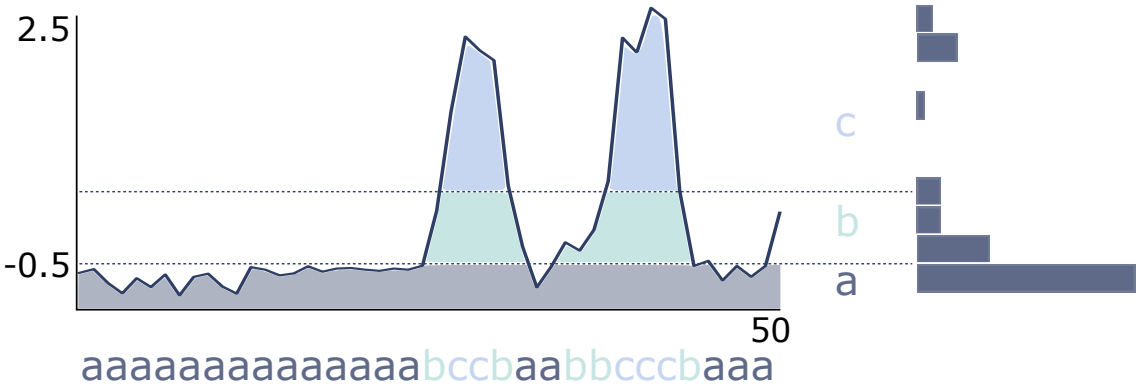


Figure 2.4: SAX representation of a power consumption signal from a Dutch Company, with window bin size of 3. The SAX representation was computed with PYTS based on [**sax**].

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 [**complexity**]:

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

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 $d = (q_i - c_j)^2$ [**dtw**]. 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 [**dtw**].

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

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

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 [**complexity**]:

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

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

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

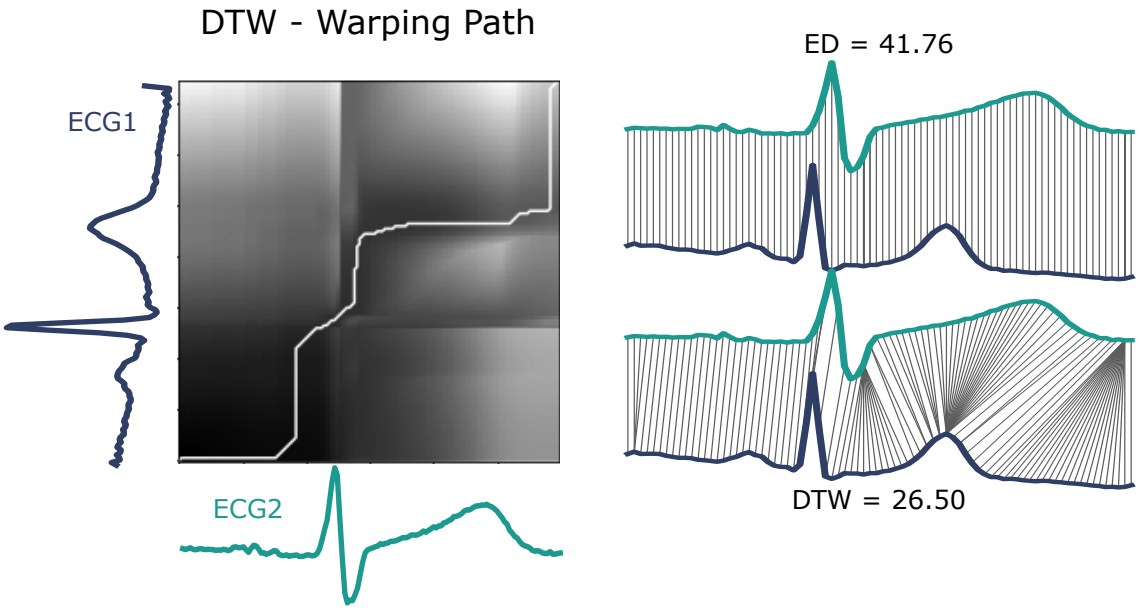


Figure 2.5: DTW and ED distances on two different ECG signals.

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[**complexity**]:

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

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 [**cosine**] (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 [**cosine**].

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* [**eamonn1**]. 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 [**fmp1**]. 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* [**fmp1**, **fmp2**].

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.

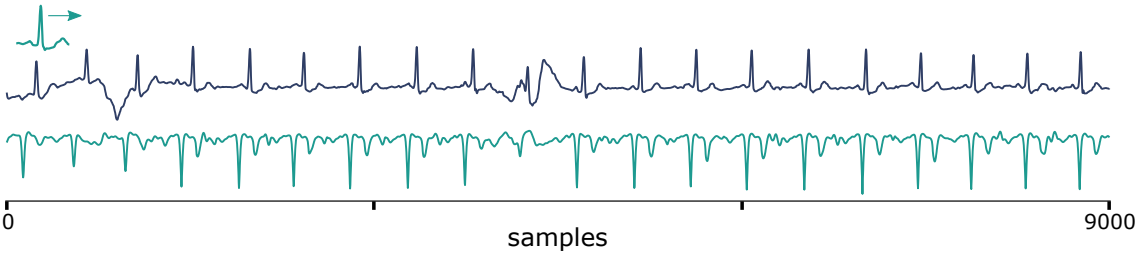


Figure 2.6: Distance profile of a query based search using the z-normalized ED. The template was a PQRS complex of an ECG.

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* [**qetch**] or text [**text_query1**]. 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 [**knn**].

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 [**knn**].

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 SVM classifiers **[scikit-learn]**.

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 [**regex2**]:

MC	Description	Example of Match
*	The preceding item will be matched zero or more times	*eve*nt* → [evnt, event, eveent]
+	The preceding item will be matched one or more times	*su+b* → [sub, suub, suuub]
?	The preceding item is optional and will be matched, at most, once	*team?*→ [tea, team]
.	Matches any character	*s.m* → [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 (here c represents the number of classes).

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

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

$$microP = \frac{\Sigma_{i=0}^{c} TP_i}{\Sigma_{i=0}^{c} TP_i + \Sigma_{i=0}^{c} FP_i} \tag{2.26}$$

$$microR = \frac{\Sigma_{i=0}^{c} TP_i}{\Sigma_{i=0}^{c} TP_i + \Sigma_{i=0}^{c} FN_i} \tag{2.27}$$

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 Complexity

As we are introducing novel ways of performing more expressive query-based searches with regex, we are measuring the legibility and difficulty in generating a query. In the programming field, *Halstead* measures are typically used for this purpose. These measures describe the complexity of the script directly from the source code, based on a set of metrics calculated with the number of distinct operators (oprt) and operands (oprd), and the total number of operators (Toprt) and total number of operands (Toprd). These metrics are the **[Halstead2]**:

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:

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

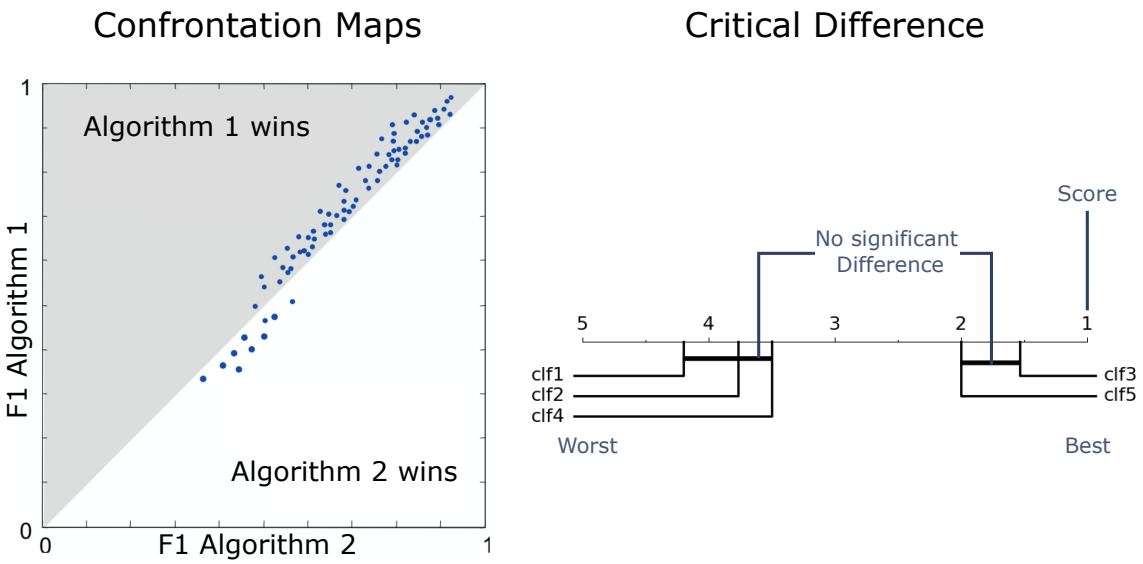


Figure 2.7: On the left are confrontation maps, used to compare classification algorithm performances. On the right is a critical difference map, which shows a statistical difference in the performance of algorithms for general purposes.

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

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 [**critical_dif**].

3

State of the Art

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

3.1 Information Retrieval from Time Series

3.1.1 Segmentation and Event Detection

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

Existing non-supervised methods include older but with state of the art performance in change point detection, such as the Baysian Online method (BOCPD) [**bocpd**], the binary segmentation (BINSEG) method [**binseg**] and the segmentation neighborhoods

(SEGNEIGH) method [**segneigh**]. These methods have been reported successful in several domains [**cpd_alan**]. Still, the BOCPD only achieved good results when parameters were hypertuned, and the BINSEG and SEGNEIGH are not used in multidimensional domains. In addition, these methods are not reported to cope with a multi-time scale change [**cpd_alan**]. An available repository provides an implementation of some of these offline methods [**review_2**], but these lack a visual output that might give the user an intuition over where a change point might be. In addition, most of these methods rely in parameter optimizations, which can make them very brittle to other datasets where these were not optimized [**eamonn1**].

Another method, called FLOSS [eamonn1], relies in searching change points based on the nearest neighbors of subsequences, being very successful in real data domains. It can also handle online problems and multidimensional datasets. As it searches for nearest neighbors, the similarity between segments might be compared and used for summarization.

3.1.2 Self Similarity Matrix

Similarity matrices are not novel in the analysis of time series. These matrices have several representations and denominations, being the most commonly known *recurrence plot*. These plots are a similarity matrix thresholded by a specific value that highlight only similar regions. The representation of these maps was used to analyze time series [**eamonn_dotplots**, **recurrenceplots1**] and search for reoccurring shapes, anomalies or symmetric behaviors [**eamonn_dotplots**]. It was also used for time series classification by inputting the recurrence plot into a neural network (Convolutional Neural Networks) [**recurrenceplots1**, **recurrenceplots2**] or forecasting [**recurrenceplots3**]. The SSM has more information than the recurrence plot (which has been thresholded). Several usages have been explored in the audio domain, namely for segmentation, thumbnailing, periodiciy search or music alignment. For this, the audio signal is transformed on a feature-based representation [**fmp1**, **fmp2**].

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

The proposed method highlights itself for being domain agnostic, work with both uni and multidimensional time series, give events with context by means of the visual information available, but also by the similarity measures in the matrix, that help in associating an event as a change or a periodic segment, and how similar are the segmented

subsequences. It is unsupervised and works offline. It can be extended to work in multi-time scale problems with a special interest in time series summarization. We will demonstrate in this work how this method can bring novelty to the problematic of event detection, with a direct application to labelling and time series summarization.

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

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

Other methods rely on a self-similarity approach, namely [**neuza**], where cyclic information is segmented by searching for minimums, in the convolution of a segment of the signal with itself. The *Matrix Profile (MP)*, which is a method that compares all sub-sequences of a given time series with themselves through an euclidean distance, has also revealed promising results. In the end, it returns the minimum value distance for each segment, highlighting the moments of the time series which are similar within themselves [**Yeh2018**]. Additionally, autocorrelation revealed itself an useful tool, as the

search over maximum values can infer the cyclic nature of the data [**Bauters2014**]. Finally, for anomaly detection in industrial scenarios, an interesting work applies an unsupervised method based on the clustering of time series segments to detect the execution of improper movements [**Varandas19**].

The following work is inspired over an algorithm for the detection of musical structures on audio signals [**foote2000**, **audiolabs1**, **audiolabs2**] by means of a *Self-Similarity Matrix* (SSM). This sort of analysis of self-similarity to collect information about the periodicity has also been performed over video datasets. This type of analysis usually consists on a framework where a Fourier analysis is performed on an SSM to characterize and highlight the periodicity of the data from the video [**Cutler2002**, **Cutler2000**, **Cutler1999**].

3.1.3 Summarization

3.1.4 Text based Query Search

There is a large literature on time series similarity search, see [26] and the references therein. However, in most cases it is assumed that the query comes from a downstream algorithm, not a human. As such, there has been relatively little attention paid to the ability of humans to formulate meaningful queries. In principle one could do "query-by-sketching" and invite the user to draw the pattern she is interested in finding [15,16]. The recent "Qetch" system is a prominent example of this approach [15]. However, there are two possible limitations to such an approach: First, it is not clear that most people have the ability to sketch their query. For example, many people cannot even draw an accurate circle [25]. Secondly, as Figure 1 hinted at, classic distance measures may be too literal and limited in expressiveness to retrieve the desired pattern. As a simple example, suppose that a user wishes to retrieve all highly symmetric patterns. There is simply no way to do this with Euclidean distance or similar distance measures. Other researchers have noted these issues and proposed more flexible queries languages for time series. For example, the SDL (shape definition language) of [11]- allows the user to formulate "blurred" queries. However, we believe that most such systems are not accessible for the typical user. For example, in our proposed system, a 3-point-turn can be successfully queried by noting that the surge axis will exhibit three consecutive "bumps" and formulating the query Surge: [peak peak peak]. In contrast, SDL would require: (Shape triplespeak (width ht1 ht2 ht3) (in width (in order spike (ht1 ht2 ht3) spike (ht1 ht2 ht3)))). Several similar query systems based on regular expressions or SQL-like languages have been proposed, but none seem suitable for general use [17,20]. There have also been a handful of other attempts at natural language querying for time series [6,7]. None of these works seem to have been adopted by practitioners. We feel that this is because they probably suffer from too broad an ambition, proposing completely domain independent search. While domain independence is a worthy ambition (and our eventual research goal), it is clearly challenging. Even the word "spike" can have a different meaning for neuroscientists, economists, epidemiologists, and astronomers. In this work we take advantage of the fact

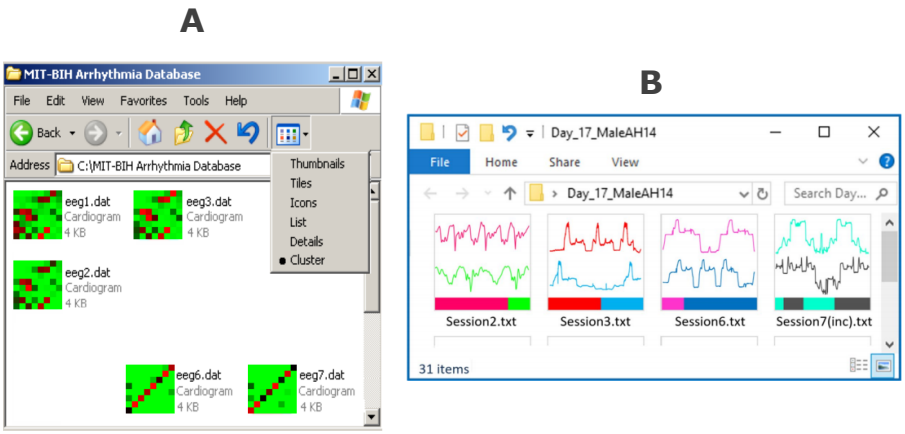


Figure 3.1: Strategies for time series summary found on the literature. These images are taken from the works from [**snippets**, **bitmap**]

that driving is a familiar, even quotidian, activity for most people, and therefore a domain for which most people have strong intuitions for. Moreover, this domain has a near unique property that allows a user to model the behavior they wish to find. We found that, in many cases we could glean intuition as to how a driver's behavior would reveal itself in telemetry by simply "puppeteering" a smartphone equipped with an app that shows its acceleration and gyroscope readings. For example, by modeling a 3-point turn by sliding the smartphone on a desk, we can see that this behavior best revels itself on the surge axis as three consecutive bumps.

3.1.5 Summarization

Very few strategies are found to make compact and meaningful representation of time series. The works that can be highlighted refer to time *snippets* and time series *bitmaps* [**snippets**, **bitmap**]. The first highlights the limitation of current methods in providing a satisfactory solution to time series *summarization*. It proposes a method that is able to segment the k most *representative* sub sequences of a time series, and use these elements as the summary.This strategy answers the segmentation and similarity. Regarding the time series *bitmap* representation, the strategy is able to provide a coded bitmap with information on cluster, anomaly and other regularities on data collection. These bitmaps were used as folder icons, and also answer several of the aforementioned characteristics, such as *similarity* and *events*. An example of both strategies can be seen on Figure 3.1.

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

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

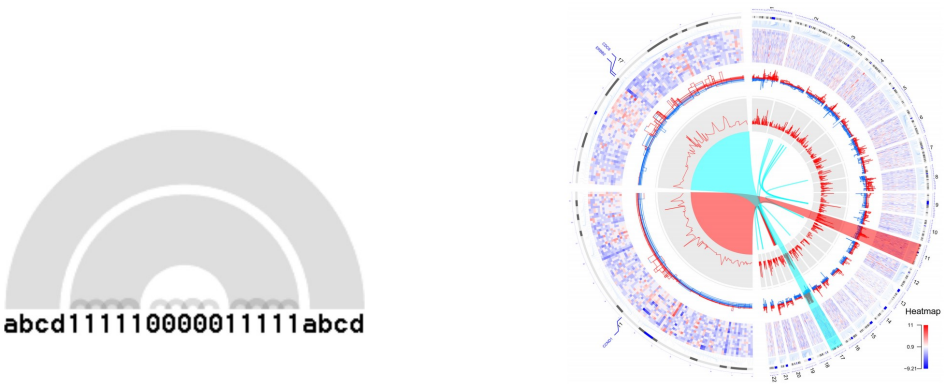
Strategies that are typically used to present information in a compact way are found in several domains. In text analysis, for instance, the relationship between repeating sequences is illustrated with arc diagrams [**bitmap**, **arcplots**]. These show where repeating sequences occur in a very concise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

One domain that has a particular relevance in data visualization is genomics. Graphical genome maps are found to concatenate a significant amount of information in a very compact way. Genome features and sequence characteristics are assessed with this visual strategy. An example can be found on Figure 3.2b. This visualization strategy can provide increasing circular layers of information. Although we are used to look at time series from left to right, a circular representation can have benefits to concatenate the information we want to include.

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

3.1.6 Classification

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



(a) (b)

Figure 3.2: A - Diagram for string association. This image is taken from the works from [**arcplots**]; B - Circular plot by OmicCircos. Several layers (Circular tracks) identify genome position, expression heatmaps, correlation between expression and CNV, among other features. The image is taken from the works from *Ying Hu, et al.* [**genomics**].

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

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

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

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

All the mentioned methods are a reference in TSC tasks with innovative concepts that merge ideas from the text-mining domain into TSC domain. One of the advantages

of structure-based methods that rely in a dictionary-based concept is to use the words extracted as an interpretable model to differentiate time series. The histogram of words generated gives the user an understanding of which patterns better represent the time series and give an intuition over patterns that differ between classes of time series. This provides a feedback and explanation over why a class is different than the other. However, dictionaries can be confusing, and the words generated are not intuitively associated with the patterns these represent in the time series. One method that went beyond the previously mentioned methods in that aspect is the SAX-Vector Space Model (SAX-VSM). This method used a weighted word vector representation of the time series and showed which are the relevant words for the classification process and what patterns these represent in the time series, demonstrating that the classification process can be interpretable by measuring the importance of the patterns found for each class of signals [**sax_vsm**].

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

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

3.1.7 Search by Query

There is a large literature on time series similarity search, see [26] and the references therein. However, in most cases it is assumed that the query comes from a downstream algorithm, not a human. As such, there has been relatively little attention paid to the ability of humans to formulate meaningful queries. In principle one could do "query-by-sketching" and invite the user to draw the pattern she is interested in finding [15,16]. The recent "Qetch" system is a prominent example of this approach [15]. However, there are two possible limitations to such an approach: First, it is not clear that most people have the ability to sketch their query. For example, many people cannot even draw an accurate circle [25]. Secondly, as Figure 1 hinted at, classic distance measures may be too literal and limited in expressiveness to retrieve the desired pattern. As a simple example, suppose that a user wishes to retrieve all highly symmetric patterns. There is simply no way to do this with Euclidean distance or similar distance measures. Other researchers have noted these issues and proposed more flexible queries languages for time series. For example, the SDL (shape definition language) of [11]- allows the user to formulate "blurred" queries.

However, we believe that most such systems are not accessible for the typical user. For example, in our proposed system, a 3-point-turn can be successfully queried by noting that the surge axis will exhibit three consecutive "bumps" and formulating the query Surge: [peak peak peak]. In contrast, SDL would require: (Shape triplespeak (width ht1 ht2 ht3) (in width (in order spike (ht1 ht2 ht3) spike (ht1 ht2 ht3)))). Several similar query systems based on regular expressions or SQL-like languages have been proposed, but none seem suitable for general use [17,20]. There have also been a handful of other attempts at natural language querying for time series [6,7]. None of these works seem to have been adopted by practitioners. We feel that this is because they probably suffer from too broad an ambition, proposing completely domain independent search. While domain independence is a worthy ambition (and our eventual research goal), it is clearly challenging. Even the word "spike" can have a different meaning for neuroscientists, economists, epidemiologists, and astronomers. In this work we take advantage of the fact that driving is a familiar, even quotidian, activity for most people, and therefore a domain for which most people have strong intuitions for. Moreover, this domain has a near unique property that allows a user to model the behavior they wish to find. We found that, in many cases we could glean intuition as to how a driver's behavior would reveal itself in telemetry by simply "puppeteering" a smartphone equipped with an app that shows its acceleration and gyroscope readings. For example, by modeling a 3-point turn by sliding the smartphone on a desk, we can see that this behavior best revels itself on the surge axis as three consecutive bumps.

3.2 Symbolic Representation

3.3 Dictionnary-based Classification

3.4 Search by Query

4

Data Description and Management

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

4.0.1 Dataset 1 - UCR Benchmark

The University California Riverside (UCR) Time Series Archive was introduced in 2002 and is one of the most used benchmarks for time series data mining tasks, specially for classification. The datasets are diverse in terms of data type, data domain, difficulty, number of classes and dimensionality[**ucr**]. Several *python* distributions make it available to download. In this thesis, all UCR archive datasets from the *pyts* distribution were used for classification tasks. These represent 107 datasets from 18 different data types (audio, devices, ECG, Electrocularogram (EOG), EEG, HAR, etc...), which also are from many different domains, such as medical, financial, motion, entomology, etc...[**ucr**]. The list of datasets can be found on the following link [**ucr_site**].
Purpose: This dataset was used in the context of classification to validate Human Readable Time Series (HeaRTS).

4.0.2 Dataset 2 - Human Activity Recognition

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

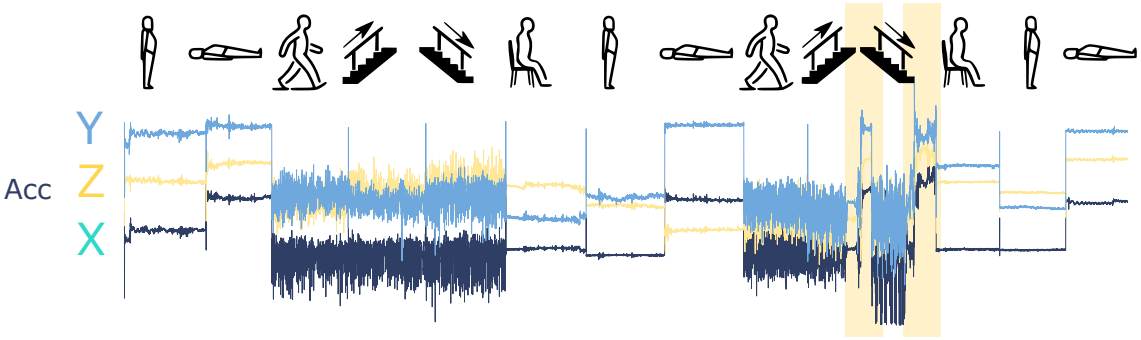


Figure 4.1: Example of the signal for this dataset. It shows the 3 axis of the accelerometer signal and the corresponding labels in each section [**dataset2**, **dataset2_2**]. Yellow areas indicate moments where there was a change on the signal not related with the labeled activity.

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

4.0.3 UCI Machine Learning Repository

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

From this archive, we focused in time series data for the validation of the proposed methods. More specifically, this repository contains time series that can be used to validate the proposed novelty and cyclic segmentation. We searched for multidimensional data that could be used for time series segmentation and had labeled data.

The following datasets were used:

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

This dataset was gathered from an experiment on 30 volunteers. Each subject was wearing a smartphone on the waist while performing several activities: *(1) Walking, (2) Walking Upstairs, (3) Walking Downstairs, (4) Sitting, (5) Standing and (6) Laying*. The activities were performed for approximately 60 seconds. The device recorded the internal ACC and ?? data at a constant rate of 50 Hz. Each activity has been categorized and labeled on the acquired data [**dataset2**, **dataset2_2**].

Purpose: This dataset was used in the context of novelty segmentation [**dataset2**, **dataset2_2**].

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

The dataset was built in the context of human activity recognition experiments. These were carried out with a group of 30 volunteers that performed a protocol with six basic activities: three static postures (standing, sitting, lying) and three dynamic activities

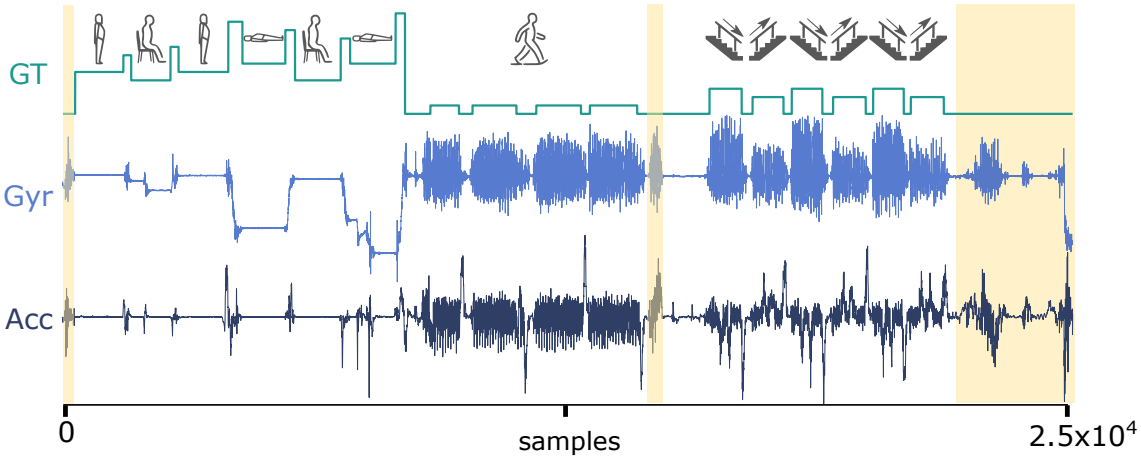


Figure 4.2: Example of the signal for this dataset. It shows one axis for the ACC and ?? signals and the corresponding labels in each section [**dataset3**]. In this datasets, labels also highlight posture transitions, such as *Standing to Sitting*. Yellow areas indicate moments where there is activity but the signal was not labeled.

(walking, walking downstairs and walking upstairs). Additionally, the experiment also included postural transitions that occurred between the static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. The data was collected with a smartphone (Samsung Galaxy S II) mounted on the waist of each subject. The data from a 3-axis accelerometer and 3-axis gyroscope was gathered at a constant rate of 50Hz. The experiments were video-recorded to label the data manually [**dataset3**].
Purpose: This dataset was used in the context of novelty segmentation [**dataset3**].

Dataset 5 - Wireless Sensor Data Mining (WISDM) Smarphone and Smartwatch Activity Biometrics Dataset
The raw data from the accelerometer and gyroscope sensors is collected from a smartphone and smartwatch at a rate of 20Hz. This experiment was conducted on 51 participants as they performed 18 activities, each for a duration of 3 minutes. Each sample of the data was labelled based on the activity it corresponds to. The activities are diverse and include dribbling, eating, jogging, sitting, walking on stairs, standing, walking, among others [**dataset4**].
Purpose: This dataset was used in the context of novelty segmentation in complex real-life scenarios.

Dataset 6 - EMG Data for Gestures
This dataset has EMG signals for recording patterns, by using a MYO Thalmic bracelet worn on a user's forearm. The bracelet is equipped with eight sensors equally spaced around the forearm that simultaneously acquire electromyographic signals. The dataset has raw EMG data from 36 subjects while they performed series of static hand gestures. The subject performs two series, each of which consists of six basic gestures. Each gesture was performed for 3 seconds with a pause of 3 seconds between gestures. The data was collected with a fixed sampling frequency of 200 Hz [**dataset5**].

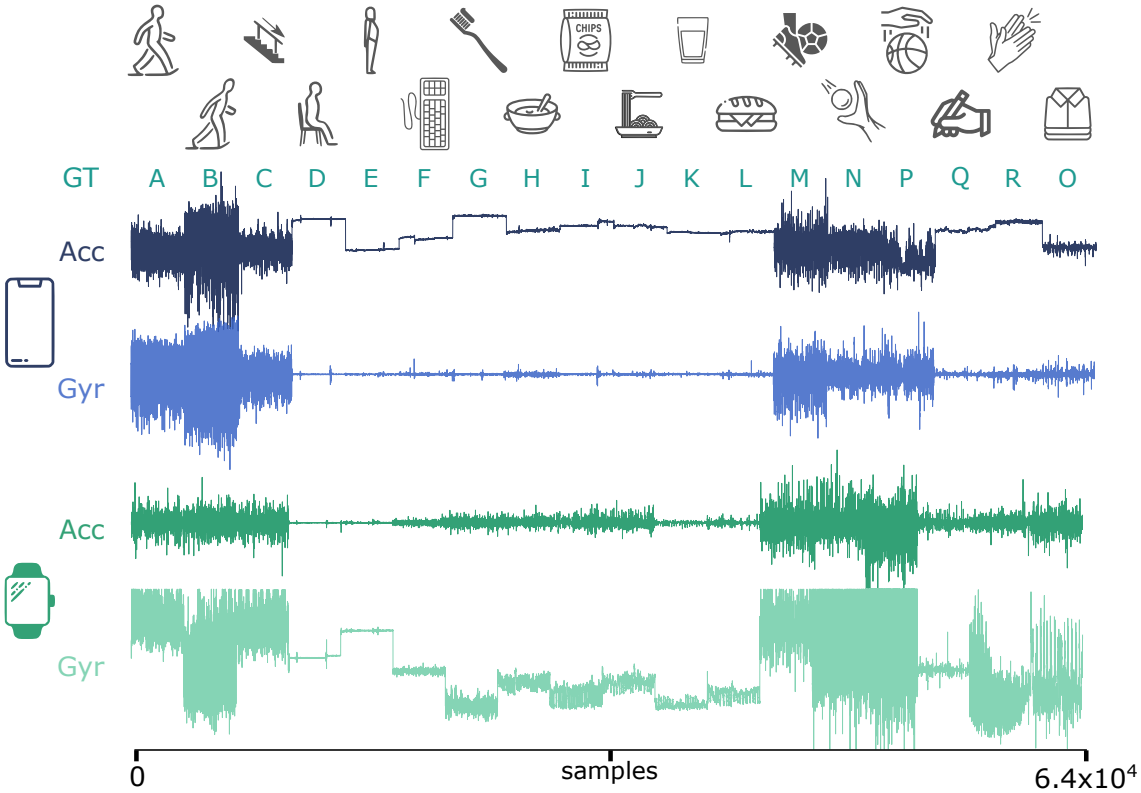


Figure 4.3: Example of the signal for this dataset. It shows one axis of each type of data acquired, for both smartphone and smartwatch. The activities are highlighted as well and labeled as: A - walking, B - running, C - walking on stairs, D - sitting, E - standing, F - typing, G - brushing, H - eating soup, I - eating chips, J - eating pasta, K - drinking, L - eating a sandwich, M - kicking, N - catching a ball, P - dribbling, Q - writing, R - clapping, O - iron [**dataset4**].

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

As indicated on Figure 4.4, the ground truth would have considered as events the absence of activity (which the original dataset uses because it was designed for a classification problem). In this case, we did not considered these events in our ground truth for this dataset for the problem of segmentation.

4.0.4 Physionet

Physionet is a platform where public datasets from the medical domain (mostly physiological signals) are available. In this thesis, we used several datasets to test the developed methods.

Dataset 7 - MIT-BIH Noise Stress Test Database

The dataset comprehends 12 half-hour ECG recordings and 3 half-hour recordings of noise typical in ambulatory ECG recordings. The noise recordings were made using

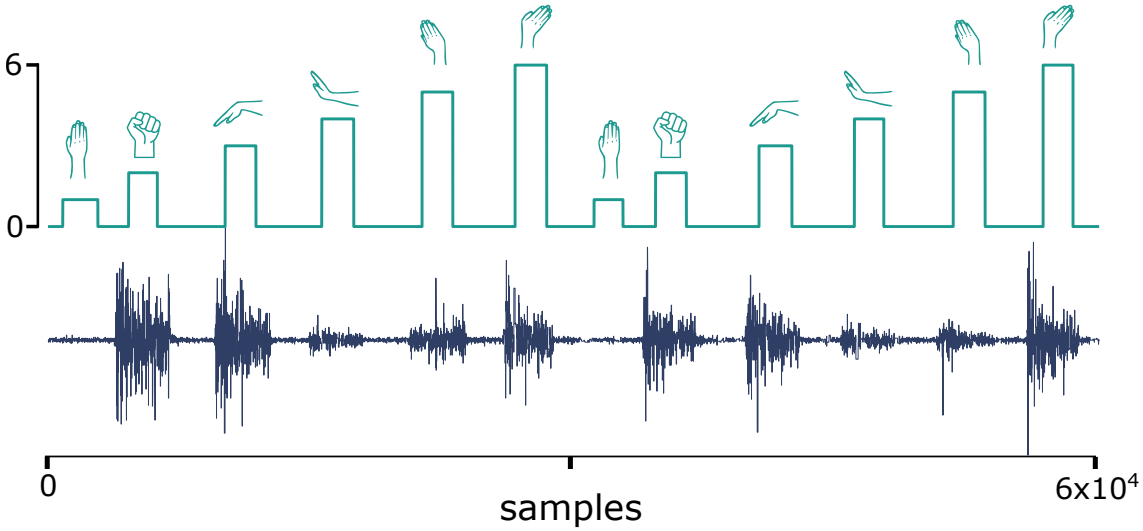


Figure 4.4: Example of the EMG data and the corresponding hand postures.

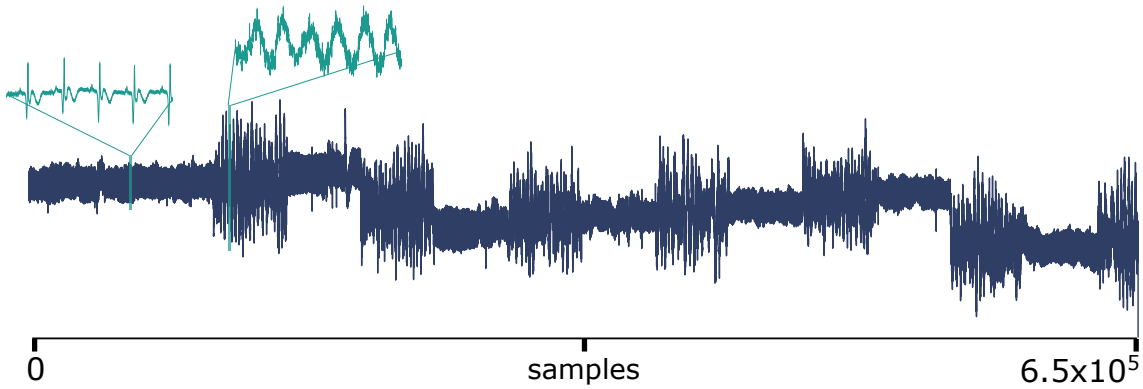


Figure 4.5: Example of an ECG record contaminated with noise. Two sections are highlighted showing the clean and contaminated area of the signal. [**dataset7**]

physically active volunteers and standard ECG recorders, leads, and electrodes. The three noise records were assembled from the recordings by selecting intervals that contained predominantly baseline wander (in record 'bw'), muscle (EMG) artifact (in record 'ma'), and electrode motion artifact (in record 'em'). Two clean ECG signals were selected and noise was added with different signal-to-noise ratios (SNR) [**dataset6**, **PhysioNet**].

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

Dataset 8 - Motion Artifacted Contaminated ECG

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

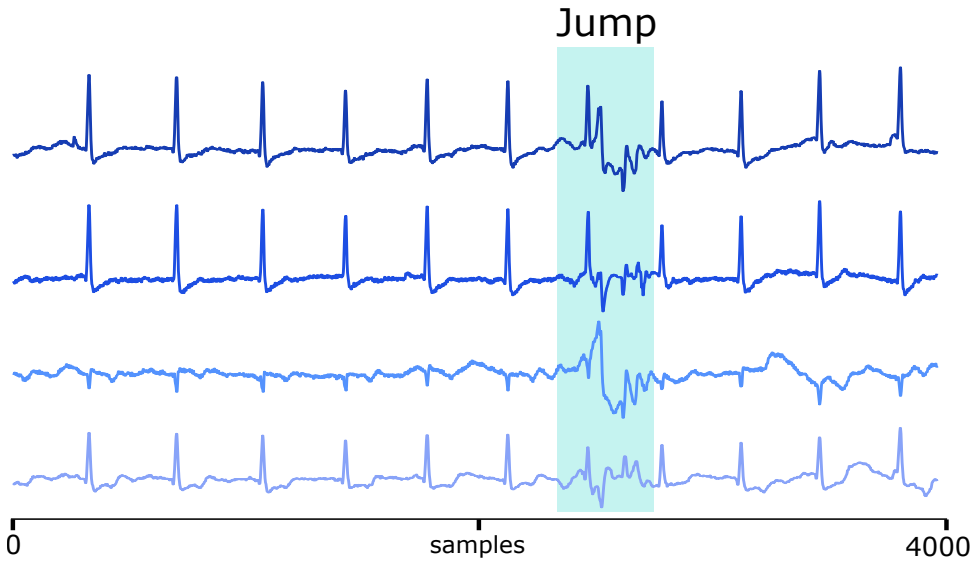


Figure 4.6: Example of an ECG signal contaminated with motion artifacts. The artifact is visible due to a jump from the subject.

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

Dataset 9 - St Petersburg INCART 12-lead Arrhythmia Database

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

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

4.0.5 Dataset 10 - Alan Turing CPD Benchmark

The Alan Turing Change Point Detection Benchmark is a recent dataset used to have a standard reference for change point segmentation tasks. The benchmark was created in 2020 and provides 42 datasets, being 42 univariate time series and 4 multi dimensional time series. The dataset comprises several time series from real-world data. It was built from a project of the Alan Turing Insitute to have a repository for the evaluation of change point detection algorithms. The available benchmark page was used to get access to the datasets, and run the performance metrics developed. This way, the performance of the proposed method is compare with the performance of several existing methods [**cpd_alan**].

Purpose: This dataset has ground-truth events for each time series. The performance

of several existing methods is available and used to compare with the performance of the proposed method on the same time series.

4.0.6 Dataset 11 - HCILab Bhevioral Driving Dataset

HCI Lab Driving Dataset

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

4.0.7 Dataset 12 - CSL-Share Dataset

For this thesis, there were allowed access to inertial acquisitions made in the context of human activity recognition. The database used was obtained on the scope of the *Arthrokinemat* project whose main objective was the development of a learning adaptive sensor-based measurement system to prevent osteoarthritis[**arthrokinemat**]. More specifically, the recording was made for the work of [**Liu2019**], which introduces a human activity recognition system able to recognize between a list of several daily activities.

A set of multiple Biosensors were used, with various internal characteristics. Beginning with two *8 channel PLUX hubs* a device which allows for the wireless acquisition of biosensors via Bluetooth, with them being part of the *biosignals plux Research Kits*. From both *plux hubs* there were used two 3-axial ACC sensors, 4 sets of EMG sensors and an electrogoniometer. Adding to these sensors there were also used 4 other types of biosensors: one airborne microphone, one piezoelectric microphone, two 3-axial gyroscopes and one force sensor. This setup was used to perform 18 different activities, namely *Sit*, *Sit-to-Stand*, *Stand*, *Stand-to-Sit*, *Stair-Up*, *Stair-Down*, *Walk*, *Curve-Left-Step*, *Curve-Left-Spin*, *Curve-Right-Step*, *Curve-Right-Spin*, *Run*, *V-Cut-Left*, *V-Cut-Right*, *Lateral-Shuffle-Left*, *Lateral-Shuffle-Right*, *Jump-One-Leg*, *Jump-Two-Leg*. As a disclaimer, **lateral-shuffle-left/right** is a motion usually done in sports that describes the subject's lateral movement of the left/right foot, with the other foot following along and continuing the shuffling in the same direction. **V-cut-left/right** means that the subject changes his direction by 90°at jogging speed [**dataset_hui**].
Purpose: This database was used to study how the algorithm could be used to detect periodic events with different levels of motion cycle complexity.

4.0.8 Dataset 13 - Industrial Job Dataset

During the period of this thesis, a private dataset was acquired at Volkswagen Autoeuropa in the context of this thesis and a master thesis from [**santos2019**]. The purpose was to test a wearable system capable of calculating the ergonomic risk through direct measures from inertial sensors. The technological setting was provided by a sensing framework named Internet of Things in Package (IoTiP), designed and provided by Fraunhofer AICOS. IoTiP is a system that intends to combine hardware, firmware and software components to promote the field "Internet of Things"[**FraunhoferAICOS**]. For this work, the technology setting consisted on 4 9-Domain of Freedom (DoF) IMU sensors (composed internally by a triaxial ACC, triaxial GYR and triaxial Magnetometer (MAG)) and an Android wireless communication system. The last one was made through an application called Recorder, also developed by Fraunhofer AICOS.[**santos2019**].

The acquisition was made in a *Volkswagen Industrial* assembly line where 12 manufacturing workers performed their work tasks while having attached 4 IMU sensors in their upper body segments. During the acquisition, the subjects performed various tasks in multiple workstations. Relevantly to this thesis, the database comprehends three different workstations from *Bodyshop assembly line*, a section where cars' doors were assembled: 1) Liftgate workstation, where back doors are mounted; 2) Fender workstation, involving front door tasks and 3) Doors workstation, which demanded tasks on the front doors and in the cars' hood [**santos2019**]. The acquisitions made involved a total of 6 operators with each one performing at least 2 different workstations. The various acquisitions were simultaneously filmed, and to synchronize the ground manual annotations of the data, in the beginning and end of the acquisition the subjects were asked to stay, firstly, in a neutral anatomic position and then perform a T pose (calibration position). There were also registered some details regarding their anthropomtetric characteristics.

The mentioned study was centered on the ergonomic assessment of the dominant arm. For this reason, the IMUs were attached in: 1) the posterior side of the hand, 2) posterior side of the forearm and 3) posterior side of the arm and a final one 4) placed in the anterior side of the thorax area. All of the devices were attached with elastic bands, in such a way that all had their Y-axis pointed up while in a neutral anatomical position. Additionally, a Smartphone was attached to the trunk of the workers as well, working as an additional IMU, from which the position of the arm in regards to the body posture was calculated. Each one of these 4 IMU devices had incorporated within them 3 triaxial sensors (ACC, GYR, MAG).

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

Some important notes regarding exceptions found in sensors:

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

and the MAG was acting in an erratic manner.

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

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

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

4.0.9 Notes for Ground Truth Annotations on Novelty Segmentation

Regarding the proposed method for novelty segmentation, the datasets used were mostly designed for classification tasks. Therefore, these were labeled with categorical values for each sample of the time series. As the problem of novelty segmentation requires the detection of specific samples of the signal, we adapted the labels of the datasets to fit the purpose of segmentation. This was made by only keeping the transitions between categories of labels, for example, in Figure 4.2, the ground-truth (GT) is categorical. For evaluation purposes, the ground-truth was converted to a binary signal, where categorical transitions were valued to **1**, to keep only the position of the change, and not the category. This is valid for segmentation purposes only.

5

Unveiling the *Grammar* of Time Series

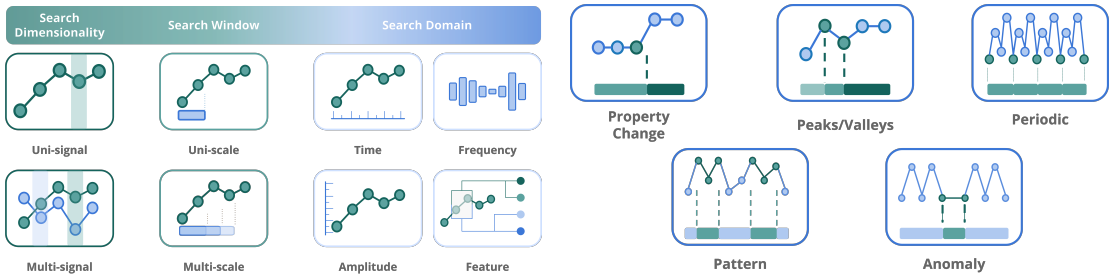
In this chapter is described the process to unveil the structure of a time series. The next sections will start by giving an introduction to the variables that have to be retrieved and demonstrates how to perform them. The main method is inspired by the audio-processing domain for *music structure analysis*, namely segmentation and audio-thumbnailing. While having already been extensively studied in this domain [**fmp1**, **audiolabs1**, **audiolabs2**, **cpd_audio**], this knowledge has not yet been extended to other types of *time series*, which could greatly benefit from it [**muller_music_health**].

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.

5.1 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.



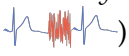
(a) (b)

Figure 5.1: (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.

5.1.1 Search Dimension

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

5.1.2 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 ).

5.1.3 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

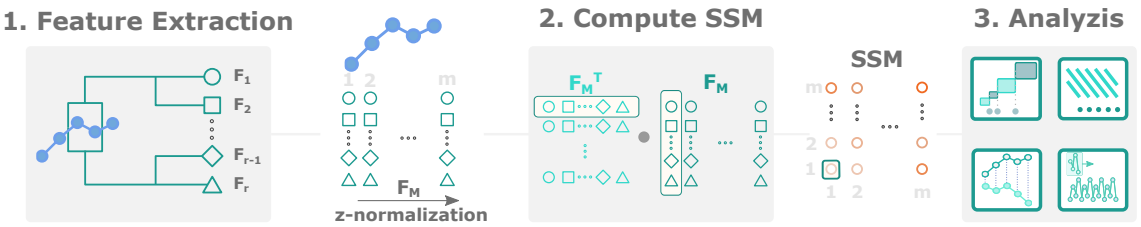


Figure 5.2: Main process to reach the SSM. The information needed to calculate the SSM is the record and the input parameters: the window size (w) and the overlapping percentage (o). The first stage involves the feature extraction process, based on w and o values. Features are extracted on each subsequence ($sT_1, sT_2..., sT_N$), 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.

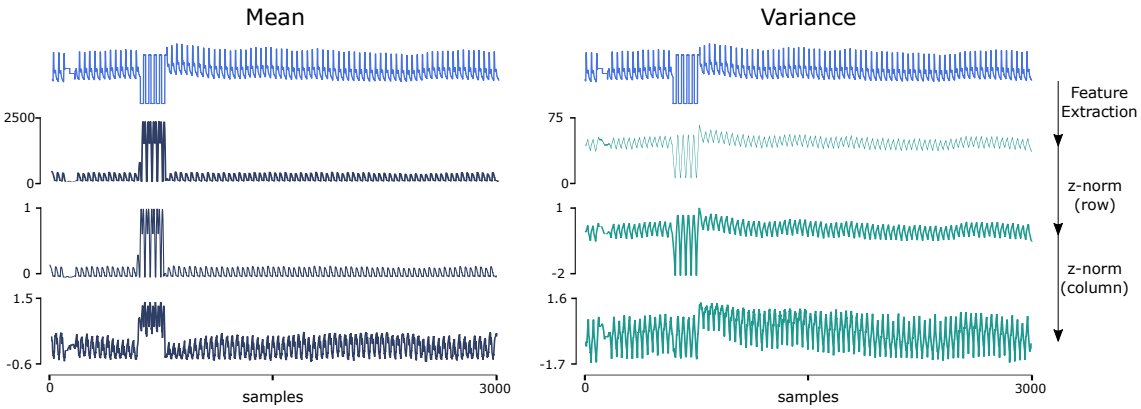
for changes in the mean, standard deviation, mean frequency or other property. By characterizing the signal into the feature space, we can explore changes in all feature representations. Additionally, an event should separate two different behaviors. The notion of *difference* in time series can be associated 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.

5.2 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 5.2 summarizes the steps involved in calculating the SSM.



5.2.1 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* [**barandas_tsfel_2020**] 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 5.2.

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 **??**.

5.2.2 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

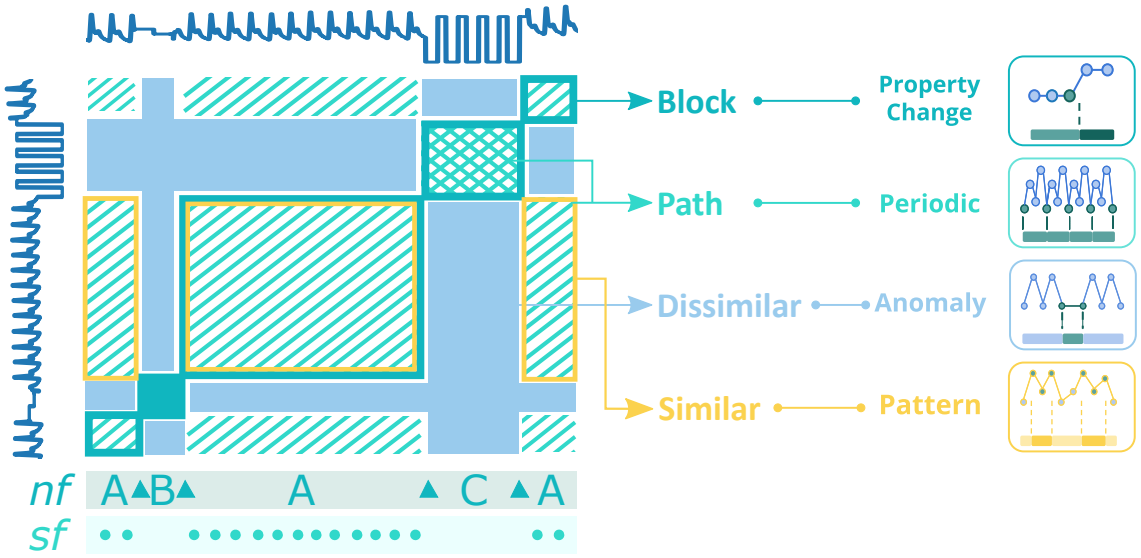


Figure 5.3: Description of informative structures of the SSM of a ABP signal. We show a simplified view with highlights on the relevant structures. The record has 4 main structures: A - homogeneous segment, which corresponds to the ABP periodic signal; B - a homogeneous segment of missed data; C - homogeneous 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

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:

$$SSM = F_M^T F_M \tag{5.1}$$

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 [**audiolabs1**, **audiolabs2**]. 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 5.3, the main structures are illustrated and highlighted in an example of an *SSM* [**audiolabs1**] computed from an ABP signal. As mentioned in Chapter **??**, 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.

Figure 5.4: Information retrieval topics explained in this section.

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 5.3, 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 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.

5.3 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 5.4, 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*.

5.3.1 Novelty Search

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

As shown in Figure 5.5, 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

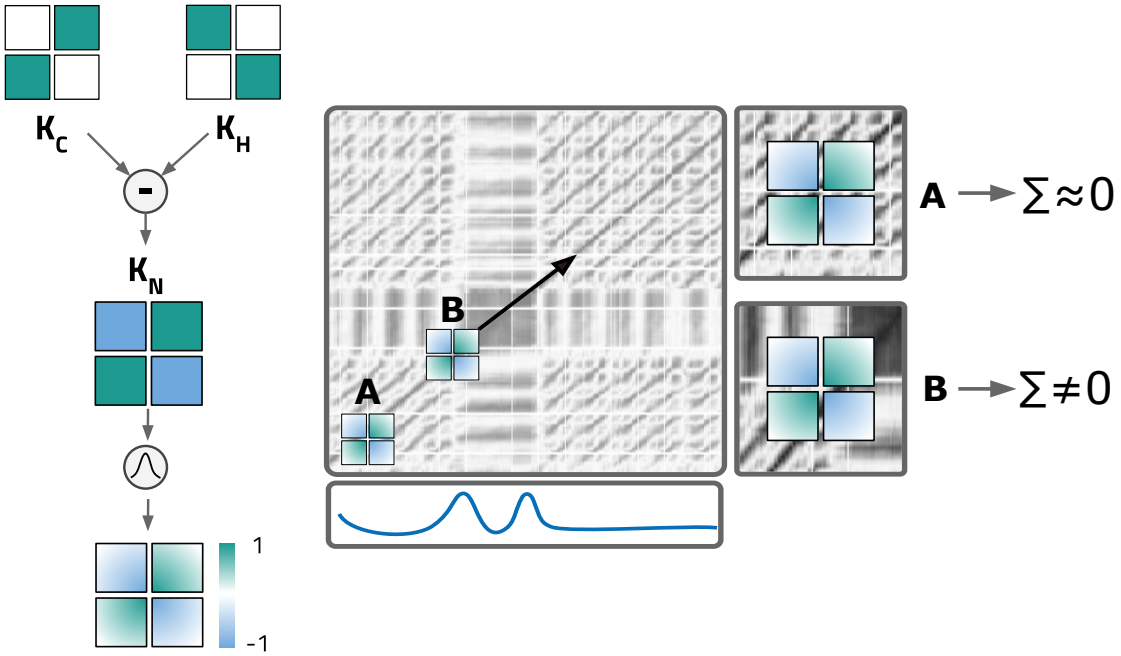


Figure 5.5: (left) Description of the matrix (kernel) used to compute the *novelty function*. The checkerboard pattern is achieved by combining kernel K_H - measure of homogeneity; and K_C - measure of cross-similarity. The resulting kernel (K_N) is combined with a Gaussian function to generate K_G. The Figure is based on the works of *Mueller et al.* [**fmp1**, **fmp2**]; (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 [**Dannenberg2008**, **fmp1**, **fmp2**].

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* [**Dannenberg2008**, **fmp1**, **fmp2**].

As shown in Figure 5.5 (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 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 [**Dannenberg2008**, **fmp1**, **fmp2**].

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 [**fmp1**, **fmp2**]:

$$K_N(i, j) = \text{sign}(a_i) \cdot \text{sign}(b_j) \tag{5.2}$$

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 [**fmp1**, **fmp2**]:

$$\phi(p, u) = \exp(-\frac{1}{2L\sigma^2}(p^2 + u^2)) \tag{5.3}$$

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:

$$K_G = \phi \cdot K_N \tag{5.4}$$

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

$$nova(m) = \sum_{i,j=0}^{2L+1} K_G(a_i, b_j) SSM(m + a_i, m + b_j) \tag{5.5}$$

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.

5.3.2 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:

$$sf(x) = \sum_{i=0}^{m} SSM_{ix} \tag{5.6}$$

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.

5.3.3 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 5.2. 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:

$$P_s(c) = \frac{\Sigma_{i=0}^{l} SSM(i, c)}{l} \tag{5.7}$$

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 5.3 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.

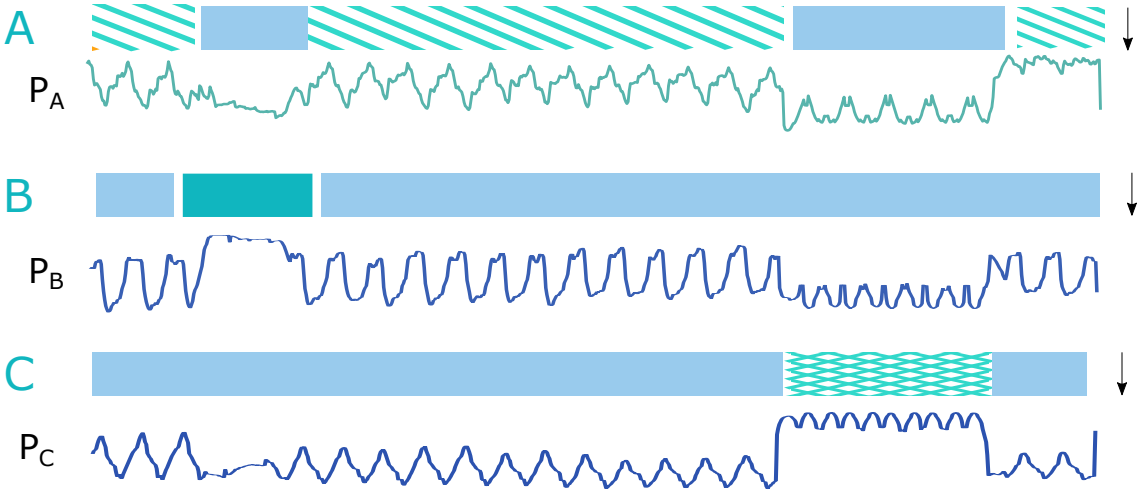


Figure 5.6: Profiles computed for each segment of the example signal used in Figure 5.3.

5.3.4 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

$$D(i) = \sum_{i=0}^{i=m} \sqrt{(SSM(i) - SSM_t)^2} \tag{5.8}$$

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.

5.4 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.

5.4.1 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 7.13.

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.

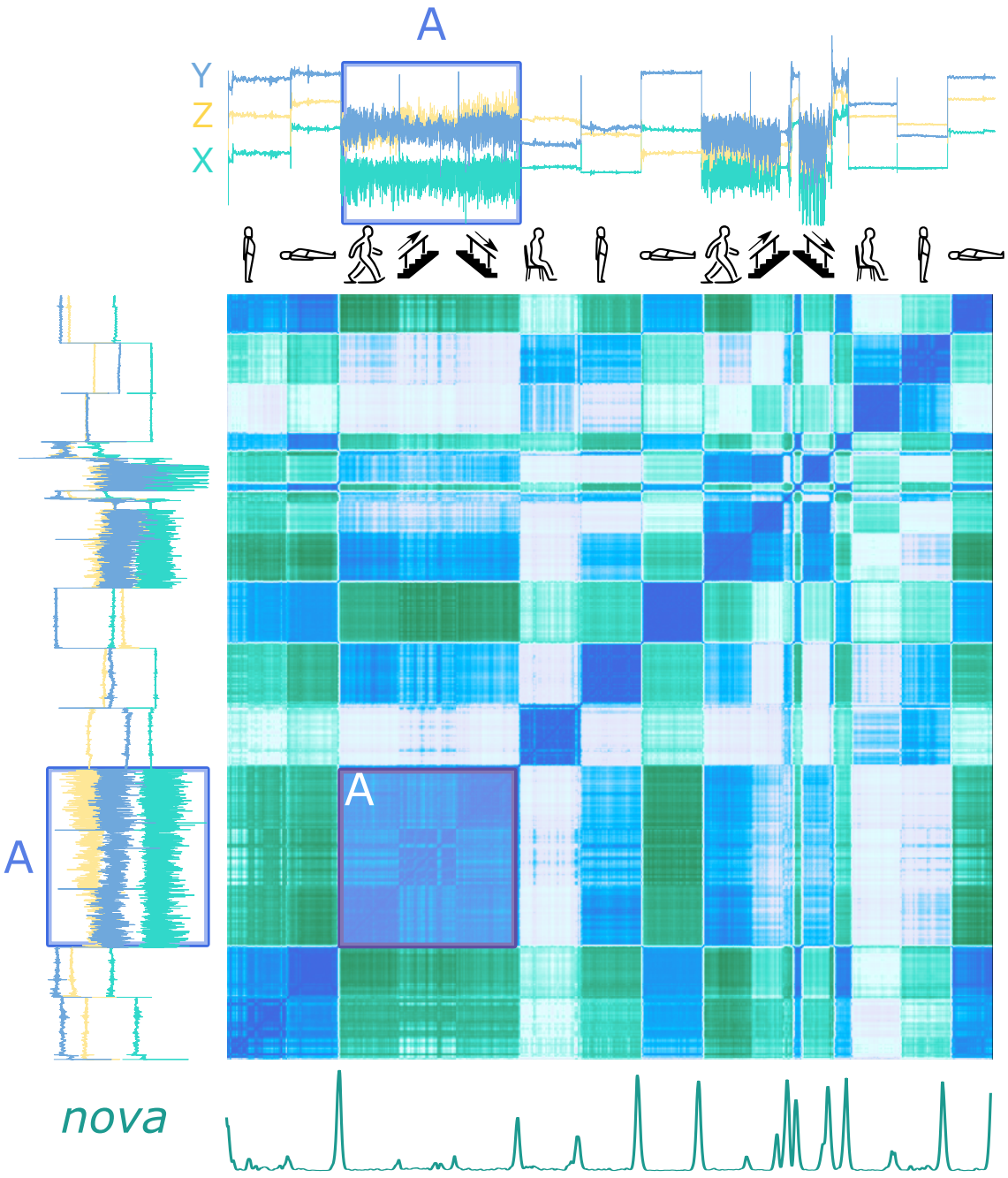


Figure 5.7: Change point event detection strategy applied on the SSM to search for change point events. The sequence of activities is presented as follows: $Sitting \rightarrow Laying \rightarrow Walking \rightarrow Upstairs \rightarrow Downstairs \rightarrow Sitting \rightarrow Standing \rightarrow Laying \rightarrow Walking \rightarrow Upstairs$. The input variables used are $time_{scale}$=250 samples, $kernel_{size}$=45 samples, overlap=95%

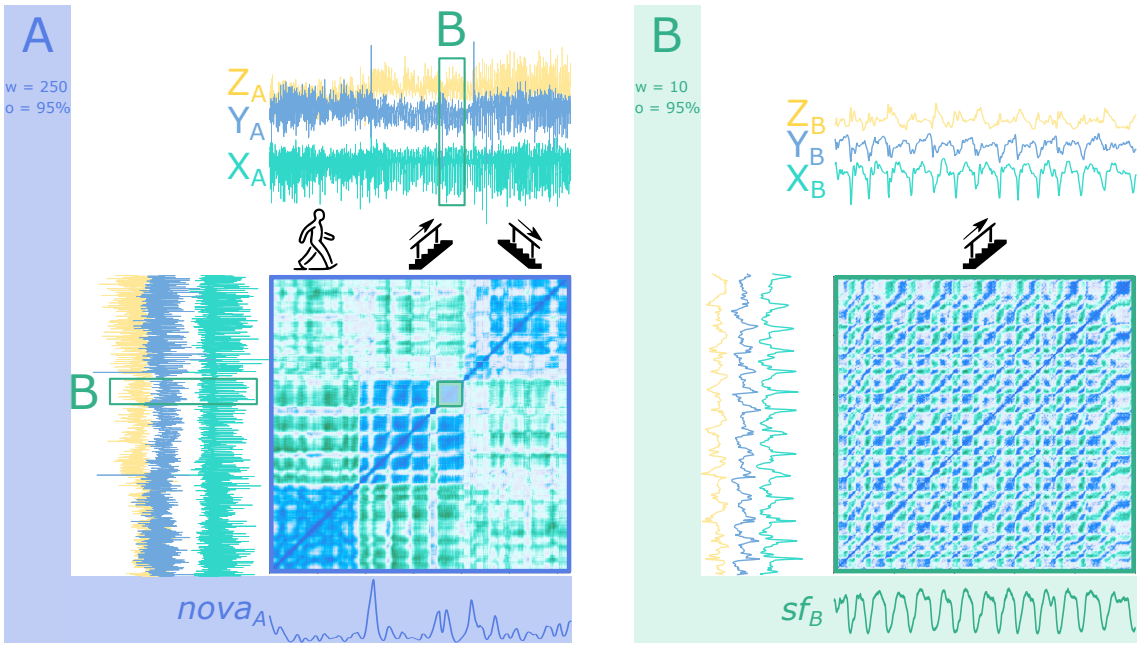


Figure 5.8: ABP signal change point detection. The parameters used were a size of 5000 samples, with an overlap of 75% and a kernel size of 25 samples.

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*).

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 5.8 (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 7.13 is too large. If features are

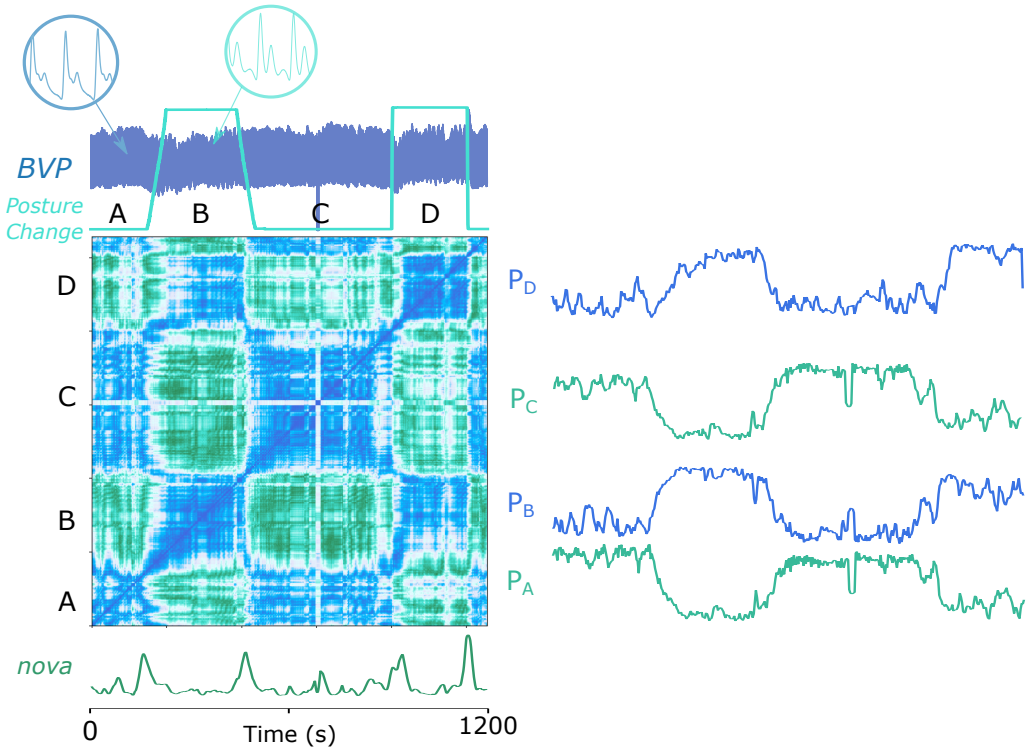


Figure 5.9: (a) ABP signal novelty search. The parameters used were a window size of 5000 samples, with an overlap of 95% and a kernel size of 200 samples.

extracted with smaller window size, closer to the walking period, the *paths* that indicate the recurrence of shapes are visible. Figure 5.8 (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 ($s\,f_B$).

5.4.2 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 [**tilt**, **PhysioNet**]. Figure 5.9 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

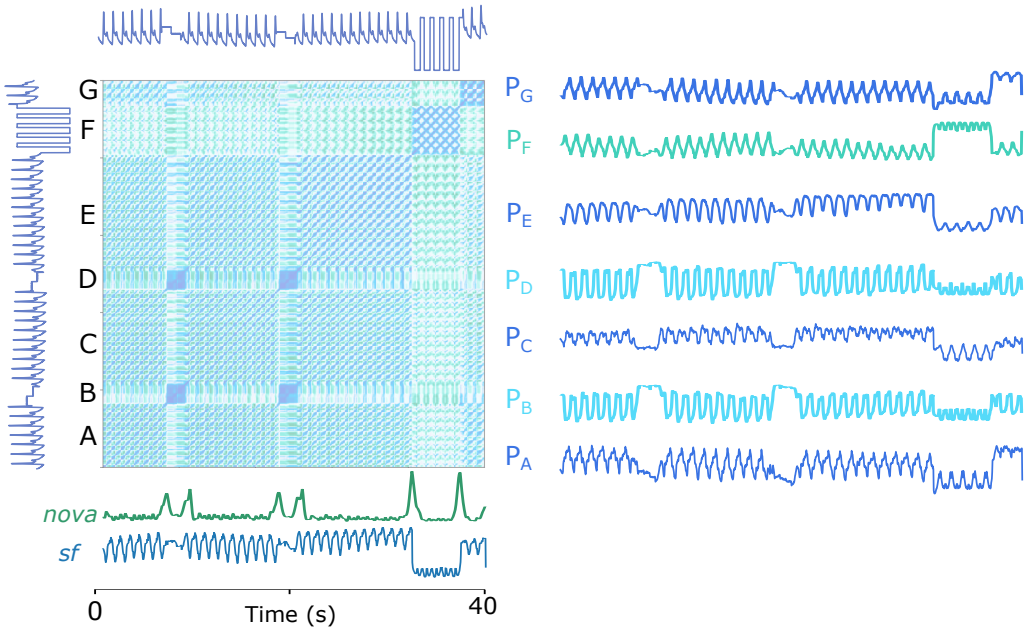


Figure 5.10: ABP signal. It represents the first 10000 samples of the signal from Figure 5.9. The parameters used were a window size of 250 samples, with an overlap of 95% and a kernel size of 200 samples.

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 5.10 where a segment of the original signal of Figure 5.9 (the first 10000 samples) is computed into the SSM. The resulting *sf* shows the periods of the ABP signal quiet well.

The SSM of Figure 5.9.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 5.10 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 5.11.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 [**pulsusparadoxus2**]. This also can occur when the patient's changes sleeping posture after a heart surgery[**eamonn_segmentation**]. 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.

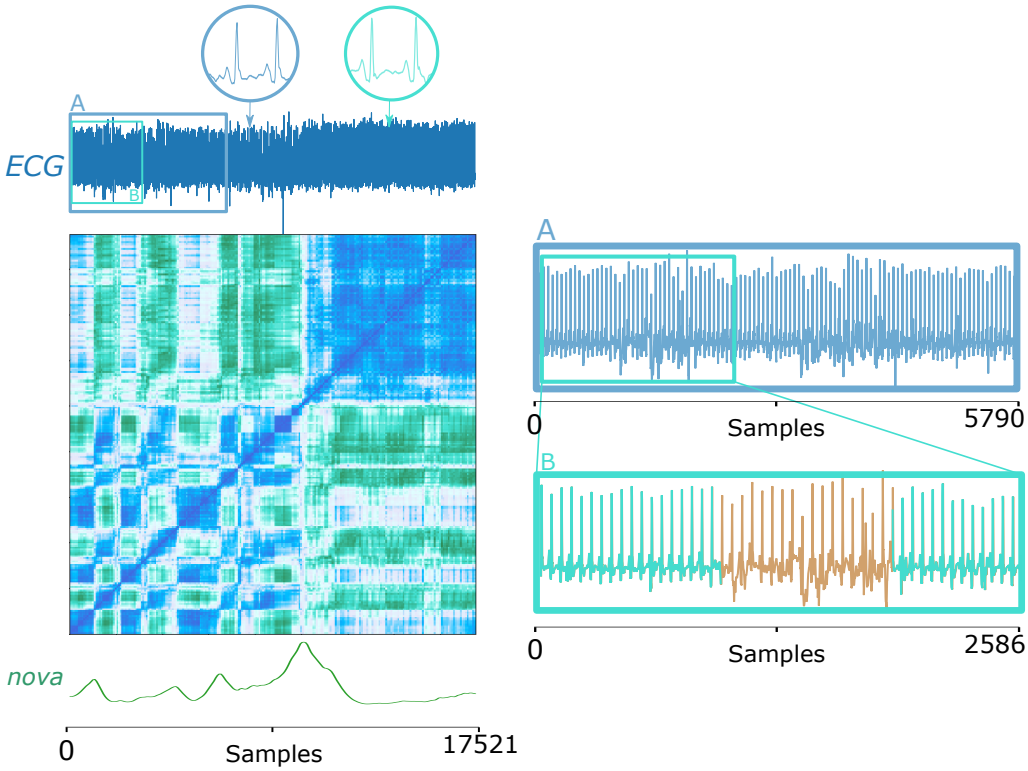


Figure 5.11: ECG signal with a *pulsus paradoxus* condition starting at the 10000th. The SSM shows the two modes of the ECG. Highlights of each mode are presented with the circle zoomed-in thumbnails. In more detailed are also shown segments A and B, which highlight an area where the SSM indicates possible changes.

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.

5.4.3 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 5.2, 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 5.12.a with record *"Occupancy"* from Dataset 4.0.5.

The record is a multi-dimensional time series that measures room occupancy based on temperature, humidity, light, and CO_2. All events can only be detected if using several time series of the record [**cpd_alan**]. In Figure 5.12.a, a single time series was analyzed by the proposed method to detect relevant events, while Figure 5.12.b is the result of using

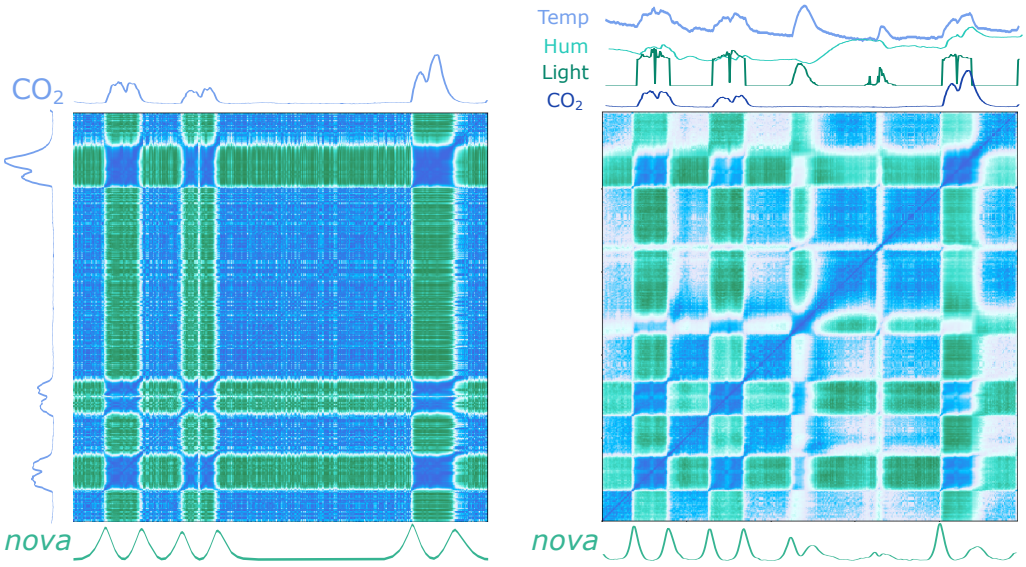


Figure 5.12: Proposed method applied on "*Occupancy*" record of Dataset 4.0.5. (a) A single time series of the record is used to extract events; while in (b) the SSM is computed with features extracted from the four available time series.

all the time series of the record.

5.5 Time Series Summarization

The aforementioned examples show how the proposed method can be used as a strong and reliable visual tool. It is possible to see how a time series is structured, how similar are segments, and if these are periodic or not. The information available is 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.

5.5.1 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

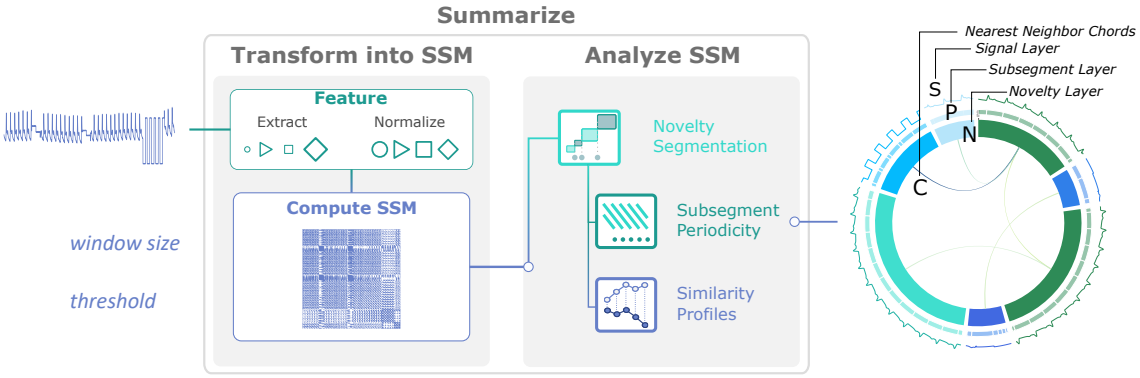


Figure 5.13: Steps to use the aforementioned methods on the SSM to summarize the time series into a circular plot with C - chords that connect nearest neighbors, S - the signal layer, P - the subseqment layer and N - the color coded novelty layer. Each of these elements of the summarized plots are built based on novelty segmentation, subsegment periodicity check and similarity profiles.

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.

5.5.2 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 [**bitmap**, **arcplots**]. These show where repeating sequences occur in a very concise way. This has a range of applications that include, for example, text and *DNA* sequence analysis.

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.

5.5.3 A step by step example

The steps are indicated in Figure 5.13 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 5.14, 5.15 and 5.16 show the step-by-step process to summarize the time series by analyzing the SSM.

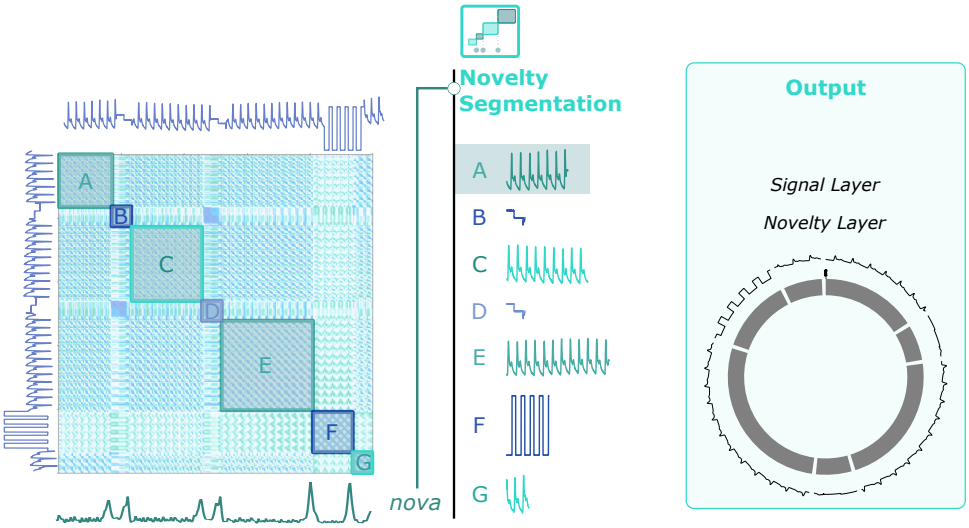


Figure 5.14: How to reach a standard format. This step applies the novelty search method to find the segmentation points. The signal is broken into A to G segments.

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 5.14.

Further, the segments are compared with the *similarity profiles* of each segment. Figure 5.15 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,

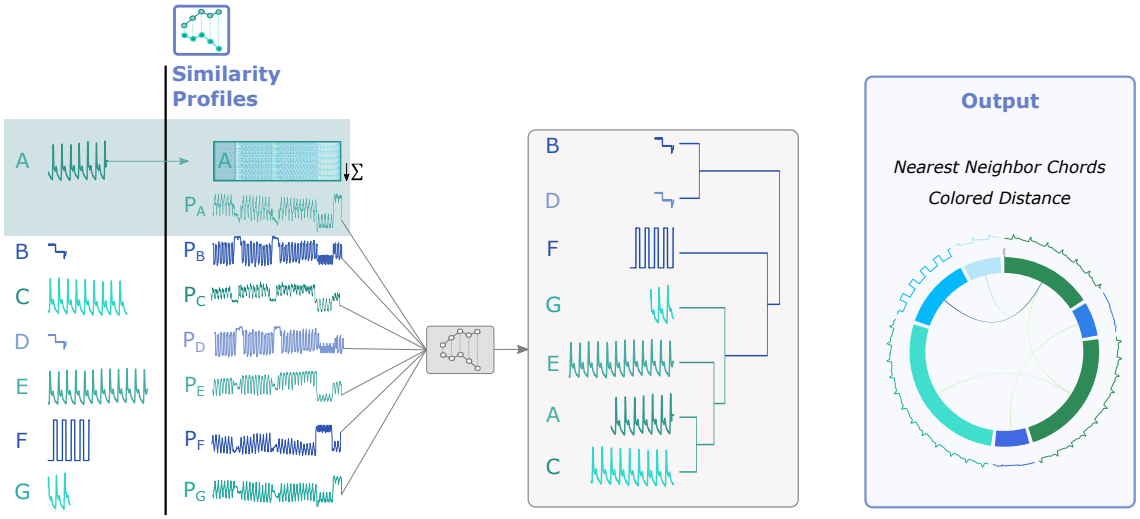


Figure 5.15: From each segment, a similarity profile can be computed. The pairwise euclidean distance is applied to these profiles, from which a dendogram is created. From this association, layer C can be drawn and the colors of the novelty layer can be added. Segment A is highlighted for the next step.

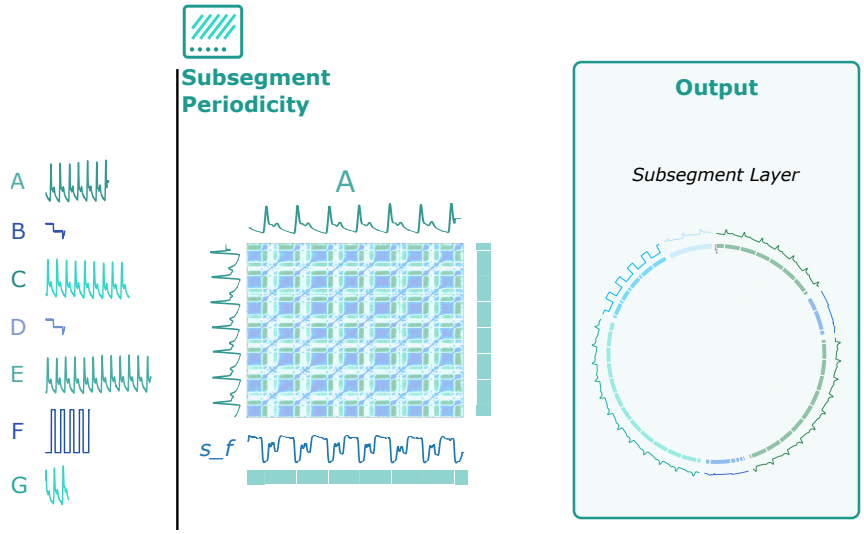


Figure 5.16: After finding the first layer of the segmentation points, the periodicity of each segment can be checked and added as a sublayer of segments, adding layer P. In this case, segment A is the example.

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 5.15. The dendrogram shows that there are three main clusters, being cluster one represented by segments A, C, E, and G; the second cluster has segment F and the third cluster groups segments B and D. It is important to note that typical distance measures, such as the 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 5.16 illustrates the *periodic search*, where periods are segmented by the minima of the s_f.

This summarization process is mostly graphical and follows the results of each process for information retrieval from the SSM. Therefore, this method will not present overall results or be validated in Chapter 7.

6

Language for Time Series Data Mining

In "Pattern Recognition: Human and Mechanical", Satosi Watanabe defines a pattern as a vaguely defined entity that is the opposite of chaos, to which a name can be given [**watanabe**]. The recognition of these entities immersed in chaos is well performed by the human brain by distinguishing or finding similarities in features that characterize a certain pattern, being an innate capability for decision making. 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

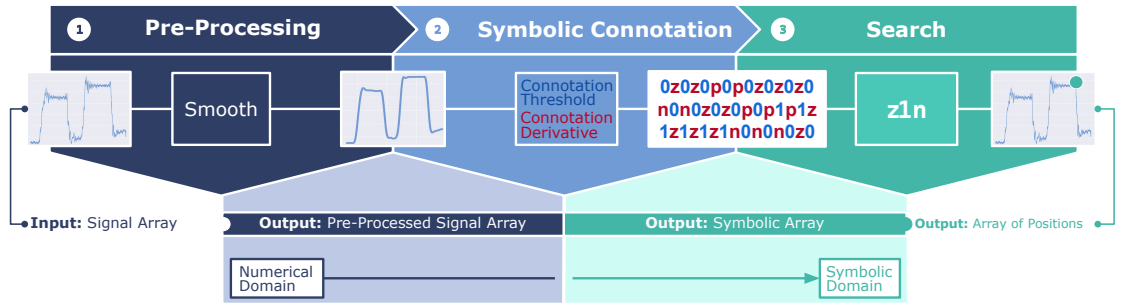


Figure 6.1: SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search. This Figure also provides an example with the sequence of changes that occur in each step.

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.

6.1 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.

6.1.1 Pre-Processing - Preparing the Data

The pre-processing stage is responsible for preparing the signal, either by removing noise that arises from several sources or adjusting the signal so that the information is unveiled. Traditional pre-processing methods, such as a pipeline of linear filters, moving window average filters and statistical de-noising or re-sampling techniques, are typical procedures to prepare the signal for further tasks [**preProcessing**]. The current approach uses a symbolic representation of these techniques, in which each is represented by its corresponding *token*, which can be a symbol or a function name. In order to manage the pre-processing tasks, a string containing a set of tokens and their corresponding arguments is written by following the polish notation [**polishNotation**], in which the token precedes the corresponding argument(s), and each element is separated by a white-space character. The available pre-processing methods to be used with SSTS are presented in Table 6.1.

In the example of Figure 6.1, 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 6.1.2.

6.1.2 Connotation - The Symbolic Time Series

In semiotics, it is described that the connotative aspect of an image or a sign corresponds to the extraction of meaning (sometimes called the *second meaning*), by the personal interpretation of its traits and characteristics [**connotation**]. The *connotation*step is 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 6.1: 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

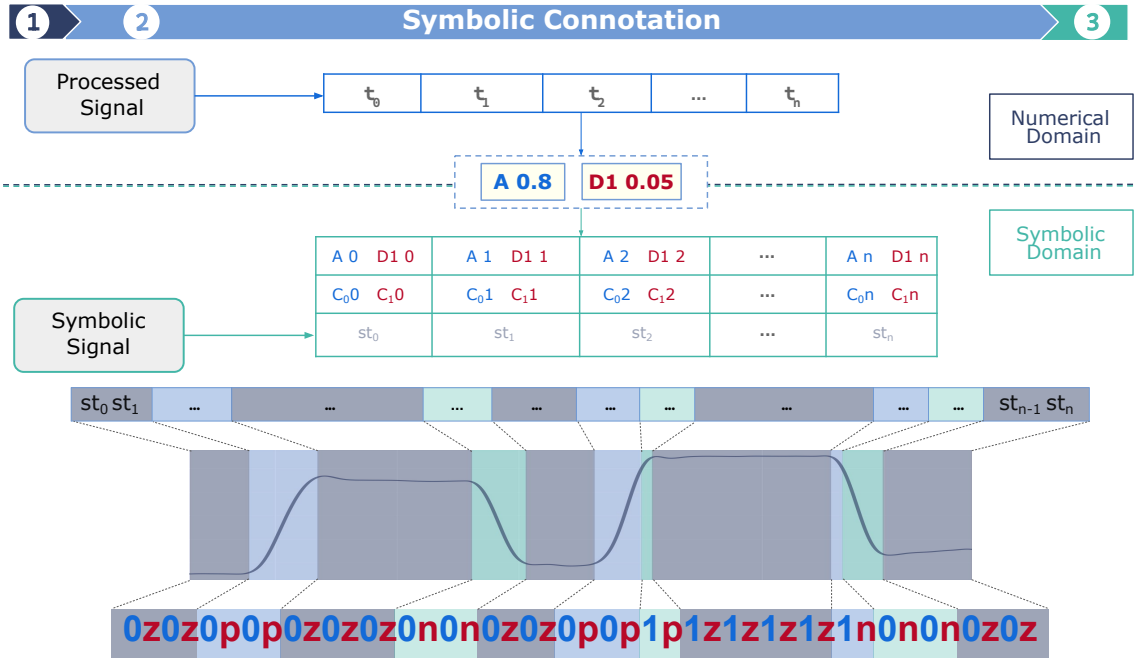


Figure 6.2: Here are highlighted the second stage of the process. It is the moment the signal is transformed from the numerical domain to the symbolic domain. A represents the amplitude method (blue) while $D1$ is the first derivative (red). The exemplified signal plus the symbolic translation are presented.

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 6.2.

Table 6.2: 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 6.3: 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.

6.1.3 Expressive Syntactic Search

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

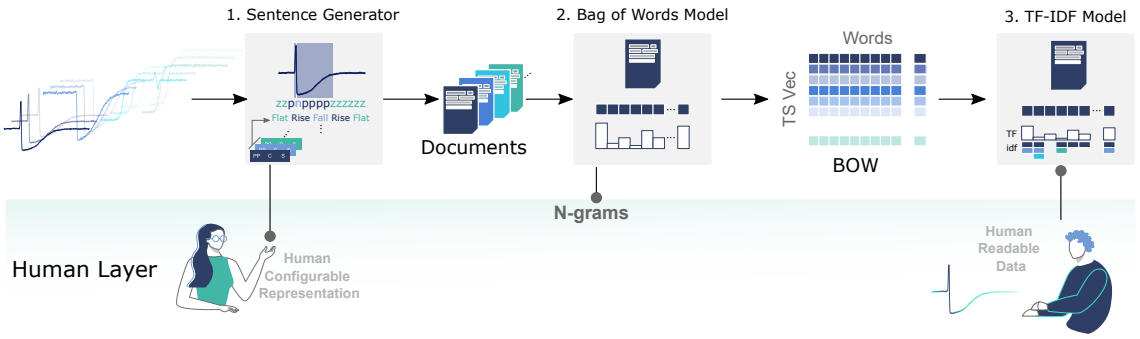


Figure 6.4: Steps of transforming time series into documents and corresponding BoW and TF-idf matrices for posterior classification. The steps are *sentence generation*, *bag of words model* and *tfidf model*. The human layer indicates when there can be human intervention. The steps are followed with 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.

6.2 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.

6.2.1 SSTS to Generate Time Series Documents

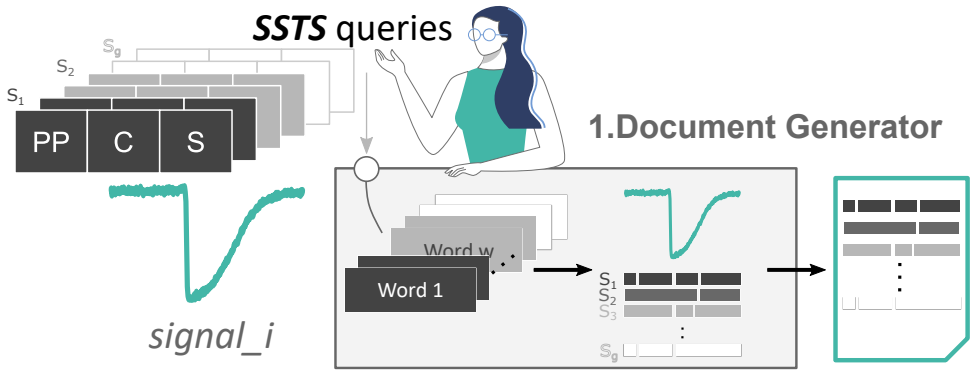


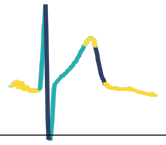
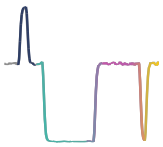
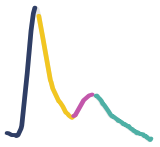
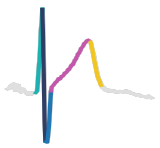
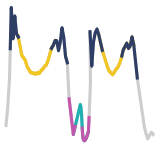
Figure 6.5: Steps for the generation of sentences from a raw time series and organization as a document. Here: PP - Pre-processing, C- connotation and S - Search are each SSTS queries used to search for the patterns and attribute the matched *subsequences* the corresponding *word*.

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 6.6.

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 6.3, 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 6.3: 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 6.6. 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.

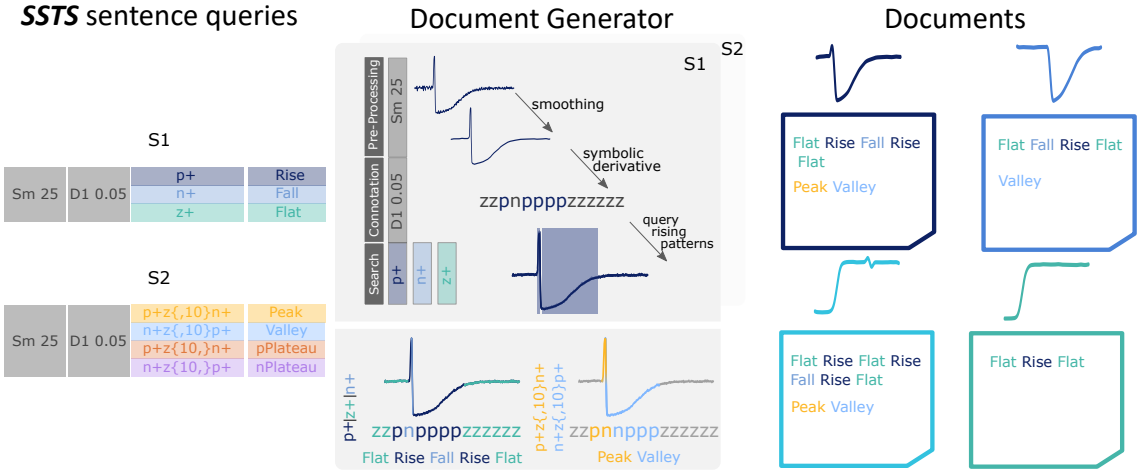


Figure 6.6: (Top) Using SSTS to detect the rising stage of a time series. Each step of the process is written described as follows: (1) pre-processing: *Sm* is the function *Smooth* with a window size of 25 samples; (2) connotation: *D1*, indicates the first derivate, from which each sample is converted 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.

6.2.2 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 [**scikit-learn**]. In addition, the BoW model feature matrix can be converted into the TF-idf feature matrix, which

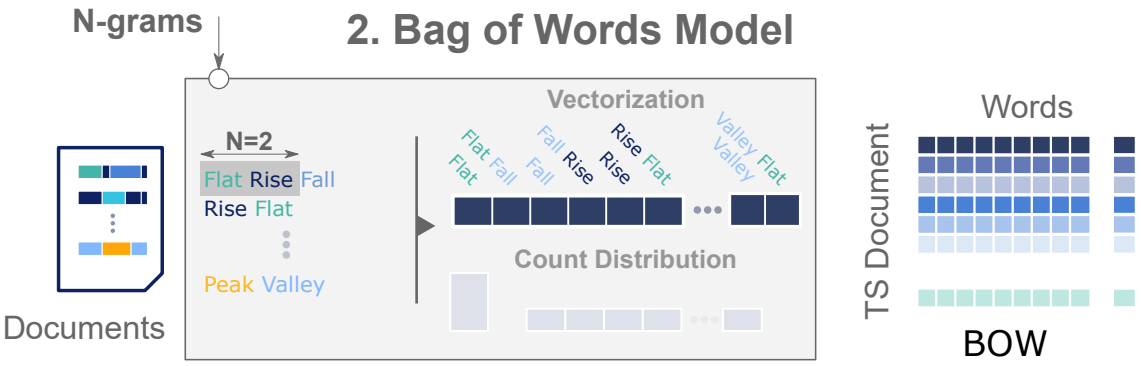


Figure 6.7: Process to transform a time series document into a vectorized representation of words and n-grams. The user can set the size of the n-gram. The documents are transformed into the BoW having a count distribution, which is the tf.

can then be trained with the same classifiers. In this work, we explored several of these combinations to understand which can give better results.

As was pointed out in Chapter 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 6.8 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),

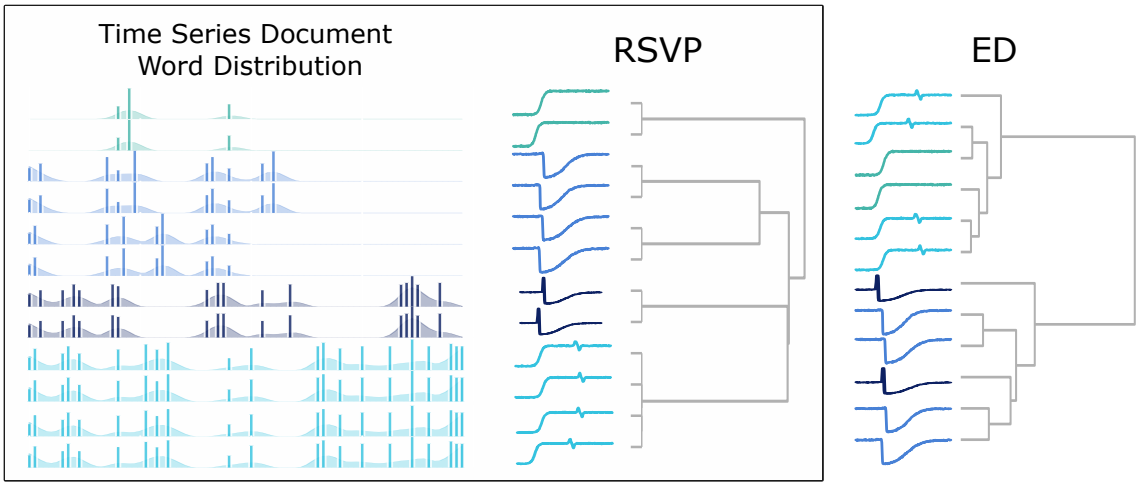


Figure 6.8: Comparing the clustering process of the ED and the proposed symbolic method. In the proposed method, each signal has a word distribution vector, represented on the left. The signals are clustered based on these.

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.

6.2.3 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 [**sax_vsm**].

The process to highlight the areas of the signal that are more relevant and able to explain the difference between classes is illustrated on Figure 6.9. 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:

$$(a, b) = ssts(t_j, w_i) \tag{6.1}$$

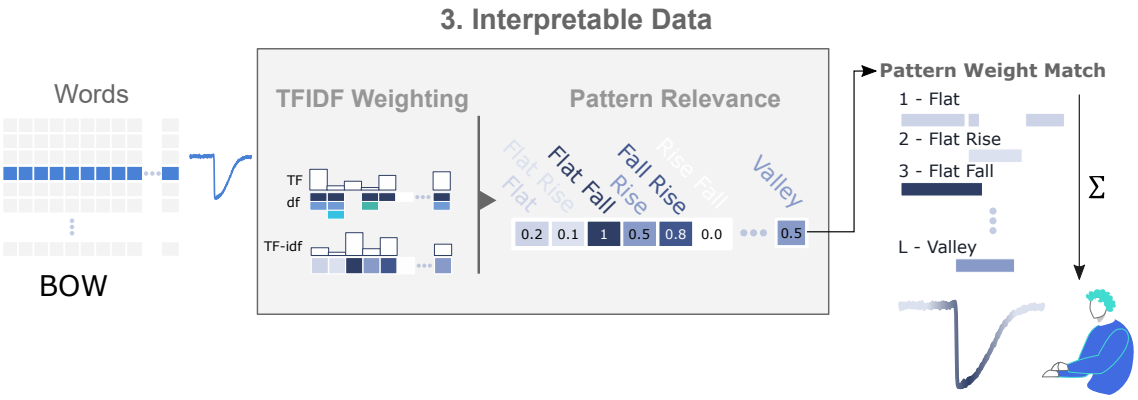


Figure 6.9: From the TF-idf matrix has weights for each word or n-gram (pattern relevance). The search of these words and cumulative attribution of weights on the segments matched can represent a feedback tool. In this case, the valley of the signal is highlighted.

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

$$WTS_i(a,b) = WTS_i(a,b) + h_{ji} \tag{6.2}$$

In Figure 6.9, the exemplified signal has a higher relevance on the `valley`, being the most characteristic element of that time series.

6.3 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*.

6.3.1 "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

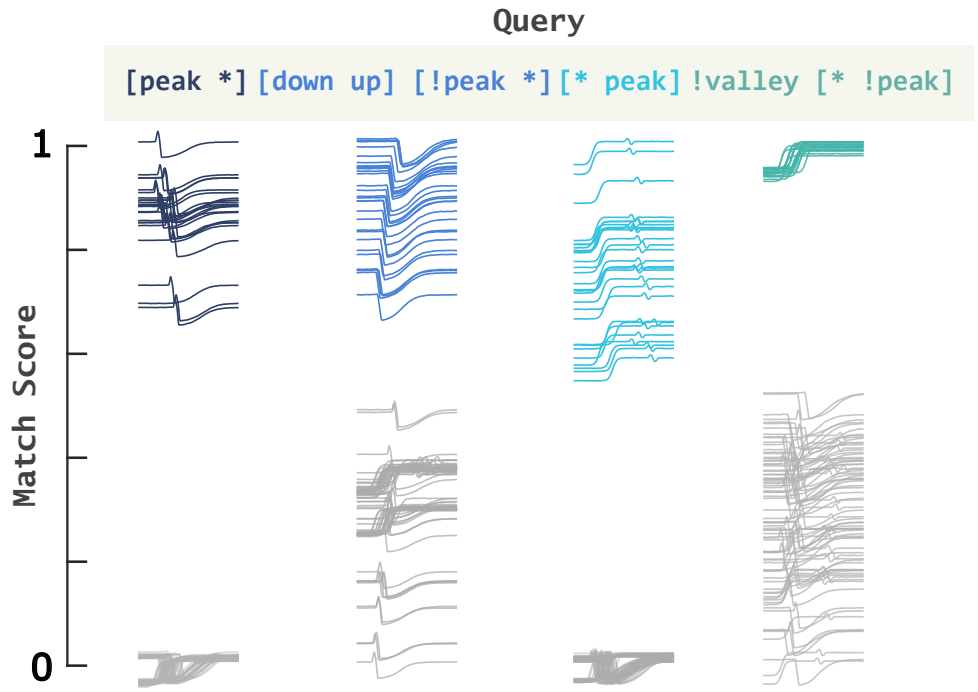


Figure 6.10: Using our proposed query language, we can create short, intuitive natural language queries to rank the 100 exemplars in Trace, separating our sought class from the remaining data. Here ⋆ is *anything*, ! is *not*, and *square brackets* are a grouping operator (more details in Section 3).

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

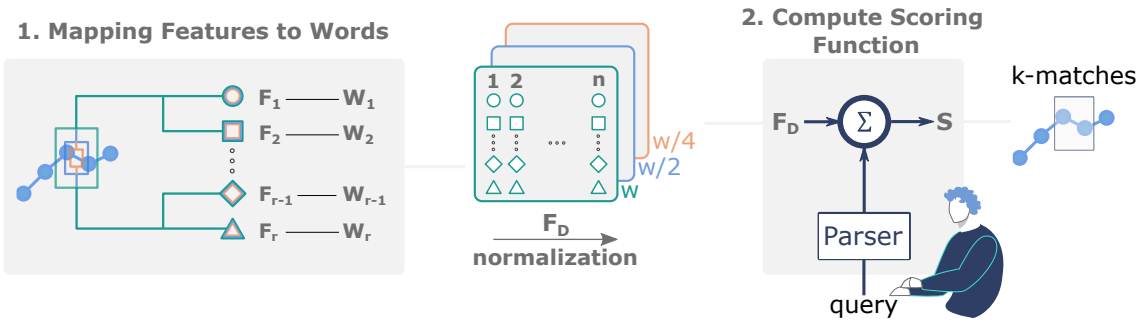


Figure 6.11: QuoTS steps. It shows the feature extractino process and matching each feature to words ($W_1, W_2, etc...$). These features are computed in three dimensions (w, w/2 and w/4). The user can input a query to search for a specific pattern. This query will be scored based on the parser and features. 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 6.11. 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.

6.3.2 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

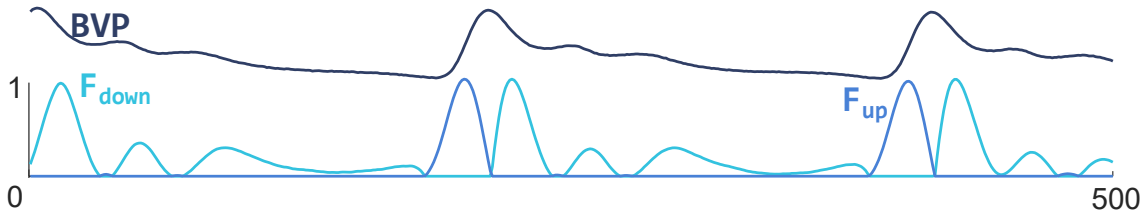


Figure 6.12: Example of a word feature vector. In this case F_{down} and F_{up} represent a negative and positive slope values.

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:

$$!W = 1 - W \tag{6.3}$$

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 6.3. 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 6.4).

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.

6.3.3 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.

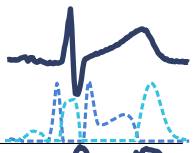
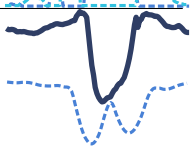
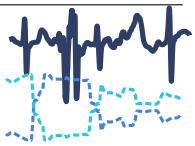
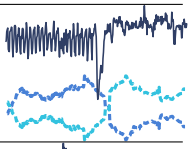
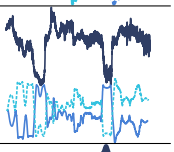
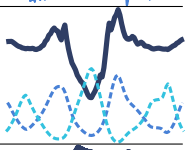
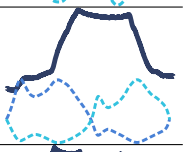
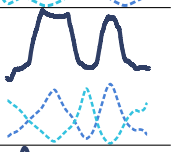
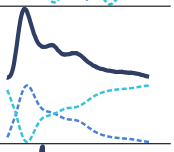
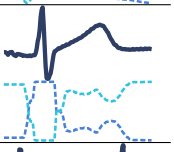
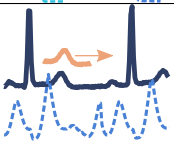
Web search engines have many operators at the user's disposal, but since a list of words is usually powerful enough to retrieve and correctly sort most of the desired results, very few (or none!) are often used. We believe that this is the case for this application as well but acknowledge that simple 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 6.3. 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 6.3 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 6.4: 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.

6.3.4 Natural Language Query for Time Series

As illustrated on Figure 6.11, 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.

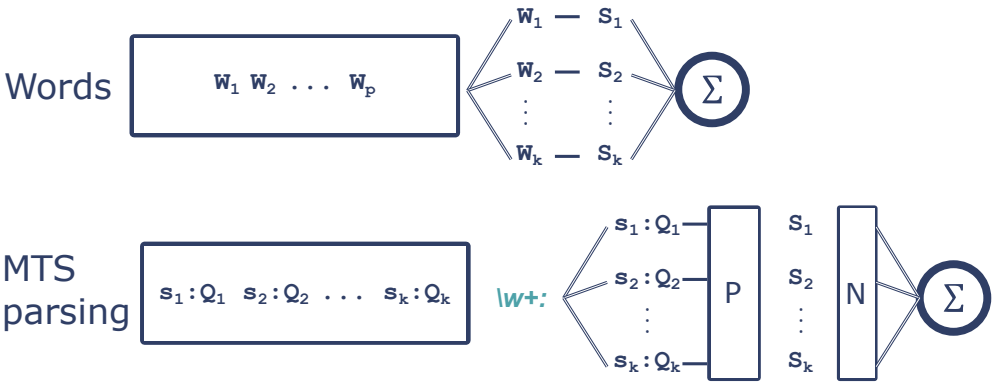


Figure 6.13: Process to parse the input text query. When having simple space separated words, each corresponding word feature vector is summed together. When in multidimensional mode, there is the *signal id* first.

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 6.13 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

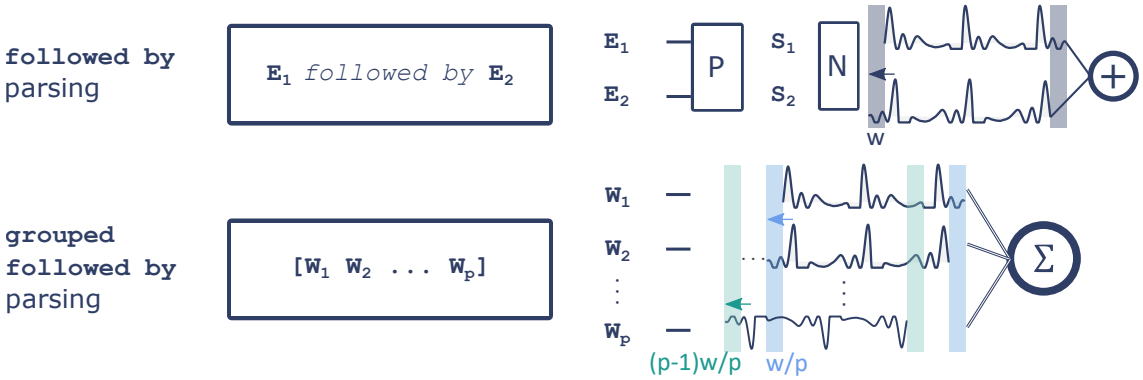


Figure 6.14: Second order of the parser. In case the `followed by` operator is used, the elements (E_1 and E_2) are parsed (P) and word feature vectors are summed, but the E_1 has to be delayed to count as the "previous" instance (w represents the selected window size). When using the more specific `grouped 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 6.13), 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 6.13) 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.

7

Results and Validation

The previous chapters introduced the methods developed for this thesis. A few examples were presented to help follow the explanation and description of the methods. In this chapter, we go further into testing the algorithms in corresponding tasks on the datasets introduced in Chapter **??**. The methods are tested and validated to provide evidence that the work developed in this thesis is relevant and contributes to the state of the art he time series data mining field.

Considering the diversity of methods developed, this chapter will be divided into every single contribution, providing results for each one of them. We will start with the methods developed under the topics of Chapter 5, more specifically for time series segmentation.

7.1 Segmentation

7.1.1 Novelty Segmentation

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

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

7.1.1.1 Performance on Public Datasets

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

Dataset	Signals	# Ch	Task	TP	FP	FN	Prec (%)	Rec (%)	F1 (%)
Dataset 1	ACC	3	HACP	98	16	16	0.860	0.860	0.860
Dataset 2	ACC-GYR	6	HACP	157	18	22	0.897	0.877	0.887
Dataset 3	ACC-GYR	6	HACP	1378	313	263	0.815	0.840	0.827
Dataset 4	ACC-GYR	12	HACP	499	71	38	0.875	0.929	0.902
Dataset 5	EMG	8	Act/Rel	309	0	0	1.000	1.000	1.000
Dataset 6	ECG	1	Noise	132	25	10	0.841	0.930	0.883
Dataset 7	ECG	4	Noise	21	2	3	0.913	0.875	0.894
Total	N.A.	N.A.	N.A.	2629	465	352	0.850	0.882	0.866

Table 7.1: Overall results for the performance of the method on novelty segmentation. The dimension of the records is presented on the column # *Ch*, as well as the types of signals used and the task in which applied (HACP - Human Activity Segmentation; Act/Rel - Activation/Relaxation of the EMG and Noise detection). The overall measures of Recall and Precision were micro-averaged.

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

Table 7.2: Distance error as a ratio of the time scale (T_s) for the detected TP.

The method has been computed in the same conditions and by following the same procedure for all records of all datasets. The features used have been the same for each record, varying the time scale parameter, the overlap size of the sliding window, and the kernel size. The peak detection strategy was the same for all records, which is based on a threshold mechanism. The threshold value varied for each record. Results for publicly available datasets are presented in Tables 7.1 and 7.2. Table 7.1 indicates the performance in detecting the change point events. Overall results for each time series of each dataset can be checked in Appendix A

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

The method achieved an overall macro-averaged 85.0% of Precision, 87.2% of Recall, and an F1 measure of 86.1%. Individually, the worst performance was found on Dataset 3, although all the Datasets accounted for similar results. A more comprehensive discussion

on *FN* and *FP* will be presented in a further section. More details are also provided regarding the time discrepancies between ground truth and estimated events.

7.1.1.2 Comparison with SOA Methods

In order to compare the proposed method with state of the art approaches, we used a benchmark provided by the Alan Turing Institute [**cpd_alan**]. The performance was evaluated by computing the proposed method to search for change point events in each of the time series available. The time scale and threshold values for peak detection were selected empirically and available in Table **??**. The results are summarized in Table 7.3 and Figure 7.1. Both cases show the F1 score. The performance considered for other methods was selected as the best score of each method available on the benchmark [**cpd_alan**].

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

Table 7.3: Comparison of performance between the proposed method (*Nova*) and other algorithms. The colors indicate if the other algorithms were better (Green) or worse (Blue) in performance than the *Nova* method for a specific dataset. Time series from *apple*, *bee_waggle_6*, *occupancy* and *run_log* are multivariate. The average results are grouped on the last row. Averages did not considered the gray columns, since these would imply that no change point should be detected, or an error on the signal was present. *T* appears when the method timedout, *M* and *F* when the method failed in compiling

.

As presented in Figure 7.1, the critical distance diagram ranks the proposed method in the top four, without a significant difference in performance (considering methods that

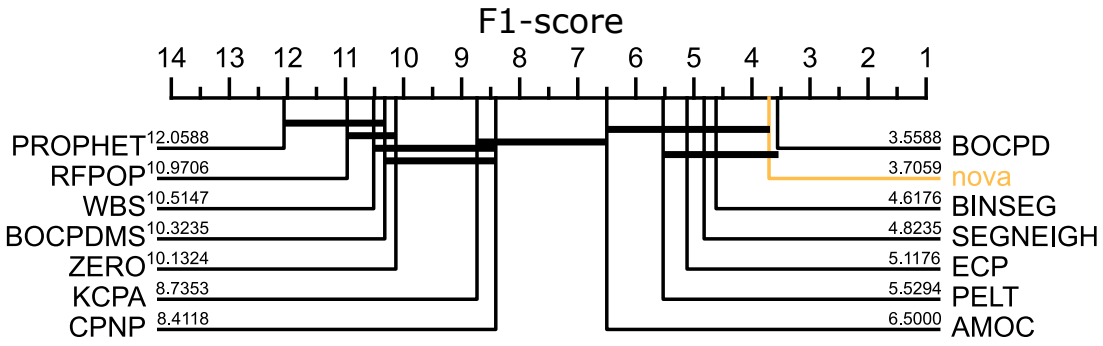


Figure 7.1: Critical distance diagram comparing the methods used in [**cpd_alan**] (except *RBOCPDMS*) and the *novelty function* (*nova* - highlighted in SeaGreen). The performance measure corresponds to the F1-score for all single-dimension datasets of the benchmark (except the ones highlighted in gray in Figure 7.3).

only work in uni-dimensional datasets). The average F1 measure is 79.6% for both uni and multi-dimensional datasets. Two of the F1 measures are null because, in these two time series, no change point was supposed to be found, but the proposed method is not yet prepared to return no results.

The results obtained in this benchmark demonstrate that this method is promising, having a performance that competes with several state-of-the-art methods in the problem of novelty segmentation. We highlight that the proposed method applies to multi-dimensional time series, while two of the best ranked methods in Figure 7.1 are not. In addition, this method is not solely focused on retrieving segmentation points, but also provides additional layers of information, namely periodic changes and the level of similarity between segments, which is an advantage over the *BOCPD*.

7.1.1.3 Insights on FN and FP

In this section, we present a more comprehensive discussion regarding the *FN* and *FP* counts. Regarding *FP* counts, these were found in several datasets to be mostly associated with changes on the signal that were not labeled. Since our proposed approach is *unsupervised*, most changes that appear significant will be detected, even if not explicitly labeled. This is visible, for example, in Figure 7.13, where the subject was climbing stairs but something happened during the acquisition that makes the signal change dramatically. We found the same type of mislabeled data on Datasets 2, 3, and 4, with Dataset 3 having the worst rate of *FP*. Specifically, in this case, part of these *FP* were due to subsequences that were not annotated, for instance, in Figure 4.2 we highlighted in light yellow areas where something happened but no label is given. As most of these datasets were used for classification purposes, some labels were not given, but our method, being unsupervised and not focused on classification tasks, can find these occurrences. The method also has a worse rate of *FN* for this dataset. We can explain these by the difficulty of the method in finding the labeled transitions between static positions (sit-to-stand, stand-to-sit, stand-to-laying, etc...). The method was able to find these as one segment but was not able to find

the start and end of each of these transitions, as it was labeled. Detecting the start and end of these would require smaller window sizes.

For datasets with categorized events, which are a transition between one categorized *subsequence* to another, we can evaluate from which categories the events were missed. Then, identify if specific types of events were the cause of the *FN* count rate. Figure **??** indicates the F1-score of the *nova* function in detecting the different transition categories of events using the color and size of circles associated with each event category. The activities before and after change point events are sorted in columns. Each row is associated with a specific event category and corresponding circle, which color indicates the detection's accuracy, while the size indicates the percentage of events associated with the corresponding event category.

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

An example of the enhancement process is better seen on Figure 5.8 from Chapter 5. The block *A* is highlighted, showing clearly the three modes of the time series. Now that we only focus on that segment, we can perceive these transitions. This example demonstrates that this tool has the potential of enhancing the detection of several transitions in a hierarchical process by re-normalizing segments of an initial segmentation. This would come without a significant extra computational cost since these measures of the SSM are already computed. Nevertheless, there is the potential of finding extra undesired events depending on the threshold level.

7.1.1.4 Comparisons to Related Work

We will now look for specific datasets from the *TCPD* benchmark that should be highlighted considering the results obtained. In Table 7.3, we notice that datasets, such as *global_co2*, *iceland_tourism*, *lga_passenger* and *us_population*, have a lower score when compared to the best methods. In the other end, the proposed method was better than the best methods in datasets *bitcoin*, *brent_spot*, *businv* and *jfk_passenger*. From these results, we notice that the proposed method is more fragile when events occur as a slight change in the trend of a signal, such as the case of *global_co2* or *us_population* records, while it is better for more significant changes in a pattern, amplitude or other property.

The results show that this method is highly competitive in performing event detection

in time series, with high performance scores in real-world datasets. From all methods tested in the *TCPD benchmark*, only the *BOCPD* method was able to obtain similar results (Figure 7.1 and Table 7.3), considering also multidimensional datasets, but it does so with an optimization of many parameters. The proposed method is being computed with several parameters as well but has the potential to accept only one.

One of the other great advantages of the proposed method is that it also provides rich visual feedback, which can be perfect to support the analyst in interpreting a time series. The content of the SSM goes beyond change point detection, providing relevant information about periodicity and similarity.

7.1.1.5 Effects of Window Size and Overlap

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

These parameters can be explained with the analogy of a camera. The window size works like the *zoom function*, defining the scale of interest in the time series. Larger windows correspond to lower *zoom values* and will compute the similarity of larger *subsequences*, while smaller windows are like a *zoom-in* function on the time series, searching for any small detail that might be a change. The overlap percentage is the *camera sensor*, which defines the pixel resolution of the image and works as a downsampler of the time series. A full resolution of the SSM is only achieved with total overlap, and the lower the overlap percentage, the less accurate are the highlighted changes.

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

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

In addition to this, there is the fact that the parameters can be configured for the type of problem and dataset. For instance, looking at the detailed tables presented in Appendix

A, the window size and overlap size are in most cases the same for the same dataset, while the kernel size varies in the same order of values.

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

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

7.1.2 Cyclic Segmentation

The detection of cycles was validated on dataset **??**. It consists of 15 multivariate time series with the same repeated cyclic activity. Each time series was segmented based on the similarity function (sf), which enhances the periodic nature of the time series when present. This method was applied to this dataset, being the results displayed in Table 7.4.

Table 7.4: Detected cycles and DE results of the detection of type 2 events, over the *HAR* database.

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

The results show the ability of this function to segment the time series into each cycle.

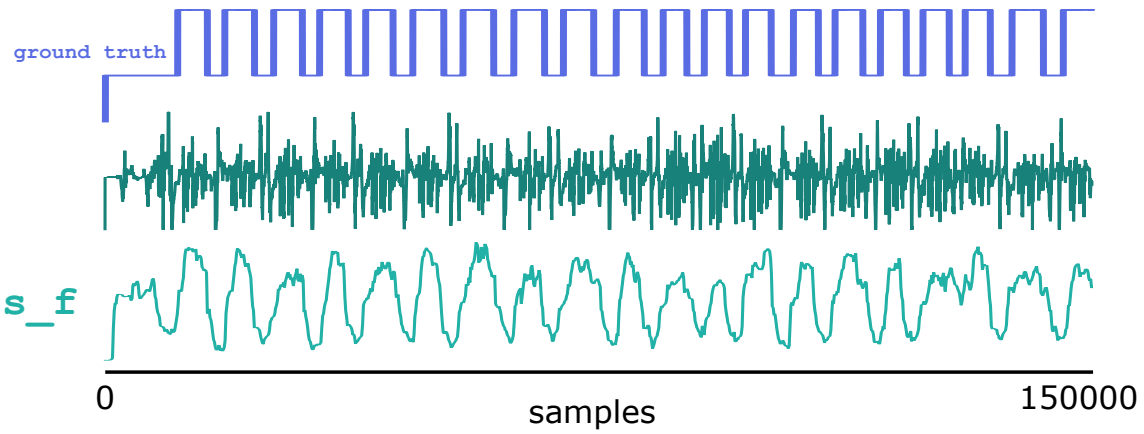


Figure 7.2: Segmentation of a walking pattern on a 1D signal with the similarity function (s_f). The window size was 5000 samples and the overlap percentage of 75%.

It has successfully segmented all time series in all desired segments while having a DE compared to the ground truth. This ground truth indicates the moment the segment starts from a manual push button.

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

It is also important to mention that this problem is not necessarily hard to solve for this dataset, considering that it is a controlled dataset with one periodic signal. But our intention with this validation was to demonstrate the ability of the SSM in finding periodic structures with a multidimensional signal, and without any prior knowledge of it. As an example, we show a more difficult walking pattern to analyze, where it is more confusing to find the cycles:

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

7.2 Pattern Search with SSTS

In order to show the value of using a symbolic representation of the time series and search with regular expressions on that representation, we provide six examples of how

to use SSTS for pattern search on time series and compare the text complexity between the python implementation and the query used, with Halstead measures.

7.2.1 SSTS on selected Use-Cases

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

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

Pre-Processing	Connotation	Search
LP 2 Sm 50	A 0.4 D1 0.05	0z0p
LP 2 Sm 50	A 0.3 D1 0.05	1z1p

- Example 3 - Segmentation of the systole of the ABP wave: The dicrotic notch corresponds to the physiological event of the aortic valve closure, which triggers the increase of the aortic pressure and signifies the end of the systolic phase. In this problem, it is asked to segment the systolic phase of the ABP wave;

Pre-Processing	Connotation	Search
BP 1 20	A 0.3 D1 0.01	(1p).+?0

- Example 4 - Electrocardiogram peak detector: Detect all the major peaks from an ECG record;

Pre-Processing	Connotation	Search
BP 5 50	D1 0.01	pn

- Example 5 - Straight Line Trajectory Tracking: The automatic detection of trajectory features, such as straight walks and turns can be used to evaluate trajectory anomalies from a vector of cartesian coordinates in 2D-space;

Pre-Processing	Connotation	Search
none	D2 0.05	z*?

- Example 6 - Since the first 5 s of the lifting exercise are unstable, the problem involves detecting the first stable lifting step, which occurs approximately 5 s after the beginning of the exercise from an accelerometer signal.

Pre-Processing	Connotation	Search
Mag Abs Sm 1000 | Mag Sm 750	A 0.2 | A -0.2 D1 0.01	(?<=1.15000,)(n.p)(.*?)(n.p)

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

7.2.1.1 Use-Case 2 - Step Detection in accelerometer signals

In this particular case, only the detection of the right heel contact will be discussed, whereas the left heel contact example's solution is presented in Table **??**.

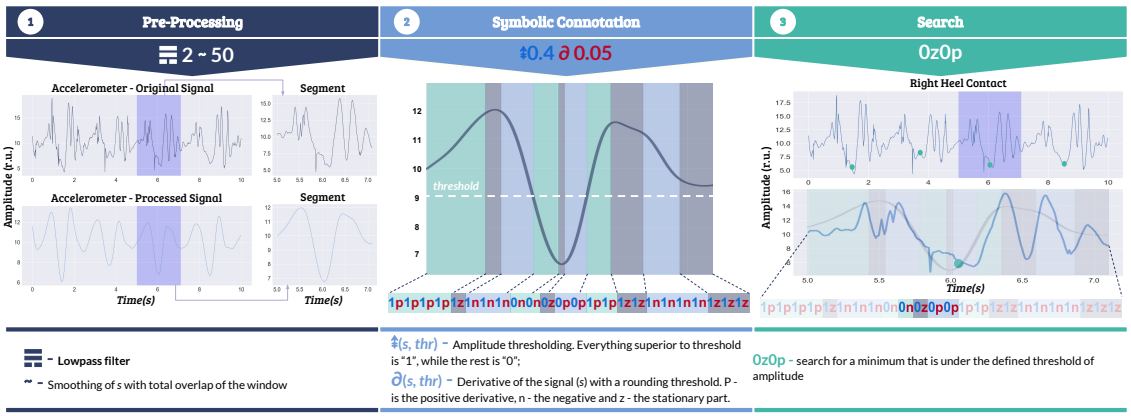


Figure 7.3: Example 2 - Solution pipeline of Step Detection example. At the bottom of the figure are summarized the operators and methods used in each step. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - A - Amplitude Comparison, Red - D1 - First Derivative). A dotted line is shown to represent the threshold level. Each color band in the signal corresponds to a specific primitive. A positive match is highlighted with green in the search step.

The signal is initially pre-processed to ease the identification of the subject's steps. This is achieved using a low pass filter (LP). The result is presented in the second image of the first step in Fig. 7.3. A highlight is also present and delineates a segment of the signal that

has a minimum peak, which corresponds to right heel contact. The string representation of this segment is depicted in the second step.

The string is a sequence of primitives composed of two connotation methods (A in blue, D1 in red). Based on these methods, the samples of the signal with amplitudes greater than *40*% of the range amplitude are transcribed into 1, while the remaining turn into 0; the slopes are converted into p, z and n, when rising, being stationary and falling, respectively.

The solution involves detecting each minimum with an amplitude inferior to the threshold level. The morphological representation of a minimum can be reduced to a negative slope followed by a positive one, which in the symbolic representation is defined by the second connotation method as the transition from n to p or z to p. Regarding the amplitude requirement, it is assigned by the first connotation method, in which any value lower than the threshold level is 0. The regular expression used to find this minimum was $0z0p$, which implies that: (1) the amplitude has to be 0; and (2) the derivative is z and then p. The detection is highlighted in green in the last plot of Fig. 7.3.

7.2.1.2 Use-Case 3 - Dicrotic notch detection in ABP signals

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

These types of signals are commonly affected by low-frequency noise that modulates the signal and is contaminated with high-frequency noise of low amplitude. The typical procedure is to bandpass the ABP signal to remove both types of noise. Fig. 7.4 depicts how a band pass filter that cuts frequencies under 1 Hz and above 20 Hz (BP) is applied to the signal to remove both types of noise. The modulation removal is shown with the linear regression in both original and pre-processed signals, which in the first case has a positive slope and after the pre-processing is approximately zero.

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

The first connotation method is necessary to distinguish between the rising slope that occurs at the beginning of the pressure wave and the one after the *dicrotic notch*. With this distinction, it is possible to find the beginning of the ABP wave as a high positive slope ($1p$) and find the *dicrotic notch* when the lower slope starts (0.). In order to find this area of the ABP wave, the regular expression has to start with the first $1p$ primitive and end with the first 0. primitive.

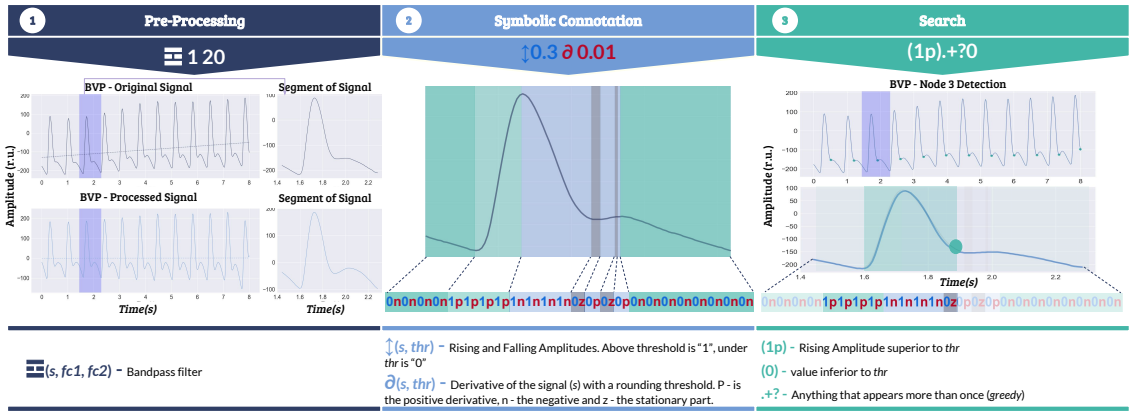


Figure 7.4: Example 3 - Solution pipeline of Dicrotic notch detection example. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between methods is represented with colors (Blue - AD, Red - D1). For each color band in the signal, there is a corresponding primitive. The match is highlighted with green in the search step.

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

7.2.1.3 Use-Case 6 - Stable lifting detection in accelerometer signals

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

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

Fig. 7.5 demonstrates how the example is solved. In both pre-processing and symbolic connotation steps, a vertical bar | separates the methods that are applied for each representation of the same signal. The pre-processing phase uses a sequence of whitening, modulus, and smoothing of the signal for "*Process A*", and a sequence of whitening and smoothing for "*Process B*". The first processing sequence turns the signal similar to a plateau, in which the beginning of the activity is easily identified; whereas, in the second sequence, the signal is smoothed such that each lifting step is well defined.

Regarding the symbolic connotation step, the first method AD (represented in blue) turns all sample values of the first signal that are higher than the threshold level into 1, while the remaining samples are converted to 0. The second set is applied to the second signal and uses the derivative of the signal D1(represented in red). Both symbolic

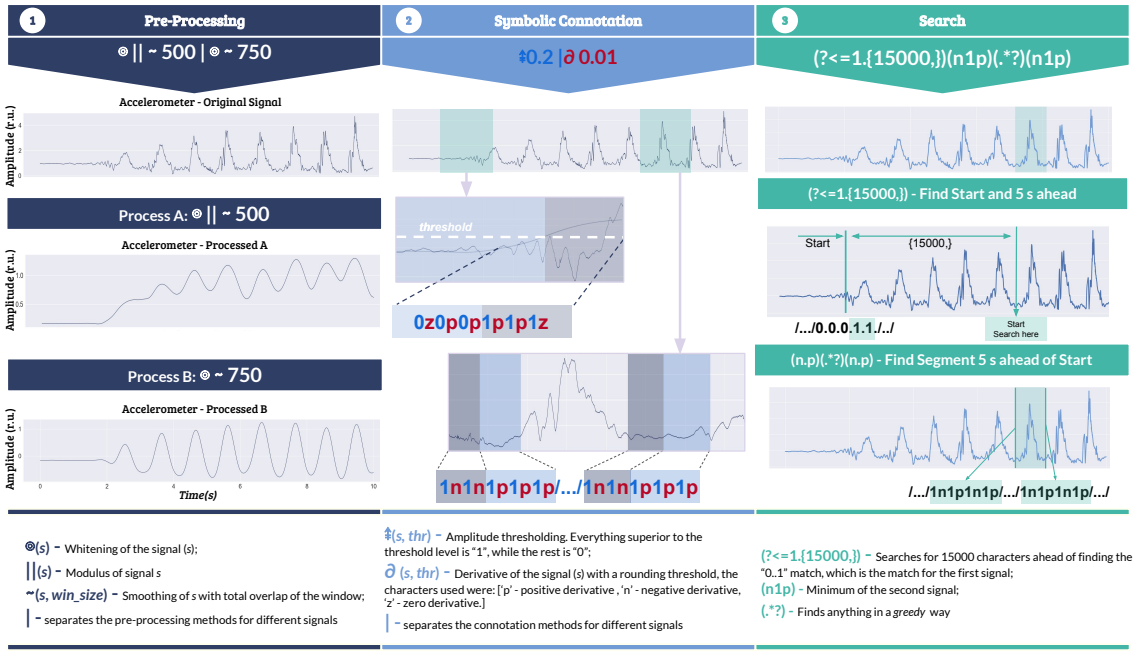


Figure 7.5: Example 6 - Solution pipeline of Stable Lifting Detection. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - AD, Red - D1). The threshold level is identified with a white dotted line. Each color band in the signal corresponds to a specific primitive. The match is highlighted with green in the "Search" step.

connotations merge into a sequence of primitives, in which the first element inspects if the exercise has already started (1.) or not (0.), and the second infers the sectioning of the lifting steps, that is if the sample of the signal is increasing ($.p$), stationary ($.z$) or decreasing ($.n$).

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

7.2.2 Measure of Expressiveness

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

with the number of distinct oprt and oprd, and the Toprt and Toprd (recall Section 2.8.3 from Chapter 2.

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

$$pks = peakdelta(s, delta = np.percentile(s, 70) - np.percentile(s, 30)) \tag{7.1}$$

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

In the second case, both **connotation** step and **search** procedure will be inspected for the **difficulty** measures. Regarding the **connotation** step, the evaluation will be made similarly to the previous method for inspecting functions, in which the function is set as an operator and its corresponding arguments as operands. For the **search** procedure, the set of instructions typically used in regular expressions (except the "." character) is defined as operatorsregex. These can be revisited in the second section. The remaining characters of the regular expression string, except the parenthesis, will be set as operands. In this are included the "." operator that matches any character, and the "\d" and "\w" that matches all digits and alphabetical characters respectively. Regarding the parenthesis, when occurring ("{}" or "[]"), will be set as one operator. For example:

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

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

Table 7.5: Evaluation of the complexity in the resolution of examples with the *syntactic* approach and with the *classical* approach. Voc - Vocabulary, Lgth - Length, CL - Calculated Length, V - volume, D - difficulty, E - effort.

Example	Syntactic Resolution						Classical Resolution					
	Voc	***Lgth***	***CL***	***V***	***D***	***E***	***Voc***	***Lgth***	***CL***	***V***	***D***	***E***
Start/End of Plateau	7.0	7.0	13.6	19.7	2.0	39.3	12.0	13.0	33.3	46.6	2.5	116.5
Step Detection (Left & Right Heel)	7.0	8.0	13.6	22.5	2.2	49.4	19.0	22.0	63.6	93.5	4.2	388.2
Systolic Phase Detection	8.0	8.0	16.4	24.0	2.5	60.0	21.0	25.0	73.0	109.8	4.7	517.7
Electrocardiogram Peak Detection	4.0	4.0	4.8	8.0	1.5	12.0	9.0	12.0	20.3	38.0	3.0	114.0
Straight Line Tracking	4.0	4.0	4.0	8.0	2.0	16.0	25.0	33.0	93.5	153.2	5.3	811.3
Stable Lifting Step Detection	13.0	21.0	35.6	77.7	4.5	349.7	22.0	23.0	77.3	102.6	5.0	512.8

Table 7.2.2 is presented the performance measures in both ways of solving the demonstrated examples. For the evaluation, the pre-processing step was omitted, which means that only the regular expression is used to compute the difficulty measures for the *syntactic* approach, and only the script after the pre-processing steps is used to compute the difficulty measures for the *classical* approach.

7.3 Time Series Classification with HeaЯTS

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

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

it from others.

7.3.1 Classification Performance

Figure 6.8 shows us how vectorized documents can be compared with the cosine distance to distinguish time series. Simply by using part of the SSTS queries, it is possible to generate TF-idf weights that are differently distributed based on the words and how these are ordered on a time series document. Of course, the *Trace* dataset used for this example is a simple dataset, with simple dynamics, but still challenging to classify well with the 1-NN ED (only 76% of accuracy). Therefore, to validate the ability of the proposed method to perform time series classification, it was computed on all datasets of the UCR Benchmark and its performance compared with the standard 1-NN ED. The process was divided into two stages. First, the parameters to compute HeaRTS were optimized on the standard train and test sets from the *pyts* distribution. The optimized performance was compared with the 1-NN ED method. Second, we used the optimized method on 10-fold cross-validation and compared the average accuracies with the 1-NN ED (the implementation from *pyts* was used to compute the results of the 1-NN ED method).

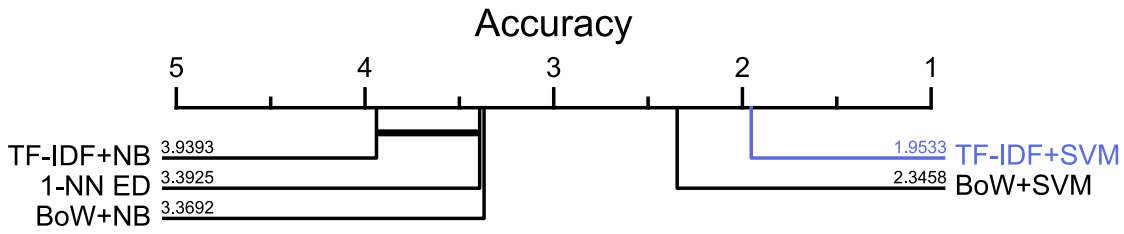
As mentioned on Chapter **??**, the method translates the time series into text with SSTS queries from Table 6.3. These have pre-processing, connotation, and search steps. For each step, relevant parameters are necessary depending on the queries used. For this experiment, we used as pre-processing a simple smoothing, which requires a *window size* (w_s). The derivative connotation steps also required a specific threshold (thr). In addition, the BoW and TF-idf models can be built with different *n-gram* sizes. These three parameters were optimized with a grid search method over a range of values:

$$\begin{aligned} w_s &\in [1, 10, 25, 50, 100, 250] \\ thr &\in [0.001, 0.01, 0.05, 0.1] \\ n-gram &\in [1, 2, 3, 4, 5] \end{aligned} \tag{7.2}$$

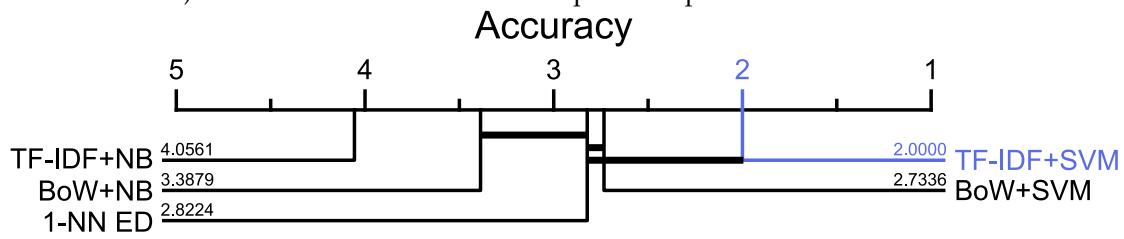
We performed the classification with several strategies and combinations, namely: BoW with Baysian Classifier (BoW+NB), BoW with SVM (BoW+SVM), TF-idf with Baysian Classifier (TF-IDF+NB) and TF-idf with SVM. The summary of the performance is presented on Figure **??**.

As can be seen, the strategies that rely on a SVM classifier are more robust. The TF-idf turns out to not be reliable when used with the Bayesian Classifier, which was expected since this type of classifier is purely probabilistic, which is why it works better with the BoW. On the other end, the TF-idf model works well with SVM classifiers, being the best method studied. We then compared it in more detail with the 1-NN ED classifier, as presented in Figures 7.6a, **??** and 7.8.

The average accuracy for the total 107 datasets is 79% for HeaRTS and 71% for the 1-NN ED. Special cases are highlighted on Figure 7.7.right. These indicate that the proposed method works poorly in problems with a high number of classes (*Phoneme-39*, *Adiac-37*,



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



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

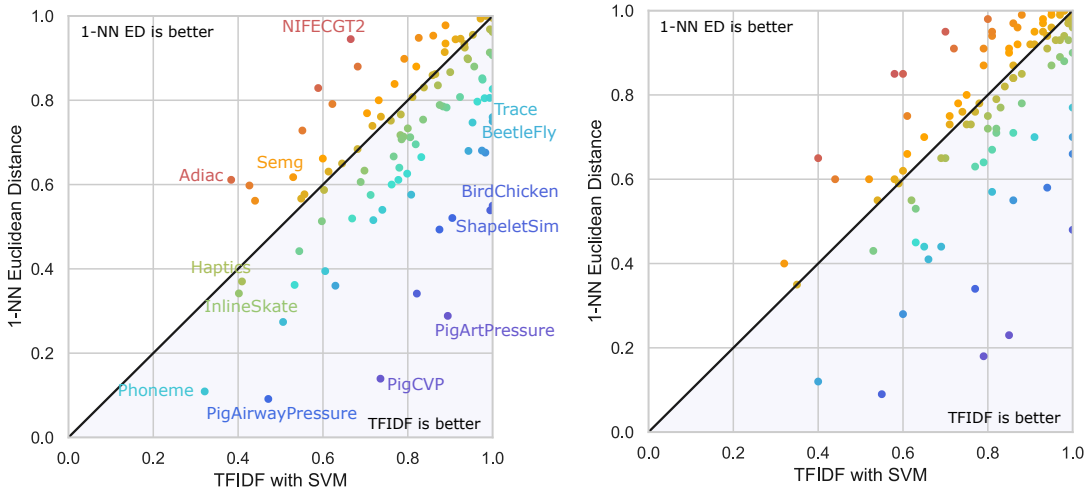


Figure 7.7: right) Classification accuracies for the Bag of Synthatic Patterns in comparison with the euclidean distance combined with 1-NN classifier (1-NN ED) with standard train and test; left) with a 10 fold cross validation.

PigAirwayPressure-52, etc...), although some exceptions are found, such as in *PigArtPressure*. Both *Haptics* and *InlineSkate* are badly classified in general by state of the art methods. Binary problems that rely in shapes well described with the derivative properties, such as *ShapeletSim*, *BirdChicken*, *BeetleFly* and *Trace*, are typically better classified.

The same conclusions are confirmed on the 10 fold cross-validation (Figure 7.7.left. Surprisingly, the proposed solution with a TF-idf and SVM classifier is very consistent with the previous results, even increasing to 80% the average accuracy for all datasets. What also is surprising is the improvement of the 1-NN ED method, which increased its accuracy to 75% on average. The critical distance for this performance is now less evident (Figure **??**, but the proposed solution is the overall best method. Even when compared to

the other methods, it is overall better in most datasets of the benchmark. The only method that competes is the BoW with an SVM classifier, which has a very similar performance (upright scheme of Figure 7.8.

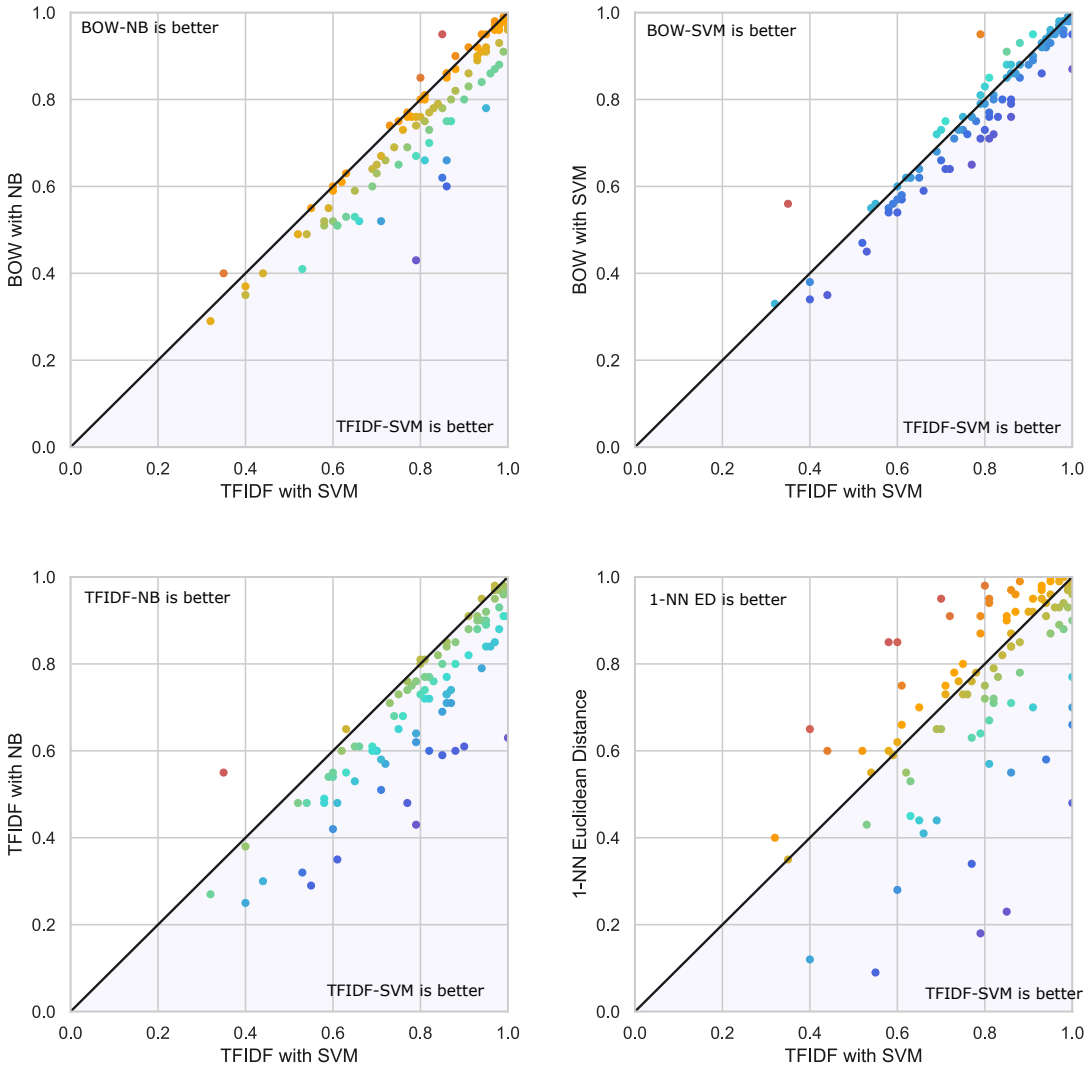


Figure 7.8: Classification accuracies for all experimented methods versus the accuracies of the TF-idf method with an SVM classifier.

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

7.3.2 Interpretable Data Outputs

Besides the fact that this approach is capable of performing well in classification tasks, we expected to go further by providing some feedback from the textual description, helping to understand what differentiates the classes. The TF-idf model gives weights for each

word that characterizes the time series. Searching back the *words* on the time series and summing the weights corresponding to these *subsequences*, a shape relevance is made on the time series, where areas of higher interest and that differentiate the time series from others are highlighted. The next Figures (7.9, 7.10 and 7.11) are displayed showing the highlighted areas based on the TF-idf weights.

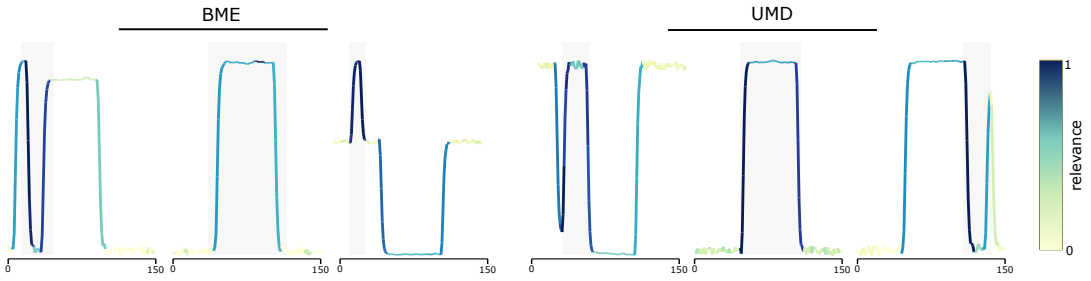


Figure 7.9: Interpretable results with highlighted shapes for the *UMD* and *BME* datasets. The *UMD* dataset was classified with with an accuracy of 99.3%, computed with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$. The *BME* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$

On Figure 7.9 are showed two similar datasets. The f1-score of this task was 1, meaning that it was able to perfectly characterize each shape with the textual representation. For instance, the *BME* dataset has two different sequences of shapes, since the first class has a `peak followed by a positive plateau`, while the third class has a `peak followed by a negative plateau`. The second class has simply a `plateau`. The main difference between all those classes is the peaks and what follows them, which is precisely what is highlighted. The same happens on the *UMD* dataset, in which the most relevant element is the peak and the plateau. As we can see, the transition from the peak to the plateau is highlighted.

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

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

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

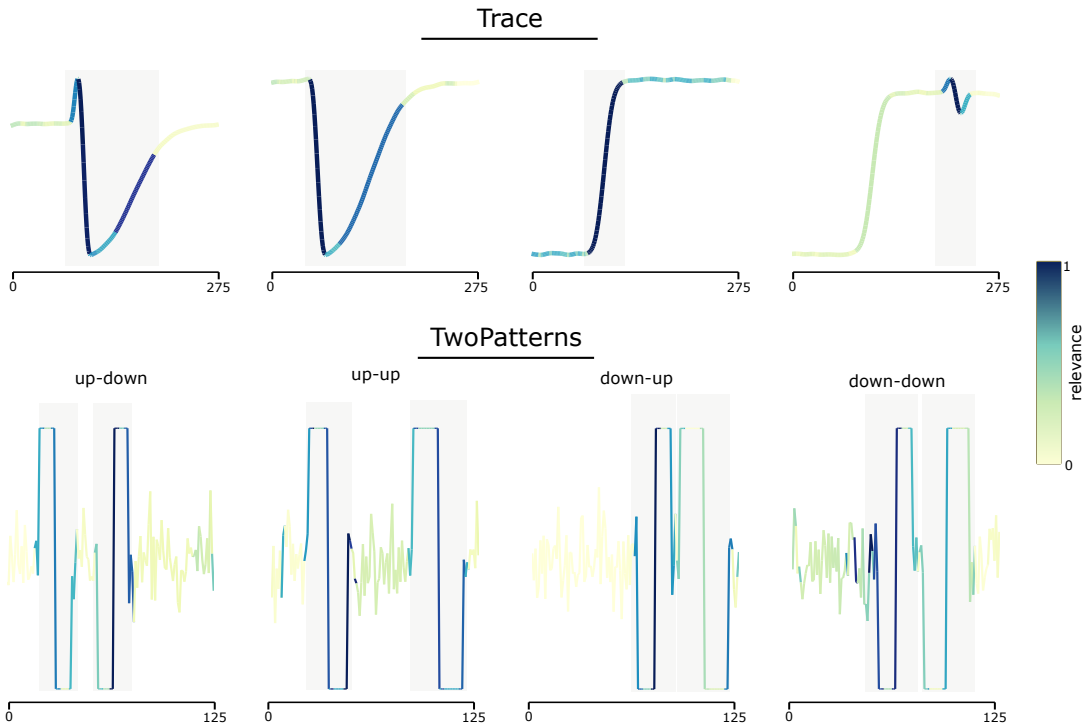


Figure 7.10: Interpretable results with highlighted shapes for the *Trace* and *TwoPatterns* datasets. The *Trace* dataset was classified with with an accuracy of 100.0%, computed with a $w_s = 25$, $thr = 0.05$ and a $1 - gram$. The *TwoPatterns* was classified with an accuracy of 100.0% with a $w_s = 10$, $thr = 0.1$ and a $5 - gram$

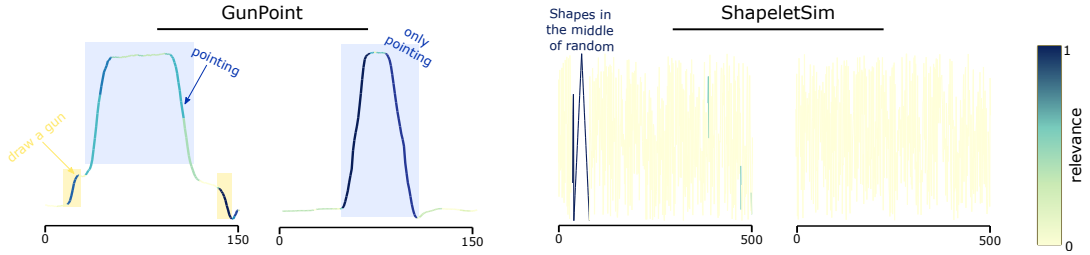


Figure 7.11: Interpretable results with highlighted shapes for the *Gunpoint* and *ShapeletSim* datasets. The GunPoint dataset was classified with with an accuracy of 99.0%, computed with a $w_s = 10$, $thr = 0.05$ and a $2 - gram$. The *ShapeletSim* was classified with an accuracy of 99.4% with a $w_s = 1$, $thr = 0.05$ and a $1 - gram$

difference between the two classes is the drawing process, highlighted by the method as the `small rise` and `small fall`.

Regarding the *ShapeletSim* dataset, the data has also two classes. In one, simple shapes, such as triangular, square, or sawtooth waves were added to a random signal. The method can highlight the representative shape of the signal.

7.3.3 Discussion of Results

In this section, we wanted to answer two main research questions: (1) can we solve time series classification tasks based on their textual representation following traditional *NLP*

methods, and (2) could we use the words that describe the time series as a readable explanation of the data and the differences between classes. For this purpose, we applied the proposed method on the *UCR* classification benchmark and highlighted several relevant use-cases and the overall results achieved.

The results provide evidence that it is possible to compare time series based on a textual representation and use traditional strategies from the text mining domain. Not only is it possible to compare the vectors of time series document with the cosine distance (as we have seen on Figure 6.8, but BoW or TF-idf models can be used with a linear SVM to solve classification tasks. The provided results can even be improved considering that the linear SVM classifier was not optimized.

Additionally, the fact that words are weighted based on their relevance to distinguish each time series document from all the others, it is possible to highlight the *subsequences* of the time series corresponding to these words and understand which parts of the signal are more relevant, being a step forward into interpretability of time series. The limited examples were still simple and intuitive and the process turns harder to understand in more complex data. In these cases, it is hard to truly find meaningful differences and explain why these classes are different, especially with words. This limits the further interpretability potential of this method because the differences might not be perceptible by the naked eye. On one end, the performance can be improved using a richer and correct set of textual descriptors. We mean richer because the differentiation is as good as the description performed, and we mean correct because the translation process from the numerical domain to the text domain is not error-proof, which means that a mistranslation may occur and make the results worse. On the other end, the words used might not be capable of expressing the differences between classes. Many cases with high accuracies can be highlighted, but their interpretability falls short. For instance, for the dataset *BeetleFly*, a binary classification problem. The method has 100% accuracy but does not show expressively the difference between classes.

The textual representation is valuable in the sense that text mining domain knowledge can be used on time series. Nevertheless, we can profit from this knowledge as much as the translation of the time series into text is rich, valuable, and domain-specific. With the current set of SSTS queries we can make a limited description that works well in simple signals described mostly by their derivative dynamics. The keywords extracted for these time series still have limited readability being even harder in the complex. However, if more expressive queries are used to make a more robust translation, the keywords extracted can be more valuable and relatable to the time series we are looking at. Therefore, a more diverse set of queries should be used to make possible the extraction of more relevant keywords. At some point, a reduction and selection of the most relevant keywords should be made, as it is performed with feature selection and dimensionality reduction.

7.4 Pattern Search with QuoTS

In this Section, we demonstrate the potential utility of QuoTS to search for relevant subsequences in real datasets. Table 6.4 from the previous Chapter highlights the keywords and operators if needed to follow these examples. This method, in contrast with SSTS, uses features and not a symbolic representation of the time series.

In order to show evidence that QuoTS is useful in a multitude of scenarios, we present several examples of its application to find relevant patterns on real domain time series. The section will start by presenting how well it matches hand gestures; then show its applicability in searching for relevant patterns in several problems and use-cases; and finally, we present the ability to include additional words based on existing patterns, with the example of *puppeteering* a toy car model.

7.4.1 *QuoTS* Matches Gestures

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

We not only demonstrate that the system can significantly distinguish gesture patterns, but it does so with very simple and intuitive queries. The ability of the written queries to match the correct class is presented both visually (Figure 7.12 and quantitatively (Table 7.6). In Figure 5, the set of eight gestures is described row-wise, having a corresponding mean shape (**MeanWave**) and a query (**Query**) that should filter it. The visual intuition is demonstrated with the barplot (**MatchScore**), which for the sake of readability, highlights the normalized match score ([0-1]) of subsequences belonging to the row-wise specific gesture. The other classes are shown in gray. For each gesture, we show the shape of all the available axis (X, Y, and Z). We attempt to create queries using only a single axis (X-axis) but used other dimensions when needed.

To quantify these results, we looked at the top-10 and top-100 matches for each class and counted how many gestures of the selected class were correctly sorted (TP/10 and TP/100). The results are presented in Table 7.6. These show that the top 10 matches always correspond to the correct class. The reader might think that the first 10 matches might be too easy considering a problem that has around 400 gesture samples per class, but note that having more gesture samples also means more opportunities to make mistakes. Moreover, considering the analogy of searching for a webpage with the *Google* search engine, a user

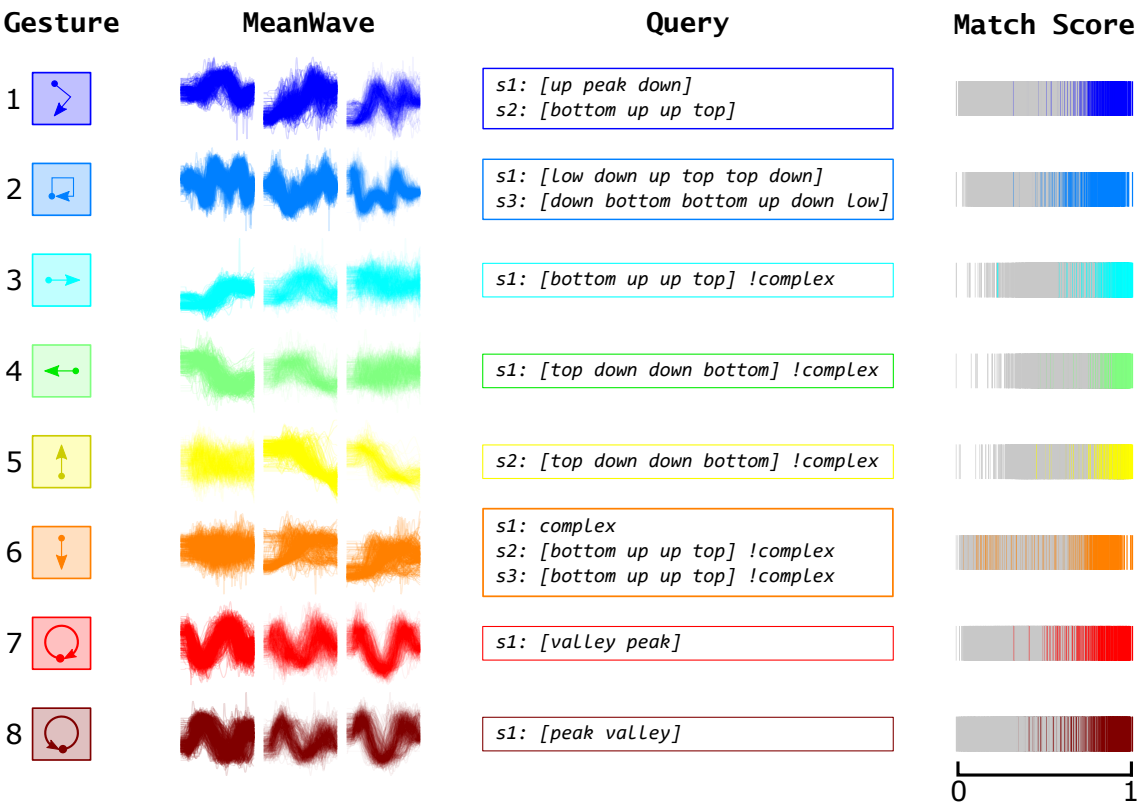


Figure 7.12: *UWaveGestureLibrary* subsequence matches with QuoTS. Column-wise are showed the gesture class, the corresponding mean wave for all subsequences, the query used to match the subsequence class and the corresponding match score for all subsequences, highlighting with the color of the specific class.

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

Table 7.6: Results for the top 10 and top 100 sorted gestures classes (G) when using the queries from Figure 7.12.

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

Although we simply wanted to demonstrate that with a set of meaningful words we can correctly sort each of the classes of this dataset, we also want to highlight that the nature of the classes can be very well expressed by the queries. This is especially evident when we look at the query for gestures that are inverse to each other, such as gestures 7 and 8. Gesture 7 is well matched by the query `[valley peak]`, implicitly, the transposed gesture should have the exact opposite query, which it indeed does (`[peak valley]`). This also occurs for the two other sets of gestures that occur in natural pair; gestures 3 (`[bottom up up top] simple`) - 4 (`[top down down bottom] simple`) and gestures 5 (`[top down`

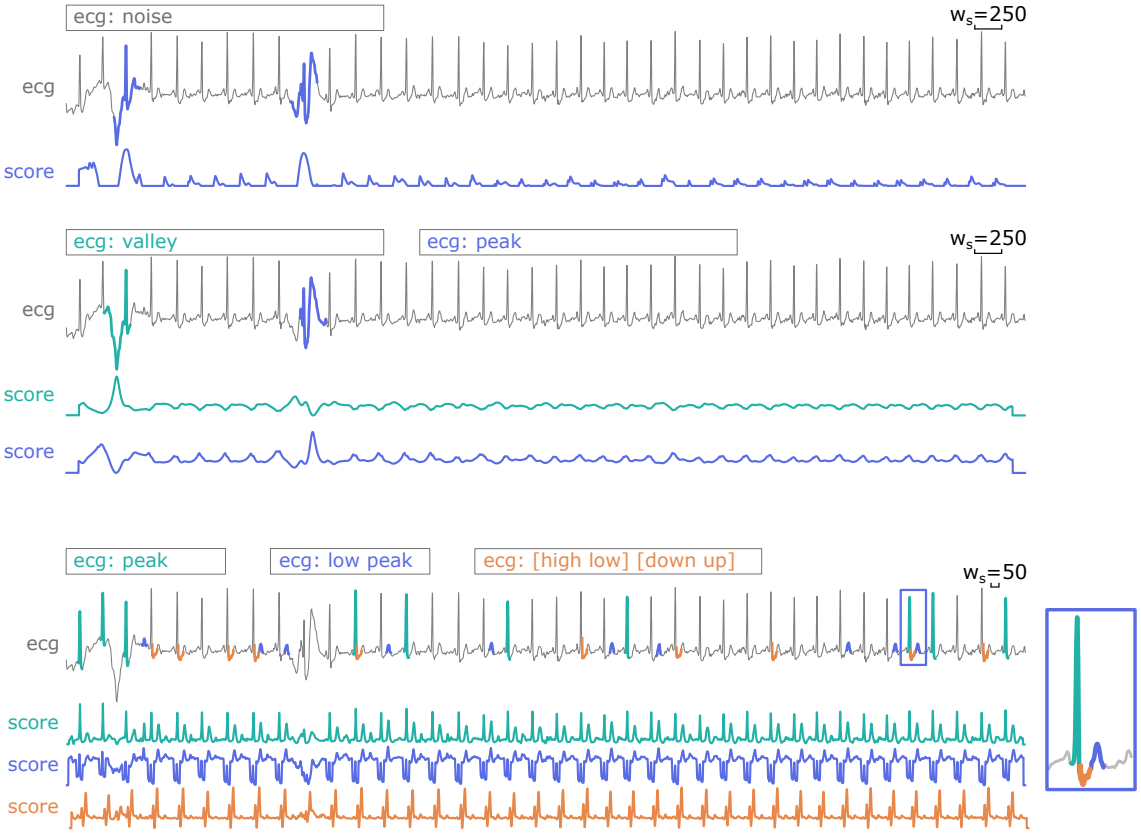


Figure 7.13: ECG use case to identify noisy sections, as well as specific segments of the ECG pattern. The queries are written in text boxes. On the side, an example of a match is showed as a larger pattern.

`down bottom]`) - 6 (`[bottom up up top]`). We also demonstrate that for the handful of cases where this did not work, there was a semantic explanation for it. (e.g. Classes 5 and 6 have not a specific pattern and seem to be especially random in their X-axis, or classes 1 and 2 are very similar, and because of that, are mislabeled). But, as our tool can perform queries in multidimensional data, we can still discriminate these by using the Y-axis in conjunction with X-axis to sort them correctly. Note that discriminating among these eight classes is not a trivial problem. Of the more than 1,000 papers to have worked with this dataset, the current best accuracy was obtained by *COTE* algorithm which achieved 76.56% using a single axis. In addition, this dataset was acquired from eight different subjects, which indicates that our system can account for the intra-subject and inter-subject variability in motion for this dataset [**uWave**].

7.4.2 *QuoTS* in Selected Use Cases

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

As a start, we were searching for the motion artifact segment that appears in two areas of the signal. This artifact can be detected in two ways, as we are showing in Figure 7.13.

The first would be to use the keyword `noise`, which searches for parts of the signal that have a higher difference from a moving average of the signal. In this case, the higher difference occurs when the motion artifacts appear. We could also search for each of these motion artifact sequences simply by pure visual intuition of their shape. For instance, the first motion artifact looks like a `valley`, while the second looks like a `peak`, which ends up matching the desired segments. This process was made by looking for subsequences with 250 samples.

On the ECG signal there are also other segments of interest, namely from the *PQRS* complex. In this case, we searched for the *R* `peak`, the *S* `valley` and the *T* `peak`. The first subsequence was simply matched by searching for the keyword `peak`, with a window size of 50 samples, which contrasts with the 250 samples from the previous example. The keyword matched well the highest peak, but other peaks were present with lower amplitudes, such as the *T* `peak`. This one was matched with the query `low peak`. In between these two subsequences, there is the *S* `valley`. In this case, we show the usage of the special `followed by` operator. The query `[high low] [down up]` indicates that the first half of the searched subsequence should have a *high change in amplitude going down* and end with a *low change in amplitude going up*. This is exactly what is matched, as highlighted and indicated by the corresponding *score* function. To be clear, the colored highlighted segments represent the 10-most relevant matches for the written query. The *scoring* functions show that the other matches would be found as well.

The ECG signal is a perfect example of how a text-based search can help find relevant occurrences, as there are specific medical conditions that are represented by variations in the original shape of the ECG signal. The next Figure (7.14) shows several examples of the detection of three different types of arrhythmia from the *St Petersburg INCART 12-lead Arrhythmia Database* (Physionet) **[PhysioNet]**.

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

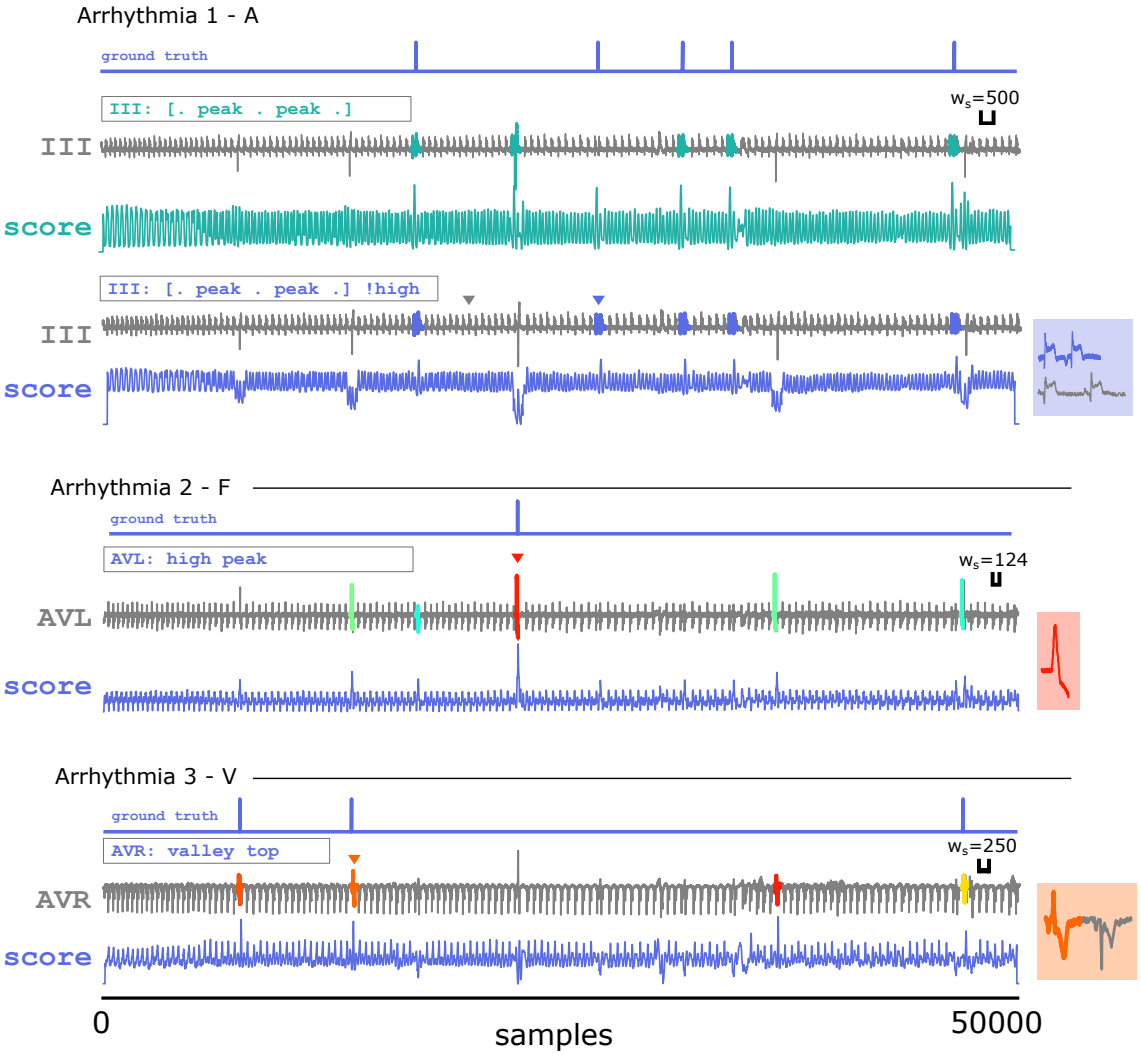


Figure 7.14: Examples of the detection of three specific cases of arrhythmia. The ground truth is selected from the annotations of specialists in the area. The queries are presented in text boxes and the found patterns are highlighted. On the side, an example of a match in a larger size is shown.

to use the other leads for the same purpose. What matters is that the desired subsequence has a particular characteristic that differentiates it from the rest.

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

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

In addition to searching for specific shapes on signals, QuoTS can be very useful for exploratory purposes in multivariate datasets. For instance, consider the following examples on a dataset acquired with the purpose to identify workload on drivers [**hci_car_dataset**]. In this experiment, drivers physiological signals, such as ECG and SCR, were monitored. Exploring this dataset with QuoTS led to discovering several interesting occurrences, as shown in Figure 7.15. In this Figure are signaled specific events with road signs. We matched these signs' positions by finding their *latitude* and *longitude* on the map and matching them with the instant in time these coordinates occur on the dataset. In addition to this information, we recall the common terms used for auto telemetry, namely: Surge - breaking (peak) and accelerating (valley); and Sway - turning right (peak) or left (valley).

The first found events were identified by searching for moments where the driver was pulling the breaks and at a similar moment in time, the driver was having an increasing value of skin conductance (indicative of higher sweat and possibly momentary stress). The query can be written as `Surge: peak high SCR: up`. We found that the presence of such events is very high on two occasions, in both cases, right before a traffic light. The second traffic light was turned red because the car stopped. This led to searching for a moment where the driver started accelerating and the skin conductance was increasing (`Surge: valley SCR: up`), which points to two main locations on the signal. The second is exactly when the car started driving after stopping at the traffic light, suggesting to be a relevant reaction from the driver. We did not have access to the videos, and can only guess what happened during the driving experience, but we can conclude that these identified moments are relevant to be observed afterward on the video, highlighting that QuoTS can be used to explore the signals very quickly, identify moments of interest (especially correlating multivariate variables) and search them on the video to validate what happened.

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

7.4.3 Matching *Known* Shapes with Words

We believe that QuoTS can be developed to be intuitive enough to allow most novice users to create simple effective queries for most information retrieval tasks. However, in some cases, the user's ability to formulate queries may be a limitation. In this section, we show that it is possible to perform a *mimicry* of the process being searched and add the query designed with the mimicry process as a *word feature vector* inside our vocabulary. We show a few examples of possible applications, starting with "puppeteering" a model car

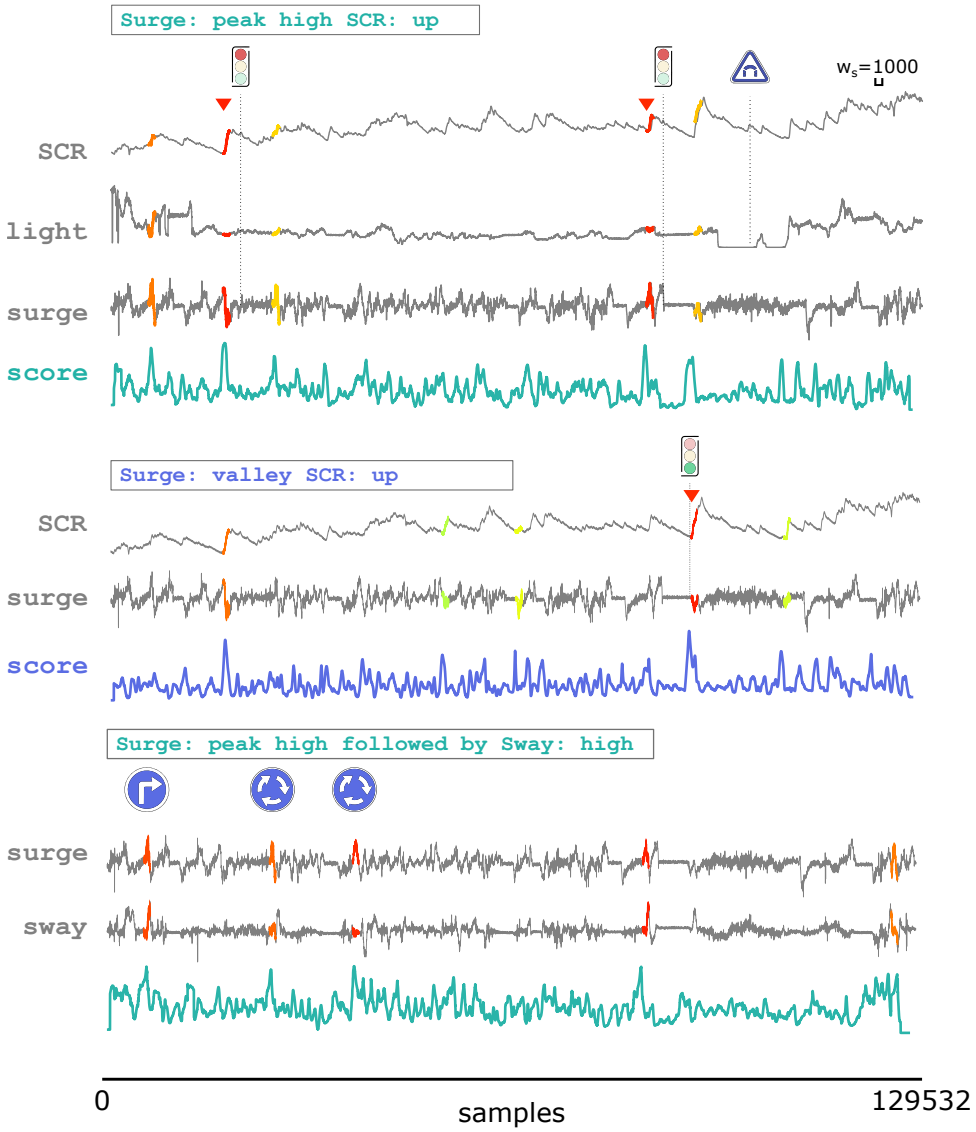


Figure 7.15: Example of multivariate patterns on a dataset from auto telemetry with physiological signals of the driver. The queries are written in text boxes. Several matches are highlighted based on the queries written. These events are associated with specific occurrences during the driving session, symbolized by traffic signs. These events were matched by searching the coordinates on the map. [**hcilab**]

in performing a 3-point-turn. First, we will show the signal where we searched this event and how we used QuoTS to find it with a text query.

Figure 7.16 is showed the search for a 3-point turn on a signal acquired with a smartphone inside a car while performing several driving exercises in a parking lot. Details of the experiment can be found at [**quots_github**]. The query used to match this event is very intuitive, being `[peak peak peak]`, which is exactly what this pattern is ( ).

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

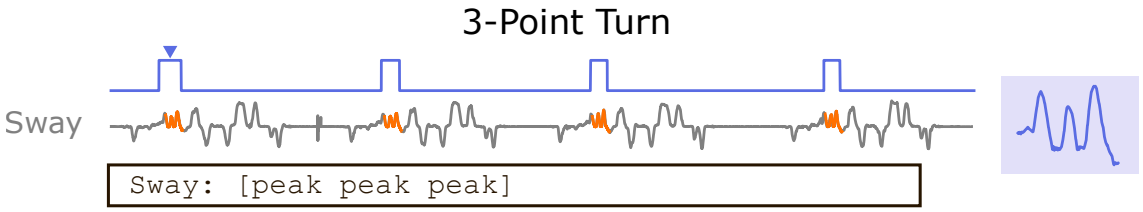


Figure 7.16: Example of searching for a 3 point turn event with QuoTS.

indicated in Figure 7.17) and acquired inertial data while mimicking a 3-point turn event on a table. The resulting shape is illustrated in *puppet data*. We believe this can be used in two ways: for the user to gain intuition as to how a motion/*manoeuver* is illustrated on motion data or how the inertial data from the motion/manoeuver can be used as a template on QuoTS:

• **Shape Intuition:** A user might not know what the shape of a 3-point turn looks like. By using a model car, he/she can mimic the motion and gain intuition over what the query should be (in this case, `[peak peak peak]`).

• **MASS template:** As mentioned above, we have a special shape word, which corresponds to a shape template given by the user, to which a word can be assigned. We then can use the word in our language to match desired patterns with the MASS distance profile as a word feature vector.

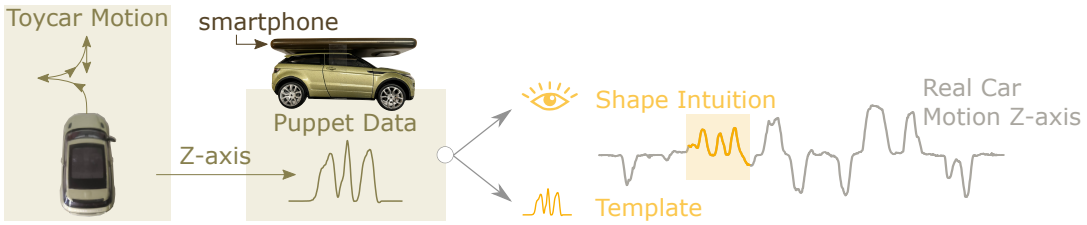


Figure 7.17: Creating query prompt data by puppeteering a car model. The puppet data for a 3-point turn (smoothed with 10 samples moving average) is presented on the left, while the real data from a real car is presented on the right.

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

7.5 Application to Occupational Scenario

8

Conclusion

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

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

8.1 Main Contributions on General Topics

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

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

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

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

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

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

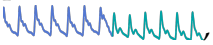
  In a standard perspective, the analyst can also perform automatic or semi-automatic segmentation and labeling/annotation of data to accelerate the preparation process to train supervised machine learning algorithms. This can either be done with the visual support of the SSM or the supporting methods.

  Regarding the process of pattern search with more expressive queries, we believe the tools provide valuable resources to both the analyst that works in the time series domain and is experienced in the computer science field (e.g. a researcher that develops machine learning methods for human activity recognition), and the analyst that is not an expert in computer science but has knowledge in a specific domain of time series (e.g. a physician that is experienced in ECG data). In this case, the process is more interactive and requires writing a regex/text query. This should be

useful to accelerate the search process by an experience computer scientist, because it should be more quick to make the search with this system than developing an algorithm for that purpose. In addition, it contributes to democratize the search mechanism to more analysts than computer scientists, because if the analyst is able to describe the shape being searched, it should be possible to find it (considering the right connotation methods/word feature vectors are used). In that aspect, the information retrieval would very quickly help the analyst in taking more informed decisions.

8.2 Scientific Contributions

8.2.1 Unveiling the *Grammar* of Time Series

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

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

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

There are still several improvements to be made, namely considering the excessive memory that is required to compute the SSM in cases where the signal is very large. The fact that a matrix has to be computed limits the ability to analyze in one run the entire signal. This process is specially relevant for periodic segmentation and computing similarity profiles. As previously explained, for novelty segmentation, the process can be

adapted to only compute the SSM along the diagonal with the size of the kernel width. An adaptation of the process has not yet been presented for the other processes, but a solution should be considered in the future. An additional limitation is the fact that the method is not invariant to trend. If events occur in slow trend changes, that is, a continuous change, the method will have more difficulty in identifying the segmentation point. The usage of pre-processing, additional time series representations, or time series decomposition methods could help in counteracting this effect. Currently, no pre-processing is made to the data. Additional methods could be performed to optimize the process, such as feature reduction and feature stacking.

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

8.2.2 Using Language for Time Series Data Mining

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

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

The concept of symbolic representation has started with SAX, which was a great inspiration for the work developed further with SSTS. We developed this method with the purpose of making pattern search with more expressive queries, more closely related with the way we look and interpret visually the existing shapes on time series. Several examples were provided showing the possibility of using regex (text patterns) on time series symbolic representation. There is still room to improvement. In one end, regex are very useful and can be used to design patterns to be searched, but in the other end, regex patterns can be very complex and limited to express the textual patterns that are being searched, and the search process is very radical and brittle, in the sense that if the time series patterns do not match the regex, no output will be given and the pattern will not be find. In this case, there is no distance measure that is continuous and this should be taken into consideration in a next iteration of the method. Additionally, having higher levels of representation could be useful to search for increasingly higher-leveled structures, such as `peaks`, `plateaus` or a combination of these. This idea was what led us perform the next method for time series classification, HeaRTS.

We imagined that if time series could be *translated* into text documents, these could be

differentiated based on the words and sentences that would represent them. Inspired by NLP techniques used for this purpose, such as BoW and TF-idf, we performed classification of time series documents, by creating a high-leveled distance measure that relies in the presence of structures, such as `peak`, `plateau`, `up`, `down` and `flat`, etc... and the order of these words in a sentence with ngrams. We then showed that it was possible to use traditional NLP methods to perform this task on the UCR classification benchmark. It was able to have a better performance than the 1-NN ED and showed to be competitive in this field. Especially because there is the possibility of extracting valuable information from the textual translation, namely by using the TF-idf weights to highlight areas of relevance on the signal or keywords that mostly represent the topic of the signal (such as topic modeling on text domain). We believe we introduced a novel idea with these processes that can be helpful for search but also for explaining which are the differences between signals. There is still a lot to improve, since as we showed, it would only be interpretable for time series with simple characteristics explained by the *connotation* and *queries* developed and used to describe the signals.

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

8.3 Other Contributions

8.3.1 Managing Rotation Plans with Exposure, Diversity and Team Homogeneity

In close collaboration with Volkswagen Autoeuropa and the Faculty of Human Motricity of Lisbon (FMH), we developed a method to automatically suggest job rotation schedules based on ergonomic standards available at the factory. These standard factors are from the AutoErgo tool, based on EAWS measures. The motivation for the development of such a tool was to help team leaders to manage job rotation schedules more quickly and

in a more informed way. Team leaders organize the working schedule for their team by assigning each worker to a sequence of workstations for the entire week, which is a time consuming tasks and not always informed in the risk level that each tasks represents for a worker. In this method, risk exposure, diversity in exposure, as well as team homogeneity, are taken into consideration when suggesting a daily rotation plan. The process was made by developing a genetic based optimization algorithm that followed an objective function developed by our team. The motivation, algorithm and results can be seen at [**jobrotation1**].

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

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

8.3.3 In using Direct Measures for Occupational Health Assessment

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

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

with different anthropometric features, which means that a specific workstation should not be weighted the same for all workers. These conclusions can be found at [**sara**]

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

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

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

8.3.4 Volatile Organic Compounds Classification

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

8.4 Scientific Production

During the period of this thesis, the work developed has been converted into research publications. In addition, several research collaborations were made that also resulted in collaborative publications. The outcomes of this work is hereby presented.

8.4.1 Journal Publications

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

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

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

8.4.2 Book Chapters

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

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

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

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

8.4.3 Conference Proceedings

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

- Alves, Rita; Rodrigues, João; Ramou, Efthymia; Palma, Susana; Roque, Ana; Gamboa, Hugo. "Classification of Volatile Compounds with Morphological Analysis of e-nose Response". Virtual, 2022.
- Mollaei, Nafiseh; Cepeda, Catia; Rodrigues, Joao; Gamboa, Hugo. "Biomedical Text Mining: Applicability of Machine Learning-based Natural Language Processing in Medical Database". Virtual, 2022.
- Rodrigues, Joao; Probst, Phillip; Gamboa, Hugo. "TSSummarize: A Visual Strategy to Summarize Biosignals". 2021.
- Santos, António; Rodrigues, João; Folgado, Duarte; Santos, Sara; Fujão, Carlos; Gamboa, Hugo. "Self-Similarity Matrix of Morphological Features for Motion Data Analysis in Manufacturing Scenarios". 2021.
- Rodrigues, Joao; Gamboa, Hugo; Kublanov, Vladimir; Dolganov, Anton. "Storage of Biomedical Signals: Comparative Review of Formats and Databases". Paper presented in Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Yekaterinburg, 2019.

8.4.4 Methods

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

- TSSummarize: Summarization of Time Series - Using the SSM to provide relevant feedback on how the time series are structured and segments are related, towards automatic and unsupervised annotation of time series.
- SSTS: Synthatic Search on Time Series - performing search on a symbolic representation of times series with regular expressions.
- HeaRTS: Human Readable Time Series - Higher level classification process of time series with visual and possible keyword feedback on data differences.
- QuoTS: *Where* on Time Series? - Text-based query search on word-feature vectors.
- Unsupervised Automatic Annotation: Using the SSM to search for segmentation points on any time series, including novelty segmentation and periodic segmentation, and automatically cluster the segments into similarity groups.

8.4.5 Projects

- Project Operator
- Project PrevoccupAI

8.4.6 Awards

8.4.6.1 Fullbright

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

8.4.6.2 Best Paper Award

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

9

Future Work

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

9.1 Overall Improvements to compute the ?? and Segmentation Process

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

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

The general idea when this method was used, was that changes on the signal should be represented by a change in the overall set of features. This leads to the idea that different

changes can be associated with a specific group of features. In the future, we should study which are the features that better describe a specific type of feature.

9.2 Unsupervised Automatic Segmentation and Labeling of Time Series

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

9.3 Hierarchical Segmentation of Time Series

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

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

9.4 Periodic Segmentation

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

9.5 Online Unsupervised Segmentation

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

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

9.6 Tool for Time Series Profiling

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

9.7 SSTS Improvements and Further Applications

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

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

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

Having ways of performing a robust search of patterns enables the usage of this kind of methods.

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

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

9.8 Further Developments for QuoTS

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

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

A

Appendix 2 - Detailed Results

A.1 Novelty Segmentation

A.1.1 UCI3

Subject	TP	FP	FN	t_scale	fs	overlap	k_scale	mph	tol
1	29	7	3	102	100	0.9	31	0.1	100
2	27	2	7	102	100	0.9	22	0.4	100
3	26	6	3	102	100	0.9	23	0.3	100
4	25	6	3	102	100	0.9	28	0.2	100
5	27	7	3	102	100	0.9	39	0.2	100
6	25	6	3	102	100	0.9	19	0.2	100
7	25	8	5	102	100	0.9	41	0.1	100
8	26	6	2	102	100	0.9	33	0.1	100
9	23	5	5	102	100	0.9	28	0.15	100
10	24	4	4	102	100	0.9	32	0.1	100
11	25	3	3	200	100	0.9	23	0.1	100
12	25	4	3	102	100	0.9	25	0.2	100
13	25	8	4	102	100	0.9	32	0.1	100
14	25	4	3	200	100	0.9	14	0.1	100
15	26	4	4	102	100	0.9	46	0.05	100
16	26	4	4	102	100	0.9	33	0.05	100
17	29	8	5	102	100	0.9	20	0.1	100
18	23	5	5	102	100	0.9	35	0.075	100
19	24	8	5	102	100	0.9	35	0.05	100
20	12	3	1	102	100	0.9	16	0.2	100
21	14	4	1	200	100	0.95	29	0.3	100
22	25	7	5	200	100	0.95	26	0.15	100
23	24	6	4	200	100	0.95	29	0.15	100

Continued on next page

Table A.0 – ***Continued from previous page***

Subject	TP	FP	FN	t_scale	fs	overlap	k_scale	mph	tol
24	25	8	5	50	100	0.9	55	0.1	100
25	24	5	4	175	100	0.95	32	0.15	100
26	25	10	4	102	100	0.9	20	0.2	100
27	24	6	4	102	100	0.9	20	0.2	100
28	23	6	5	102	100	0.9	26	0.2	100
29	26	6	2	102	100	0.9	20	0.2	100
30	23	4	5	102	100	0.9	24	0.2	100
31	26	6	2	102	100	0.9	18	0.2	100
32	25	5	3	102	100	0.9	42	0.2	100
33	25	4	4	102	100	0.9	47	0.2	100
34	23	7	5	175	100	0.85	15	0.1	100
35	22	5	6	175	100	0.85	19	0.1	100
36	25	6	3	102	100	0.9	46	0.1	100
37	23	7	5	175	100	0.9	12	0.2	100
38	21	4	8	200	100	0.95	62	0.1	100
39	25	4	4	200	100	0.95	38	0.1	100
40	24	5	4	102	100	0.9	25	0.2	100
41	22	2	6	102	100	0.9	35	0.1	100
42	22	7	6	102	100	0.9	19	0.2	100
43	27	8	3	102	100	0.9	16	0.4	100
44	22	5	6	102	100	0.75	13	0.13	100
45	26	7	2	102	100	0.85	20	0.1	100
46	23	6	7	102	100	0.85	36	0.05	100
47	25	5	3	102	100	0.9	23	0.1	100
48	18	4	10	102	100	0.9	100	0.05	100
49	23	6	5	102	100	0.9	32	0.15	100
50	21	4	7	250	100	0.9	15	0.2	100
51	23	5	5	250	100	0.9	13	0.2	100
52	23	3	5	250	100	0.9	11	0.2	100
53	23	3	7	250	100	0.9	12	0.15	100
54	23	5	7	250	100	0.9	20	0.1	100
55	23	6	5	150	100	0.9	24	0.1	100
56	21	5	9	250	100	0.9	11	0.1	100
57	23	4	5	250	100	0.9	11	0.1	100
58	21	5	7	100	100	0.75	7	0.2	100

A.1.2 UCI4

Subject	TP	FP	FN	t_scale	fs	overlap	k_scale	mph	tol
1	13	3	4	500	20	0.9	20	0.1	1000
2	17	0	0	1000	20	0.5	6	0.01	1000
3	17	1	0	500	20	0.9	90	0.05	1000
4	16	2	1	500	20	0.85	40	0.025	1000
5	14	1	3	500	20	0.75	17	0.075	1000
6	17	1	0	500	20	0.9	57	0.075	1000
7	15	2	1	500	20	0.9	56	0.075	1000
8	11	3	2	500	20	0.9	84	0.075	1000
9	16	6	0	500	20	0.9	37	0.1	1000
10	17	3	0	500	20	0.9	39	0.03	1000
11	17	3	0	500	20	0.9	38	0.03	1000
12	16	4	1	500	20	0.9	37	0.05	1000
13	17	1	0	500	20	0.9	62	0.02	1000
14	17	4	0	500	20	0.9	84	0.02	1000
15	15	1	1	500	20	0.9	32	0.02	1000
16	17	3	0	500	20	0.75	31	0.02	1000
17	16	3	1	650	20	0.9	81	0.02	1000
18	15	3	2	650	20	0.9	30	0.1	1000
19	11	3	6	650	20	0.9	17	0.1	1000
20	16	3	1	650	20	0.9	39	0.05	1000
21	13	3	4	500	20	0.9	67	0.15	1000
22	16	2	1	500	20	0.9	72	0.02	1000
23	15	0	2	500	20	0.9	72	0.05	1000
24	15	5	2	500	20	0.9	49	0.05	1000
25	17	2	0	500	20	0.9	38	0.05	1000
26	16	1	1	500	20	0.9	45	0.12	1000
27	17	2	0	500	20	0.9	53	0.07	1000
28	16	1	1	500	20	0.9	72	0.07	1000
29	17	1	0	500	20	0.9	49	0.07	1000
30	16	1	1	500	20	0.9	67	0.05	1000
31	14	1	3	500	20	0.9	71	0.1	1000
32	17	2	0	500	20	0.9	97	0.01	1000

Subject	TP	FP	FN	t_scale	fs	overlap	k_scale	mph	tol
1	21	1	3	200	200	0.5	46	0.2	750
2	21	2	3	400	200	0.75	29	0.2	750
3	22	0	1	200	200	0.75	52	0.2	750
4	22	5	2	200	200	0.75	17	0.2	750
5	21	1	3	200	200	0.75	55	0.2	750
6	22	0	2	200	200	0.75	56	0.2	750
7	22	0	2	200	200	0.75	76	0.2	750
8	21	0	3	200	200	0.75	29	0.2	750
9	26	0	2	200	200	0.75	55	0.2	750
10	22	2	2	200	200	0.9	122	0.2	750
11	22	0	2	200	200	0.75	41	0.3	750
12	22	2	2	200	200	0.75	22	0.3	750
13	23	0	1	200	200	0.75	27	0.2	750
14	22	3	2	200	200	0.75	28	0.3	750

Subject	TP	FP	FN	t_scale	fs	overlap	k_scale	mph	tol
1	13	2	0	3600	360	0.75	30	0.05	5000
2	13	2	0	4500	360	0.95	122	0.1	5000
3	12	5	1	3600	360	0.75	41	0.1	5000
4	9	7	4	2700	360	0.75	54	0.15	5000
5	13	0	0	3600	360	0.9	31	0.2	5000
6	13	0	0	4500	360	0.75	41	0.1	5000
7	13	1	0	3600	360	0.75	26	0.25	5000
8	13	1	0	3600	360	0.75	28	0.1	5000
9	12	2	0	9000	360	0.95	60	0.25	5000
10	8	5	5	9000	360	0.9	35	0.25	5000
11	13	0	0	2700	360	0.75	26	0.25	5000

A.1.3 HAR5

A.1.4 Physionet 1

A.1.5 Physionet 2

A.1.6 Kaggle

A.1.7 Alan Turing Dataset

A.2 HeaRTS Classification

Subject	TP	FP	FN	t_scale	fs	overlap	k_scale	mph	tol
1	2	0	0	750	500	0.99	69	0.3	200
2	1	1	1	375	500	0.985	82	0.55	200
3	2	0	0	500	500	0.985	63	0.52	200
4	2	0	0	375	500	0.99	92	0.85	200
5	2	0	0	256	500	0.9	21	0.5	200
6	4	0	0	256	500	0.95	30	0.5	200
7	3	0	1	512	500	0.98	38	0.45	200
8	2	0	0	375	500	0.96	34	0.45	200
9	3	1	1	750	500	0.99	36	0.6	200

Subject	TP	FP	FN	t_scale	fs	overlap	k_scale	mph	tol
1	6	1	1	260	52	0.75	50	0.25	500
2	6	2	0	260	52	0.75	45	0.25	500
3	7	0	0	260	52	0.75	39	0.25	500
4	6	1	1	260	52	0.75	19	0.3	500
5	3	0	2	260	52	0.75	36	0.3	500
6	7	1	1	260	52	0.75	14	0.55	500
7	7	1	1	260	52	0.75	10	0.4	500
8	8	1	0	260	52	0.85	82	0.13	500
9	7	2	1	260	52	0.75	20	0.25	500
10	5	2	3	260	52	0.75	10	0.5	500
11	8	1	0	260	52	0.75	49	0.25	500
12	7	1	0	260	52	0.75	53	0.095	500
13	4	1	2	260	52	0.75	23	0.5	500
14	6	0	1	260	52	0.75	53	0.3	500
15	8	2	0	260	52	0.75	23	0.3	500

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.