Information Processing and Management 56 (2019) 61–76

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

SSTS: A syntactic tool for pattern search on time series

João Rodrigues[a,*], Duarte Folgado[b], David Belo[a], Hugo Gamboa[a,*]

[a] Laboratório de Instrumentação, Engenharia Biomédica e Física da Radiação (LIBPhys-UNL), Departamento de Física, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Monte da Caparica, Caparica 2892-516, Portugal
[b] Associação Fraunhofer Portugal Research, Rua Alfredo Allen 455/461, Porto 4200-135, Portugal

A R T I C L E I N F O

Keywords:
Meta symbolic language
Time series
Query search
Regular expression
Signal processing
Grammar

A B S T R A C T

Nowadays, data scientists are capable of manipulating and extracting complex information from time series data, given the current diversity of tools at their disposal. However, the plethora of tools that target data exploration and pattern search may require an extensive amount of time to develop methods that correspond to the data scientist's reasoning, in order to solve their queries. The development of new methods, tightly related with the reasoning and visual analysis of time series data, is of great relevance to improving complexity and productivity of pattern and query search tasks. In this work, we propose a novel tool, capable of exploring time series data for pattern and query search tasks in a set of 3 symbolic steps: Pre-Processing, Symbolic Connotation and Search. The framework is called SSTS (Symbolic Search in Time Series) and uses regular expression queries to search the desired patterns in a symbolic representation of the signal. By adopting a set of symbolic methods, this approach has the purpose of increasing the expressiveness in solving standard pattern and query tasks, enabling the creation of queries more closely related to the reasoning and visual analysis of the signal. We demonstrate the tool's effectiveness by presenting 9 examples with several types of queries on time series. The SSTS queries were compared with standard code developed in Python, in terms of cognitive effort, vocabulary required, code length, volume, interpretation and difficulty metrics based on the Halstead complexity measures. The results demonstrate that this methodology is a valid approach and delivers a new abstraction layer on data analysis of time series.

1. Introduction

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, 1985). 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 (Neisser, 2014). This capacity is also revealed in the search for patterns in time series, as data scientists are able to visually understand where these patterns occur with previous training. For example, considering the electrocardiogram (ECG) depicted in Fig. 1, one might need to find the big narrow peak (*R* peak), or the peak that comes afterwards (*T* peak). This task is easily described visually, but can turn into a tedious, repetitive or even hard task in order to computationally find a solution and search for the desired pattern.

Although the human decision is affected by external and internal biases there is the possibility of profiting from this capability of recognizing patterns, which could be useful and quicken the resolution of some tasks. Based on these assumptions, we propose a novel

* Corresponding authors.
E-mail address: h.gamboa@fct.unl.pt (H. Gamboa).

https://doi.org/10.1016/j.ipm.2018.09.001
Received 29 March 2018; Received in revised form 2 August 2018; Accepted 5 September 2018
Available online 29 September 2018

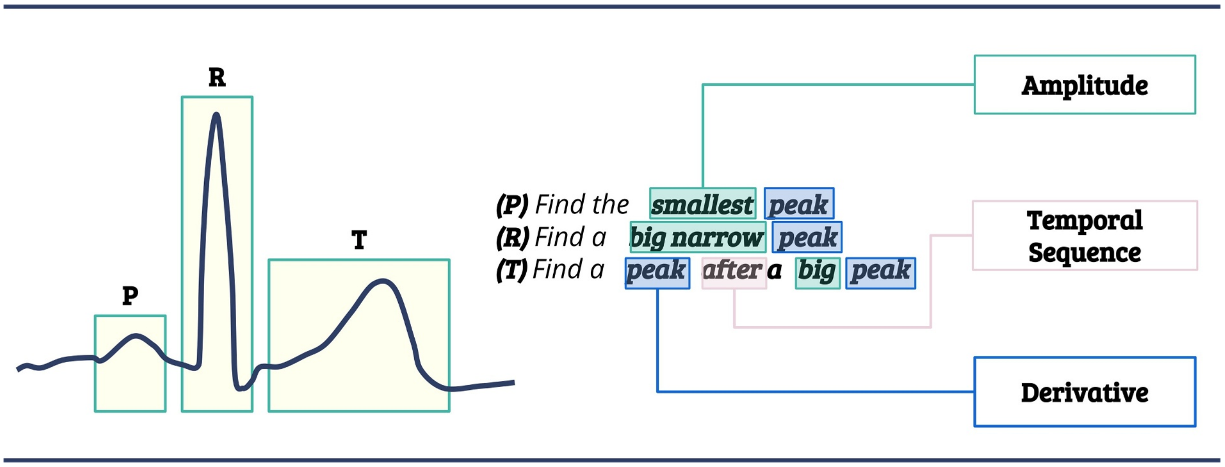

R
T
P
(P) Find the smallest peak
(R) Find a big narrow peak
(T) Find a peak after a big peak
Amplitude
Temporal Sequence
Derivative

Fig. 1. The detection of events in time series is easy to describe with language. This illustration depicts the description of event detection in an ECG signal and how the meaning of some words is related to the characteristics of the signal.

tool entitled Syntactic Search in Time Series (SSTS) that reduces the gap between the reasoning behind the visual analysis of the time series and the execution of the search for a specific pattern, by means of a syntactic approach. The proposed method consists of three symbolic steps: (1) pre-processing, in which standard signal processing methods are applied; (2) symbolic connotation, where time series are converted into a symbolic representation and (3) search, in order to find the intended pattern on the time series.

An interactive live version of SSTS and public available source code is available at https://novabiosignals.github.io/SyntacticSearchonTimeSeries/.

1.1. Motivation

Computational methods are necessary for time consuming data driven analysis, but the first layer of knowledge comes from the human mind and not from machines. Regarding this context, interactive data analysis is a great challenge as there is a lack of integrated interactive systems, tools and methods for fast, useful and meaningful information retrieval from time series data based on observational skills (Holzinger, 2013; Holzinger & Pasi, 2015; Holzinger & Zupan, 2013; Machado, Gomes, Gamboa, Paixão, & Costa, 2015). Based on this rationale, the proposed approach is inspired by the ideas that gave birth to computer languages. In order to present a well-founded inspiration, we present below historical references that are the base motivation of our work. The key ideas of inspiration were delivered by Iverson, with regards to the importance of notation, nomenclature and language as tools of thought (Iverson, 1980), that have materialized in the development of the APL language. These ideas are the root for modern vector-based languages, such as Matlab (Moler, 2006), and continue to inspire the creation of new computational methods, both expressive and compact, in several current science domains. Some examples can be found in the "*Journal of J*", where the *J* language (a synthetic version of APL language) is used in scientific problems (Saurer, 2017). In the context of our research, regular expressions are used as the tool of thought for pattern search in time series.

Regular expressions are based on regular rules created by Kleene in order to describe a model of neural net represented as a finite state machine (Kleene, 1956). Having later been used as a text parser by Thompson (Thompson, 1968), regular rules are able to describe a pattern as a sequence of characters.

Similarly to text, time series are carriers of information. Time series comprise sequences of ordered real domain, multi-dimensional, numerical data observed during a given temporal interval, which are typically plotted as variations in amplitude. In terms of morphology, many attributes can be extracted from the visual perception of the signal, such as rising and falling slopes, concavity, direction, amplitude thresholding, frequency, time and amplitude range of a slope, among others. For instance, we can identify positive peaks by finding a rising slope followed by a falling slope.

The merging of all these concepts gave rise to the idea of using a symbolic representation to characterize sequence of states of time series in multiple attributes. The combination of these attributes into sequence of primitives is a symbolic characterization of the signal that enables the use of regular expressions for searching the desired pattern. The proposed tool focuses on the knowledge retrieved from the visual interpretation of the signal that enables the search of the desired patterns by writing a regular expression that parses the syntactic representation of the signal. This representation is arranged by a specific set of grammatical rules that organize the samples of the signal into a meaningful sequence of characters, based on the most relevant morphological properties encountered. Besides, the processes involved in the first two steps of the method are based on the symbolic approach led by Iverson.

1.2. Research objectives

The focus of the current study involves the development of a tool called SSTS for time series data exploration and knowledge retrieval regarding query and pattern search tasks. The major objectives of this approach are listed as follows:

- Provide data scientists with more expressive and interactive methods that rely on a morphological thinking to retrieve knowledge from data.
- Increase the efficiency/algorithm's complexity ratio in generating a query search.
- Reduce the cognitive expressiveness to design the search method.
- Increase the abstraction over the entire solving task by means of a symbolic representation of time series.
- Turn the query search task faster.

This section presents the motivation of our method, inspired on historical research works. In the current state of the tool, its use requires knowledge on signal processing techniques and regular expressions. As a long-term goal, we expect to provide abstraction mechanisms to simplify the need of signal processing tasks and regular expressions knowledge.

In Section 2, related work concerning the foundation of syntactic analysis of numerical data and recent approaches on the symbolic representation of time series will be presented. Section 3 addresses a thorough explanation of the methods used in each step of the SSTS. Furthermore, in Section 4, we present a complete description of the dataset, and examples in which we apply the proposed tool. Section 5 is divided into two parts: (1) a detailed explanation of how SSTS is used to solve the proposed examples, and (2) an evaluation of SSTS performance using the Halstead complexity measures (Kayam, Fuwa, Kunimune, Hashimoto, & Asano, 2016). The evaluation is based on the comparison between SSTS and a regular scientific routine written as a Python script. It is important to note that Python has been chosen since it is one of the most high-level, expressive and verbose programming languages, in comparison to others like C, or Java that are low-level scripts, and are characterized for increasing the length and difficulty in generating a functional script. Therefore, using Python as the reference is a good measure for evaluating how intuitive can be the use of this tool. These results will be discussed in the Section 6. Future developments and applications of this tool will be shared along Section 7.

2. Background

2.1. Historical overview

In the field of signal processing there are few algorithms that rely on syntactic models to solve query search tasks. In the late 60s and 70s, syntactic approaches in time series started to appear, but fell into disuse over time, having recently returned to the picture. Some application of these linguistic models into mathematical domains have been led by Pavlidis (1971, 1973) and Fu (1971), who have an exhaustive list of studies in pattern recognition and computer vision fields related with the symbolic description of 2D-shapes. Moreover, Pavlidis expressed that the primary problem in the implementation of such methods with functions of one variable would involve the representation of the waveform into a one-dimensional string of primitives over a finite alphabet (Pavlidis, 1971).

These concepts inspired multiple works that have emerged and applied a syntactic approach to pattern recognition tasks. Some examples can be addressed in the field of ECG processing (Coutinho, Fred, & Figueiredo, 2010; Horowitz, 1975; Trahanias & Skordalakis, 1990; Udupa & Murthy, 1980). Horowitz et al. describe that the detection of peaks in ECG signals can be achieved by specifying a set of primitives (one symbol or a group of symbols that represent one sample of the signal) able to be representative of the shape of the ECG peaks. The generated string is made by combining the information of the amplitude and the first derivative, which is thereafter parsed, based on a set of context free grammar rules. This enables the detection of ECG peaks. Similar approaches were applied in the works of Trahanias and Skordalakis (1990) and Udupa and Murthy (1980), although in these cases the set of primitives is coded with the length and slope of the line segment. The use of symbolic representations has also been found in biometrics using the ECG by means of compression techniques (Coutinho et al., 2010).

2.2. Modern approaches on symbolic representation

Recently, approaches based on the symbolic representation of time series have reappeared and provided satisfactory and competitive results in comparison to statistical methods, proving that the use of small sets of primitives and grammatical rules are a possible way of describing large sets of patterns by efficiently representing the signal's structures into an ordered hierarchical sequence of characters (Hamdi, Ben Abdallah, & Bedoui, 2017).

In the last decade, several works that used a symbolic approximation of time-series were published. Some of these modelled a finite state automaton from the generation of symbolic time series, being able to describe the probabilistic flow of data and, with this, detect patterns on time series (Piccardi, 2006; Rajagopalan, Ray, Samsi, & Mayer, 2007). Both Lin et al., with the Symbolic Aggregate Approximation (SAX), and Senin et al. used grammar-based compression methods to detect anomalies by comparing subsequences of the signal (Lin, Keogh, Wei, & Lonardi, 2007; Senin et al., 2015). Another similar approach uses a symbolic representation of time series for information retrieval (Li & Ray, 2017).

In Ordóñez, Armstrong, Oates, and Fackler (2011), *Bag-of-words* models are used on a symbolic representation of time series to classify physiological data. Additionally, several works regarding symbolic time series analysis were also found for human gait dynamics analysis (Abbasi & Loun, 2014; Yu et al., 2017) and electroencephalographic epileptic seizures and brain dynamics (Hussain et al., 2017). In Tirabassi and Masoller (2018) the symbolic representation of time series is used to uncover regions that

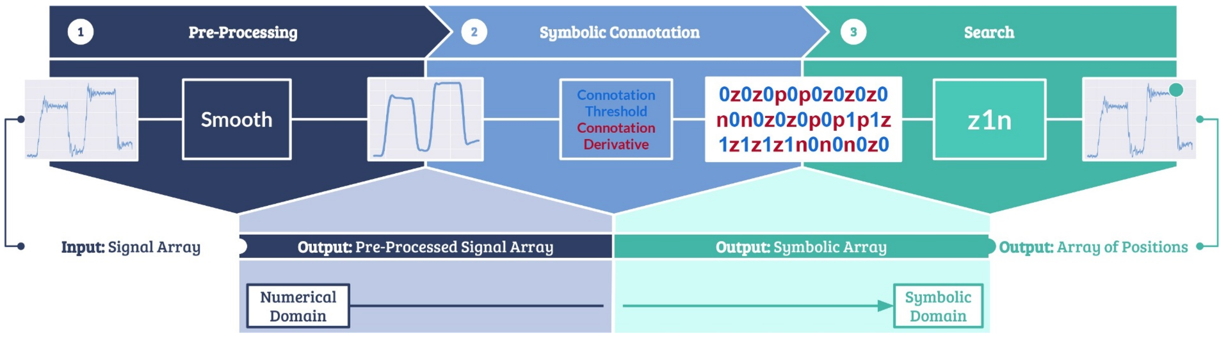


Fig. 2. SSTS modular architecture: the proposed system is divided into three main modules - pre-processing, symbolic connotation and search.

share similar climate patterns, by means of transition probabilities over the symbolic sequence. Finally, Hamdi et al. used a novel approach that applies a Deterministic Finite Automaton (DFA) and regular grammar rules for real time detection of the ECG's QRS complex (Hamdi et al., 2017).

Inspired by the aforementioned syntactic approaches for pattern recognition and by the importance of developing more interactive tools for time series analysis, we developed a methodology that comprises three major steps, which are in line with the ones typically found in the literature: pre-process the time series to return a more appropriate signal; 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 a parser to search for patterns in the symbolic representation.

This work, unlike the previously reviewed approaches, is generic and can be applied to any type of time series, resting in the assumptions that the user has a good knowledge of the signal properties, and what is the structure being searched. This new methodology appears to be quite effective for exploratory contexts, however, it might potentially lead to overcomplicated solutions for complex problems, which must be taken into account for users with limited background knowledge on signal processing and regular expressions.

3. Methods

In this section we will introduce the proposed architecture, which is depicted in Fig. 2 and is divided into three modules, each associated with a well-designed purpose: (1) pre-processing, (2) symbolic connotation and (3) search. This allows to achieve modularity, in which the user can reuse and optimize the methods independently, without loosing cohesion in the overall system. We will also introduce the tokens for the signal pre-processing and symbolic connotation phases of the SSTS. This mechanism is inspired by and has similarities to the APL language, although in our case, the list of tokens is defined by the user as presented in Sections 3.1 and 3.2. The idea behind using tokens is that the reasoning process can be better aligned with the expressions by using or creating a set of visual shorthand for the syntactic search process. This symbolic mechanism of defining a set of tokens is not mandatory, so the process behind the tokens can be made with standard function names as well.

The first part of this section will thoroughly present each step of the SSTS tool, while the second will use a practical example to demonstrate how each of the steps work.

3.1. *Pre-processing*

The pre-processing stage is responsible for preparing the signal by removing noise that arises from several sources or adjusting the signal so that the information is unveiled. Noise can appear from the dynamics of the environment where the data is sampled, it can be related to the sensor mechanics, or even consists of artefacts acquired during the acquisition process. The application of a set of pre-processing methodologies improves the signal to noise ratio and adjusts the signal for the extraction of knowledge and information from the raw time series. Typically, in signal processing tasks, a pipeline of linear filters, moving window average filters and statistical de-noising or re-sampling techniques are a set of the most used procedures to prepare the signal for further tasks (Downey, 2016).

The current approach uses a symbolic representation of these techniques represented by a 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 considering that the token precedes the corresponding argument(s), and each element is separated by a white-space character. Let $s := (s_1, s_2, \ldots, s_n)$ represent a time series of length N and $s_n \in \mathbb{R}$. Table 1 summarizes a list of pre-processing operators and associated tokens.

3.2. *Symbolic connotation*

Semiotics describes that the connotative aspect of an image or a sign, corresponds to the extraction of meaning (sometimes called the *second meaning*), by means of the personal interpretation of its traits and characteristics (Chandler, 2017). This step is the

Table 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	Token	Input parameters	Description
HP	☱	(*fc*)	Linear high-pass filter with cut-off frequency *fc*
LP	☴	(*fc*)	Linear low-pass filter with cut-off frequency *fc*
BP	☲	(*fc1, fc2*)	Linear band-pass filter with cut-off frequencies *fc1* and *fc2*
Smooth	~	(*win_size*)	Smoothing of *s* by a moving average window filtering technique of size *win_size*
Wthng	⊙	(s)	Whitening of s: $s_w = \frac{s - s_m}{\theta(s)}$, being: s_w the signal whitened, s_m the signal mean and $\theta(s)$ the standard deviation of the signal
Abs	‖	none	Modulus of s: $\|s\|$
norm	⊓	none	Magnitude: $\|s\| = \sqrt{s_0^2 + s_1^2 + s_2^2}$
N.A.	|	N.A.	Separates the pre-processing methods applied to multiple signals or multiple processing of the same signal

connotative aspect of time series, in which the interpreter has made his personal analysis of the signal and retrieved the necessary information in order to search for the desired pattern.

The symbolic connotation step generates a sequence of symbols by extracting properties of the signal that are based on a conversion rule defined by the user. Each of these conversion rules is designated by connotation methods and are responsible for translating each sample of the signal into a character that represents the state of the sample for that conversion rule. Each of the connotation methods is related with specific attributes of the time series that are considered relevant for the search procedure. The string that results from this step is, therefore, a symbolic representation of the strictly necessary attributes of the signal that are best suited to specify a pattern match.

The string can be based on a single connotation method or the combination of multiple ones. Additionally, if the resolution of the problem involves the use of multiple signals, the string can be a combination of connotation methods of multiple signals as well. Each of these is associated with specific symbols, just as the pre-processing methods.

Typically, in this step, each sample ($s_0 \ldots s_n$) of the time series is converted into a primitive ($P_0 \ldots P_n$), which is the sequence of applied connotation methods ($C_0 \ldots C_m$). For each connotation string, a specific grammar is created with a set of *terminals, non-terminals, starting symbol* and *production rules*. These concepts can be demonstrated with a context-free grammar formalism (Reghizzi, Breveglieri, & Morzenti, 2013), which depends on the choice of the connotation method(s) adopted by the user.

First, the general definition of the variables of the formalism is presented as follows:

Dictionary (Σ)
Entire set of symbols that were chosen by the user for all the connotation methods;

Group$_\mathbf{m}$
Set of symbols from the dictionary associated with a specific connotation method;

Symbol$_\mathbf{c}$
Symbol used for a given connotation method;

Signal$_\mathbf{n}$
Set of sequence of symbols associated with the corresponding signal;

Connotation
Final string representation.

The simplest case would involve the use of a single connotation method applied to a given signal. In this specific case, only one group of symbols would exist, as the dictionary would only comprise symbols of one connotation method.

$$\begin{aligned} \langle Symbol_c \rangle &\models \langle Group \rangle \\ \langle \text{Connotation} \rangle &\models \langle Symbol_c \rangle \langle \text{Connotation} \rangle \mid \langle Symbol_c \rangle \end{aligned}$$

Based on this formalism, the connotation string is generated by one or more symbols from the group of the specific connotation method used. The same reasoning can now be made to multiple connotation methods applied to one signal.

In this case, the number of groups will be the same as the number of connotation methods applied. The connotation string is, therefore, the sequence of primitives generated by the ordered sequence of *m* symbols that can be any of the *c* symbols from one of the existing groups.

$$\begin{aligned} \langle \text{Symbol}_{\text{m},C_m} \rangle &\models \langle Group_m \rangle \\ \langle Primitive \rangle &\models \langle Symbol_{1,C_1} \rangle \langle Symbol_{2,C_2} \rangle \langle \ldots \rangle \langle Symbol_{m,C_m} \rangle \\ \langle \text{Connotation} \rangle &\models \langle \text{Primitive} \rangle \langle \text{Connotation} \rangle \mid \langle \text{Primitive} \rangle \end{aligned}$$

Table 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	Token	Input parameters	Group of symbols	Description
ampComp	⇞	(*s, thr*)	["1","0"]	Amplitude comparison, If $s_i > thr \times (s_{max} - s_{min})$, s_i is "1", else s_i is "0"
diffComp	∂	(*s, thr*)	["p", "n", "z"]	Derivative of signal *s*. If $s_i^{'} > thr$, $s_i^{'}$ is "p"; elif $s_i^{'} < - thr$, $s_i^{'}$ is "n"; else, $s_i^{'}$ is "z".
diff2Comp	∂:	(*s, thr*)	["p", "n", "z"]	Second derivative of signal *s*. If $s_i^{''} > thr$, $s_i^{''}$ is "p"; elif $s_i^{''} < - thr$, $s_i^{''}$ is "n"; else, $s_i^{''}$ is "z".
ampDiffComp	↕	(*s, thr*)	["1", "0"]	Determinates if amplitude from a minimum to a maximum and vice-versa is greater than *thr*. If so, s_i is "1", else, s_i is "0".

The above formalism includes the application of the connotation methods to multiple signals as well. This is made by separating each sequence of connotation methods with a vertical bar, and sort this sequence by the signal which it corresponds to. The primitive composition depends on the number of the overall connotation methods used and not on the number of signals.

Finally, this tool can also be used with multiple signals. The above formalism is enough to describe multiple signals sources, since the number of groups is made by the count of connotation methods applied and not the number of signals. Therefore, if multiple signals are used, at least one connotation method will be applied to each one, each being assigned a group of symbols.

The set of connotation methods can be defined by the user and added as needed. A list of base connotation methods and the corresponding symbols used for the examples presented in this study are listed in Table 2.

3.3. Search procedure

The last step of the process involves searching for the desired pattern in the generated string with a regular expression. This string is governed by the rules of regular expressions from the *alternative regular expression Python module* (P. S. Foundation, 2017b) and benefits from all the functionalities and meta-characters typically used with this tool.

The search procedure returns the intervals at which positive matches occurred. In this particular work, it has been decided to use a regular expression, although any other parser, conveniently combined with the previous steps, might be used for the same purpose.

In order to understand the way that regular expressions work, some of the more often used characters are presented as follows (Friedl, 2006):

- * - The preceding item will be matched zero or more times;
- + - The preceding item will be matched one or more times;
- ? - The preceding item is optional and will be matched, at most, once;
- . - Matches any character;
- |, & - Boolean operators - or, and;
- (? = <) - Positive lookbehind - The string matches the item that is preceded by the pattern inside the lookbehind without making it part of the match;
- (? < !) - Negative lookbehind - The string matches the item that is not preceded by the pattern inside the lookbehind;
- (? =) - Positive lookahead - The string matches the preceding item that is followed by the pattern inside the lookahead without making it part of the match;
- (?!) - Negative lookahead - The string matches the preceding item that is not followed by the pattern inside the lookahead.

3.4. Example 1 – Start/end of plateau

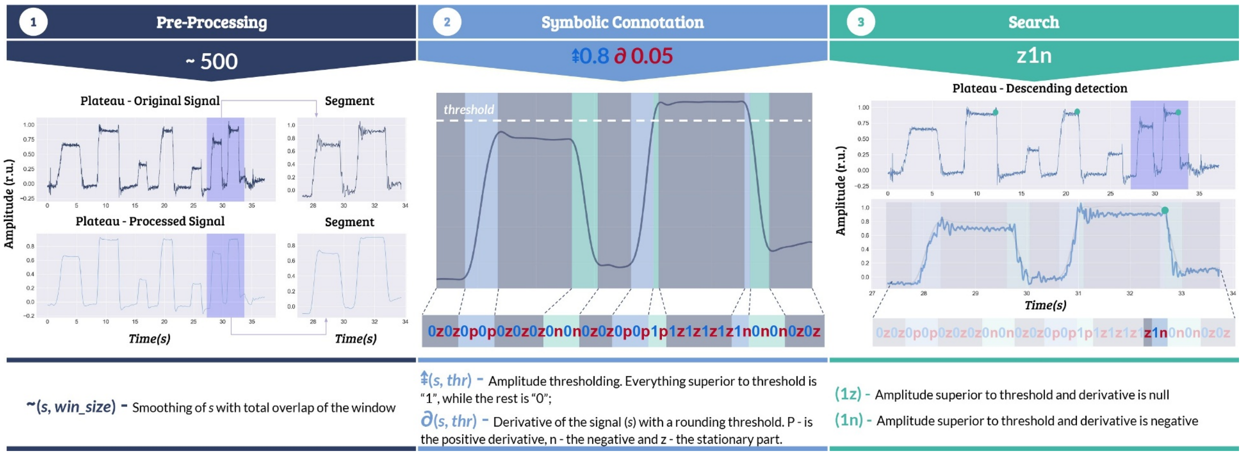
In order to elucidate more deeply how the SSTS works, an example will be given explaining each step carefully with an illustration of the process. The purpose of this example is to find the time intervals where the plateaus above a certain threshold begin to decrease. The represented signal is the *Z*-axis data of an accelerometer sensor placed on the wrist of a subject while performing a rotating task on different angles. Each resolution step of this example is shown in Fig. 3.

3.4.1. Pre-processing

In the example of Fig. 3, the pre-processing step uses the following string: "∼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 below the original one. The area delimited in the plots represents the segment of the signal that will be represented in Section 3.2.

3.4.2. Symbolic connotation

Fig. 4 demonstrates the symbolic connotation formalism described in Section 3.2. The processed signal ($s_1 \ldots s_n$) is decomposed in a symbolic sequence by two connotation methods: amplitude (blue) and derivative (red). The first implies that the sample values greater than 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 a primitive ($P_1 \ldots P_n$), which is the alternate association of the two connotation methods. The approximate result is illustrated by the



1 Pre-Processing
~ 500
2 Symbolic Connotation
⇞0.8 ∂ 0.05
3 Search
z1n
Plateau - Original Signal
Segment
Plateau - Processed Signal
Amplitude (r.u.)
Time(s)
threshold
0z0z0p0p0z0z0z0n0n0z0z0p0p1p1z1z1z1z1n0n0n0z0z
Plateau - Descending detection
~(s, win_size) - Smoothing of s with total overlap of the window
⇞(s, thr) - Amplitude thresholding. Everything superior to threshold is "1", while the rest is "0";
∂(s, thr) - Derivative of the signal (s) with a rounding threshold. P - is the positive derivative, n - the negative and z - the stationary part.
(1z) - Amplitude superior to threshold and derivative is null
(1n) - Amplitude superior to threshold and derivative is negative

Fig. 3. An example of the SSTS flow: the input signal is pre-processed using a moving average window, a symbolic connotation is applied based on the signal morphology, and a search is performed to retrieve the intended pattern.

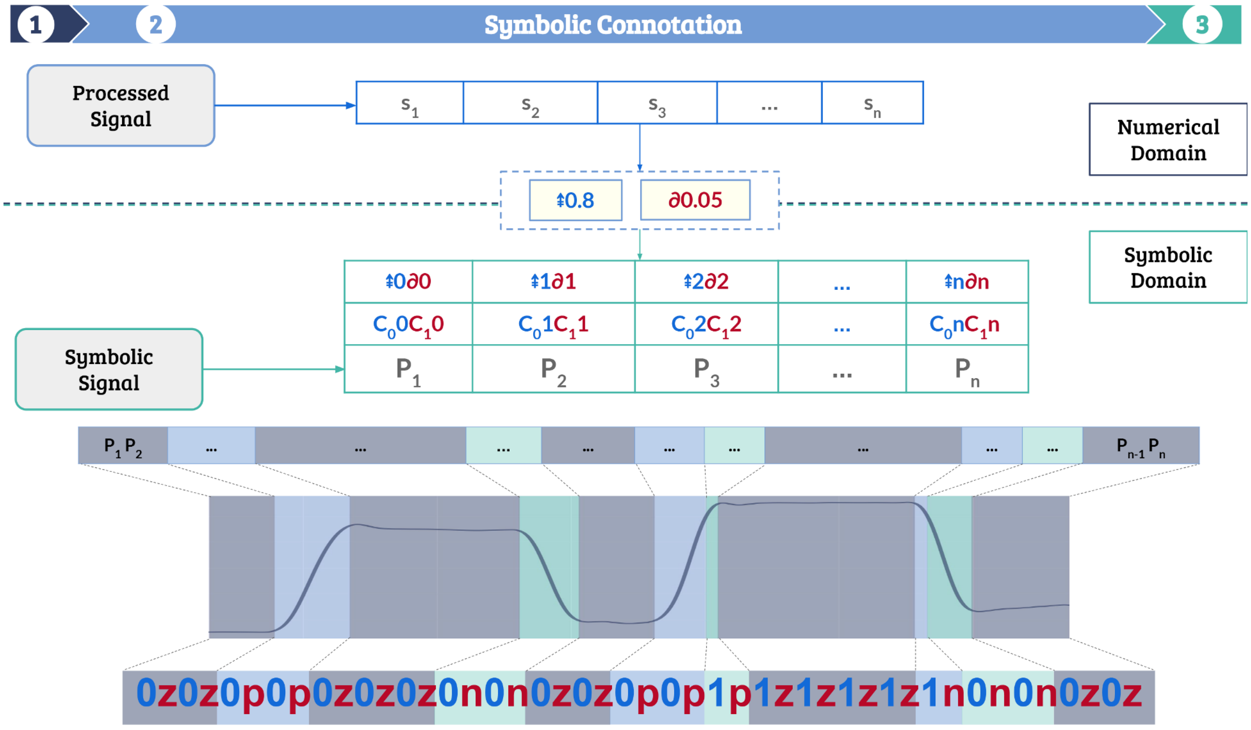


Fig. 4. Example 1 – The signal that results from the pre-processing step is decomposed into a sequence of primitives (P). Two connotation methods (C0 and C1) are used: The amplitude method (in blue) and the derivative method (in red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

bottom string, in which can be seen the alternation property and what it translates. The representation made by means of these two methods is expected to be necessary to ease the search procedure.

3.4.3. Search

In the example of Fig. 3, the purpose is to determine when a plateau greater than to 80% of the amplitude range of the signal starts to fall, which based on the string representation of the second step is indicated by a 1 and a transition from stationary (*z*) to falling (*n*). The regular expression used for the search is precisely *z*1*n*, which indicates that the signal, at some point greater than the threshold, will start to decrease. Fig. 5 presents the output of the matching by highlighting the interval in time that corresponds to the positive match of the regular expression. The following sections will provide additional examples, based on challenges arising from real data, in which a thoroughly explanation will be provided to fully understand the capabilities of the SSTS tool.

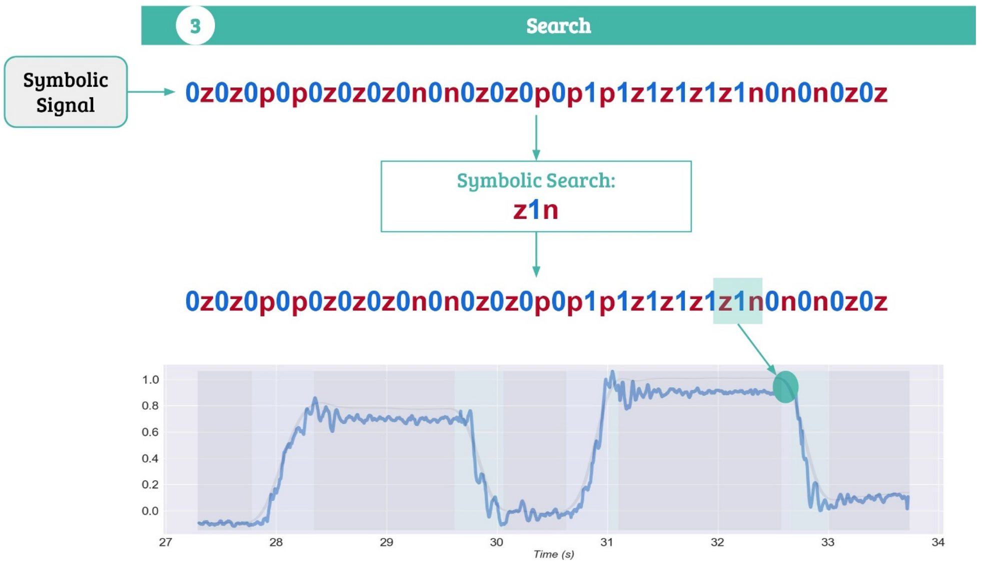

3
Search
Symbolic Signal
0z0z0p0p0z0z0z0n0n0z0z0p0p1p1z1z1z1z1n0n0n0z0z
Symbolic Search:
z1n
0z0z0p0p0z0z0z0n0n0z0z0p0p1p1z1z1z1z1n0n0n0z0z
Time (s)

Fig. 5. Example 1 – search step. The string is parsed by a regular expression to search for the desired pattern or event.

4. Dataset and examples description

In this section, several use cases will be presented to demonstrate the applicability and relevance of our approach in real time series data. These cases are a set of simple examples that attempt to core several standard problems encountered in signal processing and pattern search tasks and will be used to evaluate how the search procedure is performed. Furthermore, it will allow establishing a comparison between the syntactic resolution with the traditional method of solving these type of problems in computer science, which we designate as classical resolution. The comparison will involve the legibility and difficulty in generating a solution to the corresponding example.

Example 1 – Start and end of plateau

- Signal type: *Z*-axis of accelerometer data;
- Description: The accelerometer sensor was positioned on the wrist. The data has been acquired with a sampling frequency of 1000 Hz, while the subject performed multiple levels of rotation of the wrist (45, 60 and 90°). In each orientation, the subject has sustained the position for more than two seconds;
- Goal: Detect the beginning and ending of a plateau that corresponds to a rotation above 60°;
- The data for this example has been collected by the authors.

Example 2 – Step detection

- Signal type: Magnitude of accelerometer data;
- Description: Data collected at a sampling rate of 100 Hz, from a sensor located on the right pocket of a subject while performing sets of straight normal walk;
- Goal: Detect the instants in time when a right or left heel floor contact is achieved during normal straight walk. 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;
- The data for this example has been collected by the authors.

Example 3 – Segmentation of the systolic of the arterial blood pressure (ABP) wave

- Signal type: Continuous ABP data;
- Description: This data was downloaded from the MIMIC-II Waveform's Physiobank ATM database. It was collected from bedside patient monitors in adult and neonatal intensive care units (ICUs), at a sampling rate of 125 Hz. Multiple waveforms were acquired simultaneously, such as ECG, ABP and respiratory (Goldberger et al., 2000; Saeed et al., 2011);
- Goal: The dicrotic notch corresponds to the physiological event of the aortical valve closure, which triggers the increase of the aortic pressure and signifies the end of the systolic phase. In this problem, it is asked to segment the systolic phase of the ABP wave.

Example 4 – Electrocardiogram peak detector

- Signal Type: Electrocardiogram signal;
- Description: This data has been downloaded from the MIT-BIH Arrhythmia Database found at the Physiobank ATM. The source of the ECGs included in this database is a set of over 4000 longterm Holter recordings that were obtained by the Beth Israel Hospital Arrhythmia Laboratory between 1975 and 1979. The Database contains two main sets, one with 23 records chosen at random from this set, and the other with 25 records that include a variety of rare but clinically important phenomena. The recording was digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range (Goldberger et al., 2000; Moody & Mark, 2001);
- Goal: Detect all the major peaks from the record.

Example 5 – Straight line trajectory tracking

- Signal type: A vector of Cartesian coordinates in two-dimensional space;
- Description: This data was generated to simulate a spatial trajectory of a Human subject inside a building environment. Indoor navigation systems rely on a set of sensors that are able to convert contextual and physical information into the subject's absolute location inside a building;
- Goal: The automatic detection of trajectory features, such as straight walks and turns can be used to evaluate trajectory anomalies;
- The data for this example has been collected by the authors.

Example 6 – Event detection 5s after the start of the exercise

- Signal type: Accelerometer data;
- Description: The accelerometer was positioned on the wrist while the subject performed a repetition of a weight lifting exercise (biceps curl) during 10 s. The data has been acquired at 1000 Hz;
- Goal: 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;
- The data for this example has been collected by the authors.

5. Results

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

5.1. *Examples' solution*

The tool focuses essentially on facilitating the solving of simple tasks, although it can be used for more complex scenarios. We will present three examples: the first and second examples consists of simple problems that require the detection of local maxima and minima. The third example is a more challenging task, which requires the use of more sophisticated mechanisms of the regular expression module.

5.1.1. *Example 2 – Step detection in accelerometer signals*

The example consists of detecting the subject's steps in the acceleration signal. For this particular case, only the detection of the right heel contact will be discussed, whereas the left heel contact example's solution is presented in Table 3.

The signal is initially pre-processed in order to ease the identification of the subject's steps. This is achieved using a low pass filter (☴ or LP). The result is presented in the second image of the first step in Fig. 6. An highlight is also present and delineates a segment of

Table 3
Solutions of examples using the proposed syntactic approach. For each example, a short description, the type of symbolic connotation and the regex pattern used for query are presented.

Example	Fig.	Pre-processing	Connotation	Search
Start of Plateau	9a	~ 500	⇞ 0.8 ∂ 0.05	`p1n`
End of Plateau	9a	~ 500	⇞ 0.8 ∂ 0.05	`z1n`
Step Detection (Left Heel)	9b	☴ 2 ~ 50	⇞ 0.4 ∂ 0.05	`0z0p`
Step Detection (Right Heel)	9b	☴ 2 ~ 50	⇞ 0.3 ∂ 0.05	`1z1p`
Dicrotic Notch Detection	9c	☲ 1 20	↕ 0.3 ∂ 0.01	`(1p). +?0`
Electrocardiogram Peak Detector	9d	☲ 5 50	∂ 0.01	`pn`
Straight Line Tracking	9e	none	∂: 0.05	`z*?`
Stable Lifting Detection	9f	◎ ‖ ~ 500 | ◎ ~ 750	⇞ 0.2 | ⇞ -0.2 ∂ 0.01	`(?<=1.{15000,})(n1p)(.*?)(n1p)`

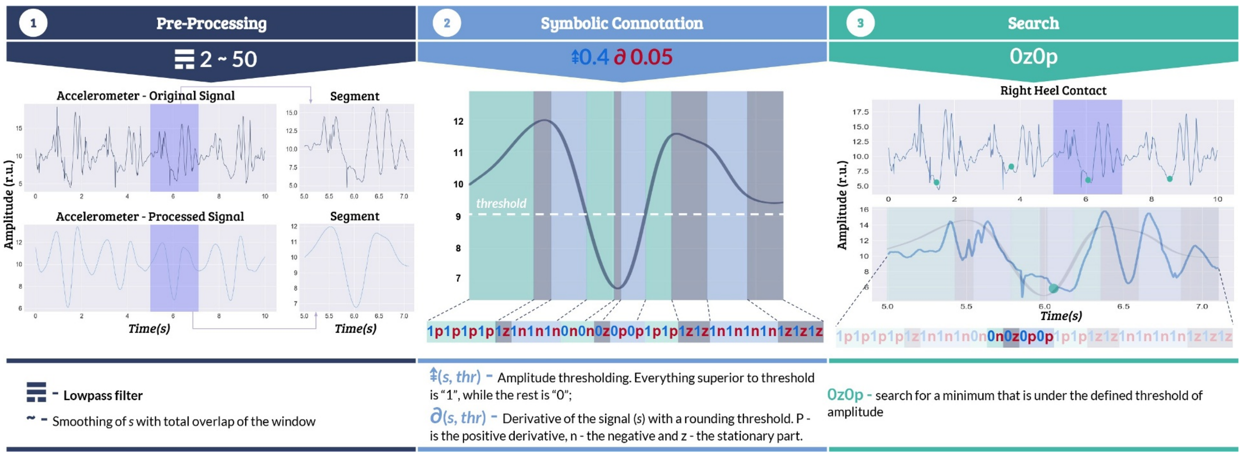

1 Pre-Processing
2 ~ 50
Accelerometer - Original Signal
Segment
Accelerometer - Processed Signal
Amplitude (r.u.)
Time(s)
2 Symbolic Connotation
0.4 ∂ 0.05
threshold
1p1p1p1p1z1n1n1n0n0n0z0p0p1p1p1z1z1n1n1n1n1z1z1z
3 Search
0z0p
Right Heel Contact
- Lowpass filter
~ - Smoothing of s with total overlap of the window
(s, thr) - Amplitude thresholding. Everything superior to threshold is "1", while the rest is "0";
∂(s, thr) - Derivative of the signal (s) with a rounding threshold. P - is the positive derivative, n - the negative and z - the stationary part.
0z0p - search for a minimum that is under the defined threshold of amplitude

Fig. 6. Example 2 – Solution pipeline of Step Detection example. At the bottom of the figure is summarized the operators and methods used in each step. In the symbolic connotation step, the alternation between the methods is made with colours (Blue - ampC- ⇞, Red - diffC - ∂). A doted line is shown to represent the threshold level. To each colorband in the signal corresponds a specific primitive. A positive match is highlighted with green in the search step. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the signal that has a minimum peak, which corresponds to a right heel contact.

The string representation of this segment is depicted in the second step. The string is a sequence of primitives composed by two connotation methods (ampC - ⇞ in blue, diffC - ∂ in red). Based on these methods, the samples of the signal with amplitudes greater than 40% of the range amplitude are transcribed into 1, while the remaining turn into 0; the slopes are converted into *p, z* and *n*, when rising, being stationary and falling, respectively. The solution involves detecting each minimum with an amplitude inferior to the threshold level. The morphological representation of a minimum can be reduced to a negative slope followed by a positive one, which in the symbolic representation is defined by the second connotation method as the transition from *n* to *p* or *z* to *p*. Regarding the amplitude requirement, it is assigned by the first connotation method, in which any value lower to the threshold level is 0. The regular expression used to find this minimum was 0*z*0*p*, 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. 6.

5.1.2. Example 3 – Dicrotic notch detection in ABP signals

The ABP waves are morphologically represented by a high positive slope that corresponds to the systolic uptake and ends in the peak of the systolic pressure. After this behaviour, follows the systolic decline, which ends with the aortic valve closure, named the dicrotic notch in the signal representation (Nirmalan & Dark, 2014).

These types of signals are commonly affected by low frequency noise that modulates the signal and contaminated with high frequency noise of low amplitude. The typical procedure is to bandpass the ABP signal in order to remove both types of noise. Fig. 7

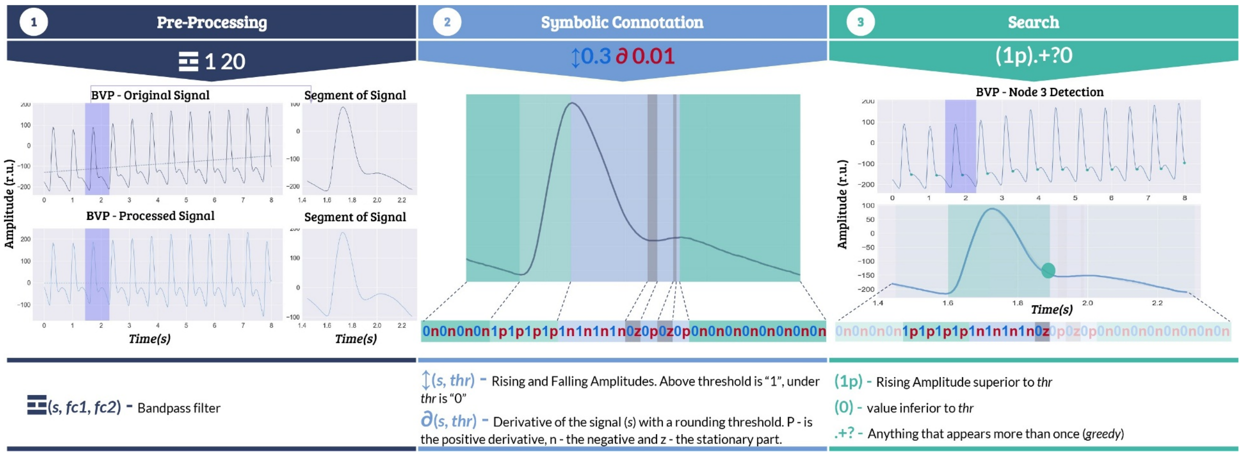
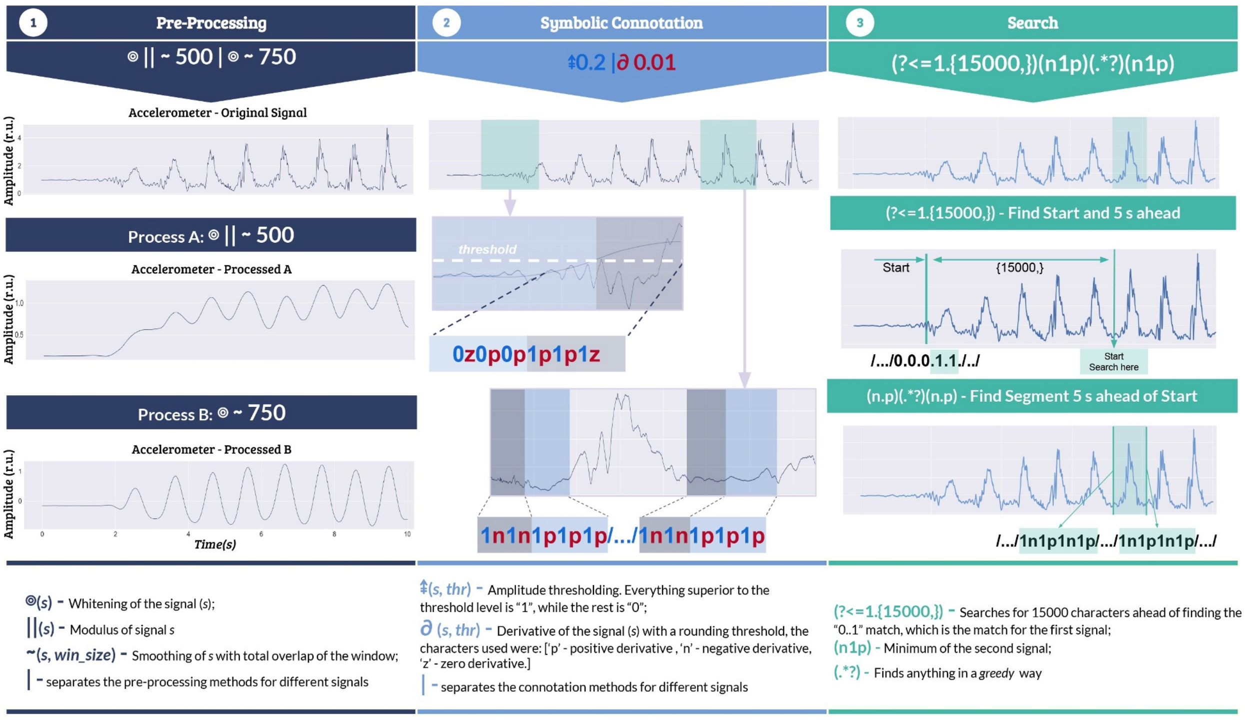

1 Pre-Processing
1 20
BVP - Original Signal
Segment of Signal
BVP - Processed Signal
Amplitude (r.u.)
Time(s)
2 Symbolic Connotation
0.3 ∂ 0.01
0n0n0n0n1p1p1p1p1n1n1n1n0z0p0z0p0n0n0n0n0n0n0n0n
3 Search
(1p).+?0
BVP - Node 3 Detection
(s, fc1, fc2) - Bandpass filter
(s, thr) - Rising and Falling Amplitudes. Above threshold is "1", under thr is "0"
∂(s, thr) - Derivative of the signal (s) with a rounding threshold. P - is the positive derivative, n - the negative and z - the stationary part.
(1p) - Rising Amplitude superior to thr
(0) - value inferior to thr
.+? - Anything that appears more than once (greedy)

Fig. 7. Example 3 – Solution pipeline of Dicrotic notch detection example. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between methods is represented with colors (Blue - RiseAmp - ↕, Red - diffC - ∂). For each colorband in the signal there is a corresponding primitive. The match is highlighted with green in the search step. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)


1 Pre-Processing
⊚ || ~ 500 | ⊚ ~ 750
2 Symbolic Connotation
⇕0.2 | ∂ 0.01
3 Search
(?<=1.{15000,})(n1p)(.*?)(n1p)
Accelerometer - Original Signal
Amplitude (r.u.)
Process A: ⊚ || ~ 500
Accelerometer - Processed A
Process B: ⊚ ~ 750
Accelerometer - Processed B
Time(s)
threshold
0z0p0p1p1p1z
1n1n1p1p1p/.../1n1n1p1p1p
(?<=1.{15000,}) - Find Start and 5 s ahead
Start
{15000,}
/.../0.0.0.1.1./../
Start
Search here
(n.p)(.*?)(n.p) - Find Segment 5 s ahead of Start
/.../1n1p1n1p/.../1n1p1n1p/.../

⊚(*s*) - Whitening of the signal (*s*);
||(*s*) - Modulus of signal *s*
~(*s, win_size*) - Smoothing of *s* with total overlap of the window;
| - separates the pre-processing methods for different signals

⇕(*s, thr*) - Amplitude thresholding. Everything superior to the threshold level is "1", while the rest is "0";
∂ (*s, thr*) - Derivative of the signal (*s*) with a rounding threshold, the characters used were: ['p' - positive derivative , 'n' - negative derivative, 'z' - zero derivative.]
| - separates the connotation methods for different signals

(?<=1.{15000,}) - Searches for 15000 characters ahead of finding the "0..1" match, which is the match for the first signal;
(n1p) - Minimum of the second signal;
(.*?) - Finds anything in a *greedy* way

Fig. 8. Example 6 – Solution pipeline of Stable Lifting Detection. The operators or methods used in each step are summarized at the bottom of the figure. In the symbolic connotation step, the alternation between the methods is made with colors (Blue - RiseAmp - ⇕, Red - diffC - ∂). The threshold level is identified with a white dotted line. To each colorband in the signal corresponds a specific primitive. The match is highlighted with green in the "Search" step. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

depicts how a band pass filter that cuts frequencies under 1 Hz and above 20 Hz (☰ or BP) is applied to the signal to remove both types of noise. The modulation removal is shown with the linear regression in both original and pre-processed signals, which in the first case has a positive slope and after the pre-processing is approximately zero.

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

The first connotation method is necessary in order to distinguish between the rising slope that occurs in the beginning of the pressure wave and the one after the dicrotic notch. With this distinction, it is possible to find the beginning of the ABP wave as a high positive slope (1*p*) 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 1*p* primitive and end with the first 0. primitive. The example is solved with the following regular expression: (1*p*).*?(0.). This string means that the search will match anything (represented by ".*?") between the first "1*p*" primitive and the first "0." primitive.

5.1.3. Example 6 – Stable lifting detection in accelerometer signals

In the previous example, the solution was achieved by searching for one simple transition in the string generated by the sequence of connotation methods. This tool may also be used to solve more complex examples in the same manner. The next problem involves the segmentation of a lifting step that has occurred 5 s after the start of a weight lifting exercise. The example can be solved in two steps: (1) find the start of the exercise, (2) search for the segment 5 s after the start. This rationale can be expressed by combining two distinct symbolic representations of the same original signal, which requires the use of two pre-processing and symbolic connotation sets, in which one is used for the detection of the start and the other to find the desired segment.

Fig. 8 demonstrates how the example is solved. In both pre-processing and symbolic connotation steps, a vertical bar|separates the methods that are applied for each representation of the same signal. The pre-processing phase uses a sequence of whitening, modulus and smoothing of the signal for "Process A", and a sequence of whitening and smoothing for "Process B". The first processing sequence turns the signal similar to a plateau, in which the beginning of the activity is easily identified; whereas, in the second sequence, the signal is smoothed such that each lifting step is well defined.

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

The regular expression written to solve the example is $(? <= 1.\{15{,}000, \})(n1p)(.*?)(n1p)$. 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 15,000 is calculated by the number of characters that correspond to 5 s 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 5 s, respectively.

The resulting signal of "Process B" in Fig. 8 shows that each lifting step can be segmented by detecting the local minima of the signal. In (2), the compiler searches for two local minima ($n1p$) and matches anything in between them (.*?). Combining (1) and (2) results in searching for two local minima and match anything in between them, 15,000 characters ahead of the starting point.

5.1.4. Summary

At this point, introductory examples have been explained and can be used to understand the workability of the tool. Table 3 and Fig. 9 summarize the example's resolution for each step of the pipeline. Additional examples are also included and can be found in the GitHub repository of the tool.[1]

5.2. Performance

One of the purposes of this study concerns the ability to create a way of solving generic tasks in a simpler and more intuitive manner. This requires to evaluate the legibility and difficulty in generating the solution for the corresponding task. In the programming field, Halstead measures are typically used with 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 operators (*oprt*) and operands (*oprd*), and the total number of operators (*Toprt*) and total number of operands (*Toprd*). The metrics used are listed below (Kayam et al., 2016):

Vocabulary

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

$$Voc = oprt + oprd \tag{1}$$

Length

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

$$Lgth = Toprt + Toprd \tag{2}$$

Calculated length

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

$$CL = oprt*log_2(oprt) + oprd*log_2(oprd) \tag{3}$$

Volume

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

$$Vol = Lgth*log_2(Voc) \tag{4}$$

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

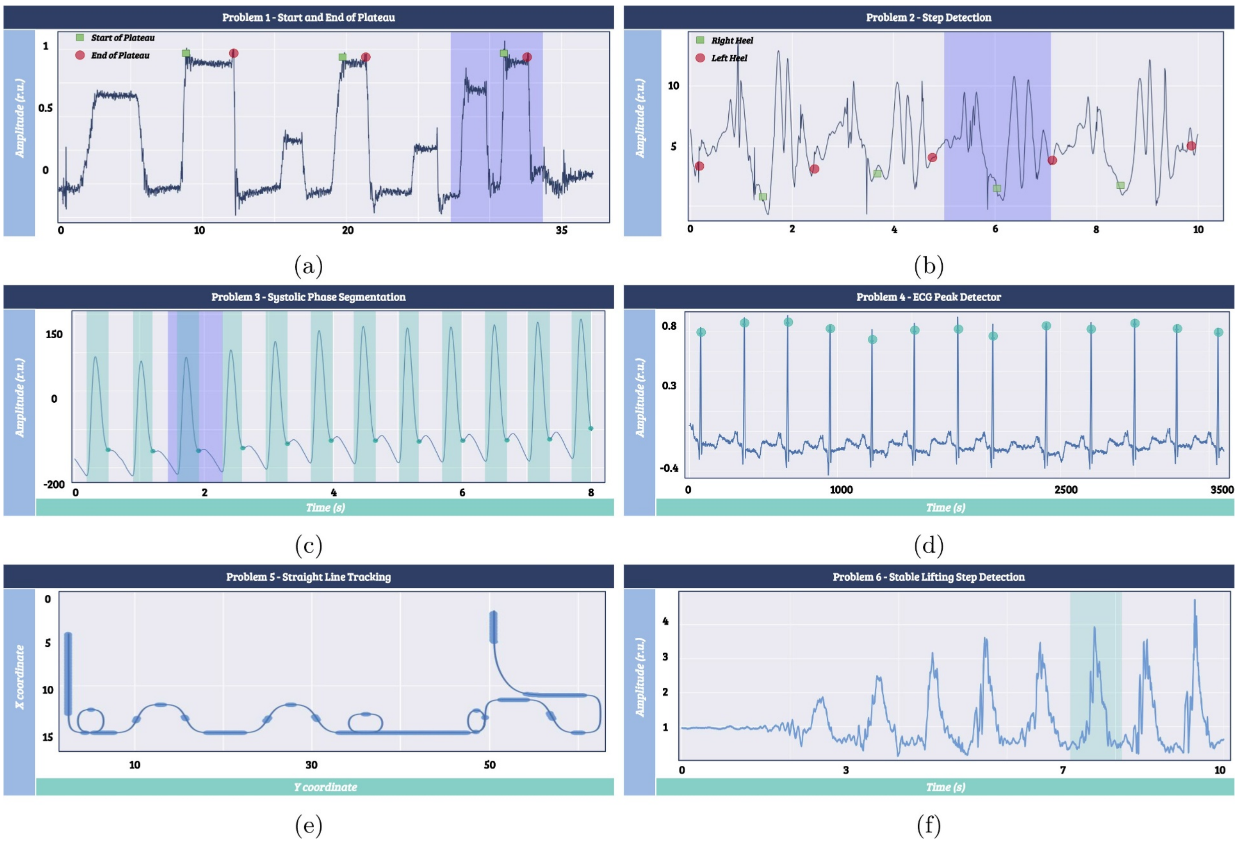


Fig. 9. Summary of the resolution of the examples listed above with the SSTS.

Difficulty

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

$$Dif = \left(\frac{oprt}{2} + \frac{T\,oprd}{oprd}\right) \tag{5}$$

Effort

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

$$Efrt = V\,ol{*}Dif \tag{6}$$

A complexity evaluation was performed to compare the script written in the classical approach and the syntactic approach. Concerning the classical approach, the list of operators and operands are selected from the operator module from Python (P. S. Foundation, 2017a,b), which includes the list of arithmetic, logical, comparison, bitwise, assignment and special operators; 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:

$$pks = peakdelta(s,\ delta = np.\ percentile(s,\ 70) - np.\ percentile(s,\ 30)) \tag{7}$$

- 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′;

Table 4.
Evaluation of the complexity in the resolution of examples with the *syntactic* approach and with the *classical* approach.Voc - Vocabulary, Lgth - Length, CL - Calculated Length, V - volume, D - difficulty, E - effort.

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

- 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 with 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) are defined as operators (Jan Goyvaerts, 2012). 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: ∼ 500 ⇞ 0.8 ∂ 0.05
Search: (z1n).*?

- Total List of Operators: '∼', '⇞', '∂', '()' and '*?';
- Total List of Operands: '500′, '0.8′, '0.05′, 'z', '1′, 'n' and '.';
- List of Distinct Operators: '∼', '⇞', '∂', '()' and '*?';
- List of Distinct Operands: '500′, '0.8′, '0.05′, 'z', '1′, 'n' and '.'.

Table 4 presents the performance measurements for both methods of solving the demonstrated examples. For the evaluation, the pre-processing step was omitted, which means that only the 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.

6. Implications

The previous section demonstrated the potential of the proposed framework in delivering a more expressive methodology of solving query search tasks in time series. For the complete group of examples presented, all the steps involved were written in a symbolic manner. This fact enabled to ease the way in which time series are interpreted and how the final search is achieved.

The results from the Halstead measurements summarized in Table 4 demonstrates that, in general, the complexity is decreased using the SSTS approach in comparison with the classical solution. In the circumstances in which the difference is not significant, such as the Example 6 (Stable Lifting Detection), the complexity measurements increase significantly. This fact suggests that for more complicated problems, the complexity of the regular expression increases, which is in line with the assumptions referred at Section 6.1.

6.1. Symbolic series generation

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

6.2. Required background knowledge

Each step of the SSTS pipeline has an important role in solving the examples, especially the two initial steps (pre-processing and symbolic connotation). Both steps aim to simplify the signal, such that the search procedure can be composed as easily as possible. This approach aims to prepare the signal so that a sequence of symbols can be used to simplify the search step. Therefore, choosing an adequate set of pre-processing methods and proper coordination with the symbolic connotation methods will reduce the search task complexity, both in the writing and interpretation processes. Section 1.2 mentions that SSTS users should have background knowledge in both signal processing techniques and regular expressions. In fact, without prior consolidation of these topics, the user might have an additional effort in generating a solution to solve the required problems. Firstly, for circumstances in which the user experiences difficulty in using such methods, the pre-processing and the symbolic connotation steps will not be so intuitive. Secondly, for in case the user is not familiarized with regular expressions, it will be hard to generate appropriate expressions to search for patterns on the symbolic time series, especially for more difficult problems, which requires the use of advanced operators.

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

7. Conclusion and future work

This work has proven to be effective in solving straightforward query tasks with a syntactic approach in time series. SSTS demonstrated capabilities in finding maxima, minima, transitions and other events on time series, which could be used to solve a multitude of problems related to a wide range of signals. All these capabilities were achieved using a more empirical, expressive and legible approach, both in terms of reading and writing.

Although difficult to measure, we experienced that this tool has a fast learning curve because it focuses on the empiric and morphological analysis of the signal and it can enable a user, with appropriate learning and training, to reduce the algorithm's complexity and increase the time efficiency in writing query tasks.

This work shows promising results but has still room for improvements in multiple ends. Regarding its dimensionality concern, the string generated during the symbolic connotation step increases rapidly with the number of connotation methods and turns the search procedure more complex. It would be interesting to simplify the symbolic representation that is generated, whether by giving meaning to specific sequences of primitives that are most frequently present in this representation (e.g. minima or maxima), and assign it a specific symbol; or use an encoding technique to reduce the dimensionality of the symbolic representation.

In this work, we presented functions to which were assigned specific tokens and could be used either during the pre-processing or the symbolic connotation steps. SSTS is completely customizable, since users can create their own dictionary of symbols, custom tokens and extend the set of functions and methods presented. This fact might facilitate to solve query search problems in different domains. Both the encoding techniques for dimension reduction and the customization of the tool give rise to an improvement in the search step as well, which involves the possibility of using a *meta*-parsing technique that is more synchronized with the two previous steps, giving more appropriate and diverse functionalities to the search procedure.

The provided examples were convenient to show some of the potential applications of this tool, but lack diversity to fully disclose the promising capabilities that it can have in a near future. A solid and representative set of symbols can be created for signal representation, and it can be used to directly translate the signal in a "generic" representation without using connotation methods and by automatically learning the set of symbols that are more appropriate for the symbolic representation.

The new abstraction layer introduced by SSTS regarding how time series can be interpreted can also lead to future research regarding explainable Artificial Intelligence (AI) in time series. Deep learning approaches trained with large datasets have remarkable results, but lack of explainability and transparency, thus, constituting "black-boxes", which might result in barriers to the adoption of technology, especially in the clinical context (Holzinger, Biemann, Pattichis, & Kell, 2017). The ability to express the behaviour of time series using symbolic connotation and syntax can be reused to provide explanations justifying its predictions and determine if the model can be trusted.

Acknowledgements

We would like to acknowledge the financial support obtained from North Portugal Regional Operational Programme (NORTE 2020), Portugal 2020 and the European Regional Development Fund (ERDF) from European Union through the project Symbiotic technology for societal efficiency gains: Deus ex Machina (DEM), NORTE-01-0145-FEDER-000026. We would like to acknowledge as well the projects AHA CMUP-ERI/HCI/0046 and INSIDE CMUP-ERI/HCI/051/2013 both financed by Fundaç̃ao para a Ciência e Tecnologia (FCT).

References

Abbasi, A. Q., & Loun, W. A. (2014). Symbolic time series analysis of temporal gait dynamics. *Journal of Signal Processing Systems, 74*(3), 417–422. https://doi.org/10.1007/s11265-013-0836-1.
Chandler, D. (2017). *Semiotics: The basics* (3rd ed). Routledge.
Coutinho, D. P., Fred, A. L. N., & Figueiredo, M. A. T. (2010). One-lead ECG-based personal identification using ziv-merhav cross parsing. *20th International conference on pattern recognition* (pp. 3858–3861). . https://doi.org/10.1109/ICPR.2010.940.
Downey, A. B. (2016). *Think DSP – Digital signal processing in python* (1st ed.). 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media95472.
Friedl, J. (2006). *Mastering regular expressions* (3rd ed.). O'Rilley Media, Inc.

Fu, K. S. (1971). Stochastic automata, stochastic languages and pattern recognition. *Journal of Cybernetics, 1*(3), 31–49. https://doi.org/10.1080/01969727108548630. arXiv: http://dx.doi.org/10.1080/01969727108548630URL http://dx.doi.org/10.1080/01969727108548630.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., et al. (2000). Physiobank, physiotoolkit, and physionet. *Circulation, 101*(23), e215–e220. https://doi.org/10.1161/01.CIR.101.23.e215. arXiv hTtp://circ.ahajournals.org/content/101/23/e215.full.pdf.

Hamdi, S., Ben Abdallah, A., & Bedoui, M. H. (2017). Real time QRS complex detection using DFA and regular grammar. *BioMedical Engineering OnLine, 16*(1), 31. https://doi.org/10.1186/s12938-017-0322-2. URL https://doi.org/10.1186/s12938-017-0322-2.

Holzinger, A. (2013). Human-computer interaction and knowledge discovery (HCI-KDD): What is the benefit of bringing those two fields to work together? In A. Cuzzocrea, C. Kittl, D. E. Simos, E. Weippl, & L. Xu (Eds.). *Availability, reliability, and security in information systems and HCI* (pp. 319–328). Berlin, Heidelberg: Springer Berlin Heidelberg.

Holzinger, A., Biemann, C., Pattichis, C. S., & Kell, D. B. What do we need to build explainable ai systems for the medical domain?, 28 December 2017. arXiv preprint arXiv:1712.09923.

Holzinger, A., & Pasi, G. (2015). Introduction to the special issue on “interactive data analysis”. *Information Processing & Management, 51*(2), 141–143. doi:hTtps://doi.org/10.1016/j.ipm.2014.11.002. URL http://www.sciencedirect.com/science/article/pii/S0306457314001101.

Holzinger, A., & Zupan, M. (2013). Knodwat: A scientific framework application for testing knowledge discovery methods for the biomedical domain. *BMC Bioinformatics, 14*(1), 191. https://doi.org/10.1186/1471-2105-14-191. URL https://doi.org/10.1186/1471-2105-14-191.

Horowitz, S. L. (1975). A syntactic algorithm for peak detection in waveforms with applications to cardiography. *Communication of the ACM, 18*(5), 281–285. https://doi.org/10.1145/360762.360810http://doi.acm.org/10.1145/360762.360810.

Hussain, L., Aziz, W., Alowibdi, J. S., Habib, N., Rafique, M., Saeed, S., et al. (2017). Symbolic time series analysis of electroen cephalographic (EEG) epileptic seizure and brain dynamics with eye-open and eye-closed subjects during resting states. *Journal of Physiological Anthropology, 36*(1), 21. https://doi.org/10.1186/s40101-017-0136-8.

Iverson, K. E. (1980). Notation as a tool of thought. *Commun. ACM, 23*(8), 444–465. https://doi.org/10.1145/358896.358899. URL http://doi.acm.org/10.1145/358896.358899.

Jan Goyvaerts, S. L. (2012). *Regular expressions cookbook* (2nd ed.). 1005 Gravenstein Highway North, Se-bastopol, CA: O'Reilly Media, Inc95472.

Kayam, M., Fuwa, M., Kunimune, H., Hashimoto, M., & Asano, D. K. (2016). Assessing your algorithm: A program complexity metrics for basic algorithmic thinking education. *2016 11th International conference on computer science education (ICCSE)* (pp. 309–313). . https://doi.org/10.1109/ICCSE.2016.7581599.

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In C. Shannon, & J. McCarthy (Eds.). *Automata studies* (pp. 3–41). Princeton, NJ: Princeton University Press.

Li, Y., & Ray, A. (2017). Unsupervised symbolization of signal time series for extraction of the embedded information. *Entropy, 19*(4).

Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007). Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery, 15*(2), 107–144. https://doi.org/10.1007/s10618-007-0064-z. URL https://doi.org/10.1007/s10618-007-0064-z.

Machado, I. P., Gomes, A. L., Gamboa, H., Paixão, V., & Costa, R. M. (2015). Human activity data discovery from triaxial accelerometer sensor: Non-supervised learning sensitivity to feature extraction parametrization. *Information Processing & Management, 51*(2), 204–214. doi:hTtps://doi.org/10.1016/j.ipm.2014.07.008 http://www.sciencedirect.com/science/article/pii/S0306457314000685.

Moler, C. (2006). The growth of matlab and the mathworks over two decades. https://www.mathworks.com/tagteam/72887_92020v00Cleve_Growth_MATLAB_MathWorks_Two_Decades_Jan_2006.pdf.

Moody, G. B., & Mark, R. G. (2001). The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine, 20*(3), 45–50. https://doi.org/10.1109/51.932724.

Neisser, U. (2014). *Cognitive psychology: Classic edition.* Psychology Press.

Nirmalan, M., & Dark, P. M. (2014). Broader applications of arterial pressure wave form analysis. *Continuing Education in Anaesthesia Critical Care & Pain, 14*(6), 285–290. https://doi.org/10.1093/bjaceaccp/mkt078. arXiv:/Oup/backfile/content_public/journal/ceaccp/14/6/10.1093/bjaceaccp/ mkt078/2/mkt078.pdfURL http://dx.doi.org/10.1093/bjaceaccp/mkt078.

Ordóñez, P., Armstrong, T., Oates, T., & Fackler, J. (2011). Using modified multivariate bag-of-words models to classify physiological data. *Proceedings – IEEE international conference on data mining, ICDM* (pp. 534–539). . https://doi.org/10.1109/ICDMW.2011.174.

P. S. Foundation. (2017a). 10.3. operator - standard operators as functions. URL https://docs.python.org/3/library/operator.html.

P. S. Foundation. (2017b). regex 2017.09.23 https://pypi.python.org/pypi/regex/.

Pavlidis, T. (1971). Linguistic analysis of waveforms. In J. T. Tou (Vol. Ed.), *Software engineering – Computer and information sciences: 2*, (pp. 203–224). Academic Press.

Pavlidis, T. (1973). Waveform segmentation through functional approximation. *IEEE Transactions on Computers, C-22*(7), 689–697. https://doi.org/10.1109/TC.1973.5009136.

Piccardi, C. (2006). On parameter estimation of chaotic systems via symbolic time-series analysis. *Chaos: An Interdisciplinary Journal of Nonlinear Science, 16*(4), 043115. https://doi.org/10.1063/1.2372714http://aip.scitation.org/doi/10.1063/1.2372714.

Rajagopalan, V., Ray, A., Samsi, R., & Mayer, J. (2007). Pattern identification in dynamical systems via symbolic time series analysis. *Pattern Recognition, 40*(11), 2897–2907. https://doi.org/10.1016/J.PATCOG.2007.03.007https://www.sciencedirect.com/science/article/pii/S003132030700115X.

Reghizzi, S. C., Breveglieri, L., & Morzenti, A. (2013). *Introduction to automata theory, languages, and computation* (2nd ed.). London: Springer.

Saeed, M., Villarroel, M., Reisner, A. T., Clifford, G., Lehman, L.-W., Moody, G. et al. Multi- parameter intelligent monitoring in intensive care II (MIMIC-II): A public-access intensive care unit database. Critical Care Medicine 39 952–960 (2011).

Saurer, M. (2017). Document similarity using tf-ldf model - the J way. *Journal of J – An Interdisciplinary Journal on J Programming Language and Applications in Science, 5*, 2–18.

Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A. P., et al. (2015). Time series anomaly discovery with grammar-based compression. *EDBT* (pp. 481–492). .

Thompson, K. (1968). Programming techniques: Regular expression search algorithm. *Communication of the ACM, 11*(6), 419–422. https://doi.org/10.1145/363347.363387. URL http://doi.acm.org/10.1145/363347.363387.

Tirabassi, G., & Masoller, C. Unravelling the community structure of the climate system by using lags and symbolic time-series analysis, *Scientific Reports* 6. doi:10.1038/srep29804.

Trahanias, P., & Skordalakis, E. (1990). Syntactic pattern recognition of the ecg. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*(7), 648–657. https://doi.org/10.1109/34.56207.

Udupa, J. K., & Murthy, I. S. N. (1980). Syntactic approach to ecg rhythm analysis. *IEEE Transactions on Biomedical Engineering BME, 27*(7), 370–375. https://doi.org/10.1109/TBME.1980.326650.

Watanabe, S. (1985). *Pattern recognition: Human and mechanical.* New York, NY: John Wiley & Sons, Inc530.

Yu, J., Cao, J., Liao, W.-H., Chen, Y., Lin, J., & Liu, R. (2017). Multivariate multiscale symbolic entropy analysis of human gait signals. *Entropy, 19*(10).