2

Time Series Fundamentals

The content of this thesis is diverse and covers several different topics. Therefore, the reader will appreciate that we set the foundations that are necessary to fully capture the essence of this work. For this, we introduce the concepts and topics covered and addressed, provide the definitions and define the used notation in this work. We start by explaining global definitions related to *time series*, which is the data of interest in this work. Then, examples of *time series*, ~~mostly~~ biosignals, are given. Further, standard pre-processing methods, representation forms, and distance measures are also explained.

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

2.1 Global Definitions

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

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

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

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

event.

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

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

Definition 5 - Moving Window (MW): A *moving window* is a process of sliding along a time series T to apply a specific method on each *subsequence* it ~~hovers~~ extracts. 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~~ computed. We will show several utilities of this technique further when introducing methods used to pre-process a raw time series and apply standard distance measures. Before explaining these strategies, we will give examples of time series covered in this work, focusing on *biosignals*.

2.2 Filtering

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

2.2.1 Spectral Filtering

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

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

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

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

2.2.2 Smoothing

Another method often used to reduce the presence of noise and represents a variation of a low-pass filter is the smoothing technique. Several variations of this technique exist, being the simplest one a moving average, which uses a moving window, to calculate the mean in each iteration. Other approaches also convolve the signal with a specific window (H) (e.g. *Hanning window*), which instead of giving the same weights to all the samples of the moving window (moving average), attributes a higher weight to the center samples.

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

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

2.2.3 Wandering Baseline

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

2.3 Normalization

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

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

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

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

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

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

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

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

2.4 Transformation

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

2.4.1 Spectral Transformation

One of the first and most well-known techniques suggested for time series transformation was the DFT [1]. The idea behind this concept is that any signal, of any complexity, is a decomposition of a finite number of sine waves. Each wave is represented by a complex number, known as the Fourier coefficient, transforming the signal from the time domain to the frequency domain [65]. This transformation allows us to see the signal differently, highlighting which frequencies concentrate more or less energy. It unveils the presence of specific types of noise or artifacts, or periodic shapes. Figure 2.1 shows the transformation of a signal into the frequency domain. The signal is the sum of two different sine waves with 2 and 15 Hz respectively. The result is a frequency series with two main peaks, at the frequencies of the sine waves.

2.4.2 Feature-based Representation

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

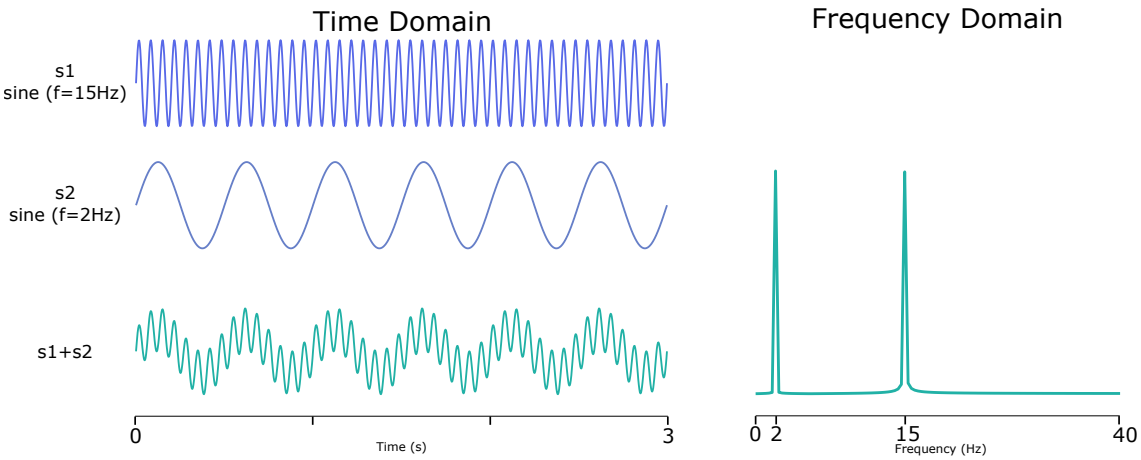

Figure 2.1: DFT of a sum of sine waves.

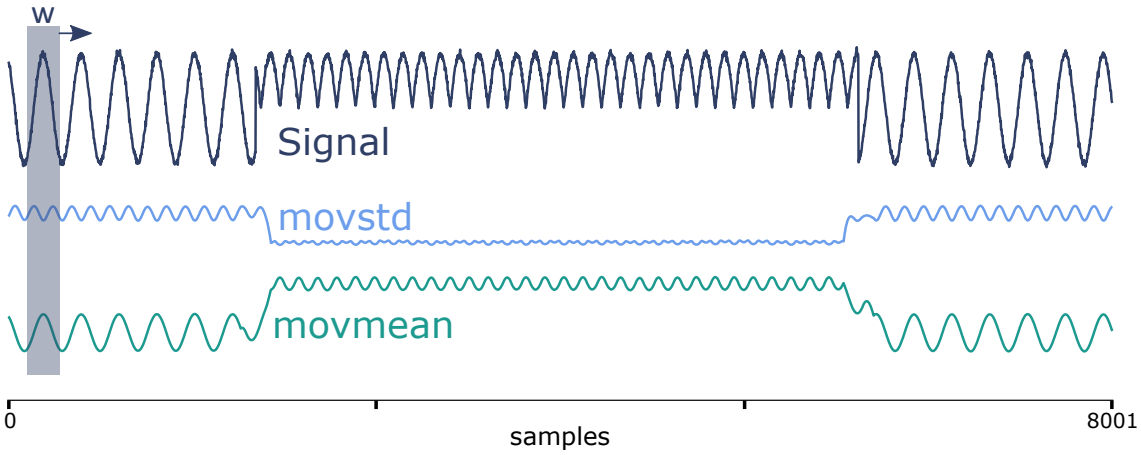

Figure 2.2: Moving window used to extract features with total overlap. The mean and standard deviation are extracted from the signal. The left shows the sine waves, while the right shows the frequency spectrum of the combination of sine waves.

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

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

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

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

2.4.3 Piecewise Aggregate Approximation

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

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

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

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

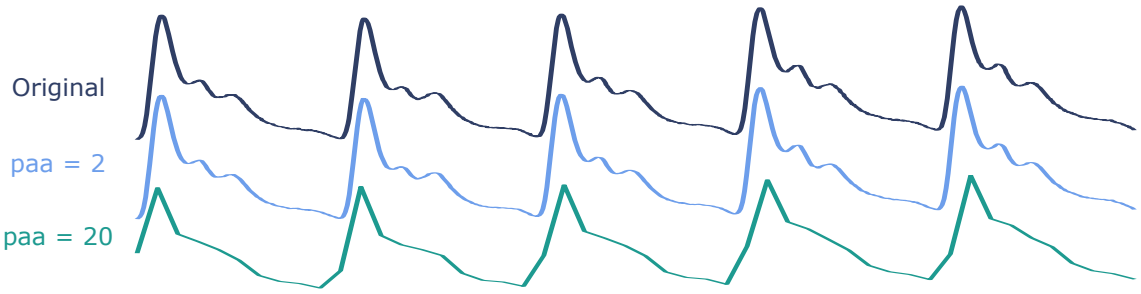

Figure 2.3: PAA representation of a ABP signal, with window sizes of 2 and 20, repsectively. The PAA representation was computed with the PYTS module [19], based on [39].

2.4.4 Symbolic Aggregate Approximation

From this method, a new representation technique was born, transforming the signal from the numerical to the symbolic domain. It is called SAX [43]. This method applies PAA to a z-normalized time series and indexes a *character* to each sample of the simplified signal based on the distribution of its amplitude values. The signal's amplitude values are separated in bins with equal probability. The number of bins is equal to the size of the *alphabet* chosen. Figure 2.4 shows an example of the signal transformed into a string with 3 letters in its alphabet. Such as the DFT, SAX opens doors to analyze time series in a completely different manner, profiting from the much-acquired knowledge in text mining.

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

2.5 Distance Measures

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

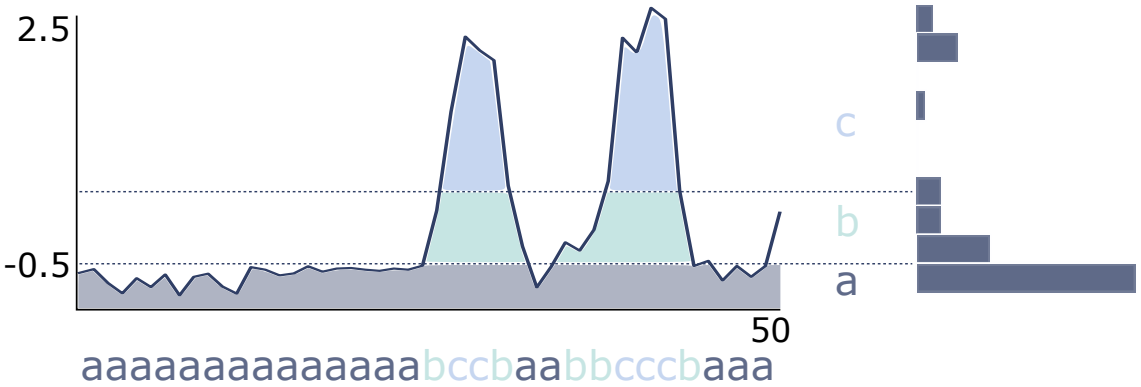

Figure 2.4: SAX representation of a power consumption signal from a Dutch Company, with window bin size of 3. The SAX representation was computed with PYTS based on [43].

2.5.1 Euclidean Distance

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

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

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

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

which represents the square root of the sum of the squared amplitude differences between the samples of each signal. Although the distance measure is simple to compute, it is highly susceptible to typical distortions on time series. When using ED, these distortions must be removed, otherwise, other methods, invariant to these distortions, should be used. Examples of distortions are amplitude and offset distortion, phase distortion, and local scaling ("warping") distortion. The first can be compensated by the z-normalized ED [7]:

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

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

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

2.5.2 Dynamic Time Warping

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

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

The alignment is measured by means of a distance matrix with size n-by-m, where the (i^{th}, j^{th}) cell of the matrix contains the $d(q_i, c_j)$ between the two points q_i and c_j, being $d = (q_i - c_j)^2$ [38]. Figure 2.5 shows an example of a distance matrix between two time series. The matrix fully describes the difference between the two time series and maps where these align. The mapping is made by a warping path, W, that represents the set of matrix cells that minimize the warping cost, also defined as the cumulative distance of these cells [38].

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

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

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

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

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

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

2.5.3 Complexity Invariant Distance

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

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

The CF is defined as [7]:

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

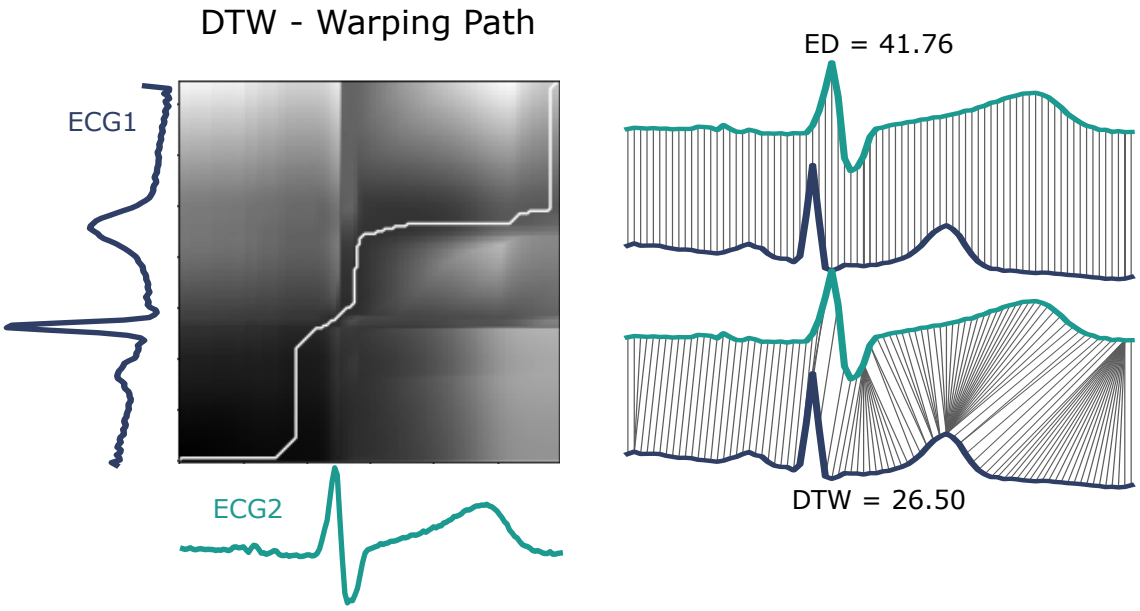

Figure 2.5: DTW and ED distances on two different ECG signals.

where CE represents the complexity estimate of a time series. This estimate is calculated based on the intuition that if we could "stretch" a time series until it becomes a straight line, this line would be as long as the complexity of the signal. It can be computed as the sum of the $n-th$ discrete differences along the time series[7]:

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

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

2.5.4 Feature-based Distance

As mentioned, a feature series F can be computed from the original time series to represent it based on a specific feature. If the size of the *moving window* is equal to the size of the time series then F is represented by a single value. Otherwise, each *subsequence* scanned by the *moving window* is characterized by the feature value, and a F is computed as an array. When multiple features are extracted, each *subsequence* is characterized by a set of features, creating a feature vector $\vec{f}$ with r feature values (to be clear, a feature vector is the set of feature values for a *subsequence* of the time series, while a feature series is a feature representation of the entire time series).

Vector-based distance measures can be used with feature vectors to compare different time series or *subsequences*. There are several vector-based distance measures, including

the already mentioned ED or the manhattan distance, but we will only describe the cosine similarity/distance.

The cosine similarity is a measure of the angle between two vectors determining if these are pointing in the same direction. Consider two feature vectors $\vec{f_A}$ and $\vec{f_B}$. Their cosine similarity is computed as their normalized dot product [30] (equation 2.15).

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

being $||\vec{f_A}||$ and $||\vec{f_B}||$ the euclidean norm of each feature vector, defined as $\sqrt{\Sigma_{i=1}^{r} f_{Ai}^2}$ and $\sqrt{\Sigma_{i=1}^{r} f_{Bi}^2}$, respectively [30].

2.6 Applying Distance Measures

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

2.6.1 Distance Profile

As mentioned in the previous subsection, measuring all the distance pairs of a time series provides the ability to retrieve relevant structural information. When computing the distance of a *subsequence* to all the other *subsequences* of the time series, a *distance profile* is calculated. Each *subsequence* can have a *distance profile* and when computing all the distance pairs, a self-distance matrix is the result.

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

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

Definition 9 - Motif (mtf): The NNbr pair that has the lowest distance forms a motif. That means that on the entire time series, these two *subsequences* are the closest ones. The

opposite is a *discord*.

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

2.6.2 Self-Distance Matrices

A time series can reveal relevant information when each *subsequence* is compared to all the other *subsequences* of the same time series. The result is a pairwise distance matrix that unveils *homogeneity*, *repetition* and *novelty* on the time series [52]. Each are relevant assets for segmentation and summarization tasks.

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

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

being d a distance measure between elements $x_i, x_j \in X$ for $i, j \in [1 : N]$. $SDM(i, j)$ represents a cell of SDM that contains the distance value. When $i = j$ the distance should be zero, therefore the diagonal of SDM has the lower values. Besides the main diagonal, other relevant structures can be found in SDM. These include *homogeneous blocks* and *paths* [52, 53].

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

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

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

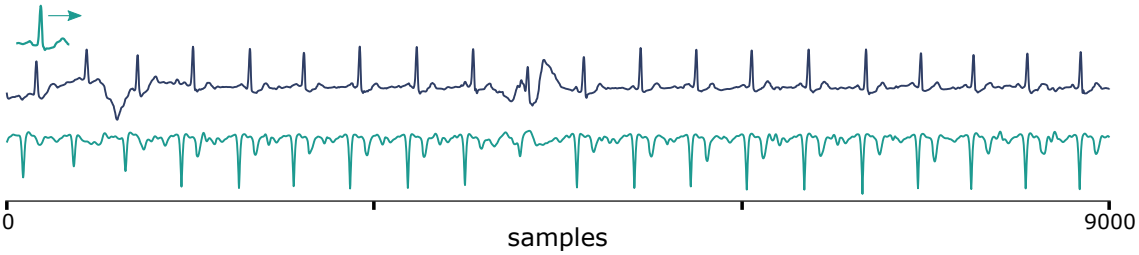

Figure 2.6: Distance profile of a query based search using the z-normalized ED. The template was a PQRS complex of an ECG.

2.6.3 Template-based Search

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

An example of using a query-based search is illustrated in Figure 2.6, where a PQRS complex of an ECG is used as a template to search for all the other complexes. The method applied a simple z-normalized ED. The result shows a distance profile from which *minima* indicates the match with the template.

Other methods can be used to perform query-based search tasks, even using other types of templates, which can be a *drawing* [48] or text [2]. In this work, we will introduce novel ways of performing a query-based search with regex and natural language.

2.6.4 The k-Nearest Neighbors

Having distance measures we can compare signals for several purposes, namely classification. One traditional supervised algorithm for this purpose is the k-NNbr. The assumption of this method is fairly straightforward: new examples will be classified based on the class of their NNbr, that is, the class of the example is the average of the class of its k-NNbr. The process has two steps: (1) finding the k-NNbr and (2) choosing the class of the example based on the neighbors [17].

To find the NNbr, the distance from the example to all the time series of the training set has to be computed. The k-NNbr is the k with the lowest distance. Having the NNbr, the next stage is the determination of the example's class, which can be made with several strategies, such as majority voting, or distance-weighted voting [17].

In this work, we will propose a novel method for time series classification that will have its performance compared with a 1-NNbr based on the z-normalized ED. The proposed method is from the text-domain. The reader will appreciate that an in-depth explanation of the relationship between text and time series is made.

2.7 Text Mining on Time Series

2.7.1 Time Series Textual Abstraction

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

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

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

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

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

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

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

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

2.7.2 Text Features

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

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

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

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

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

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

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

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

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

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

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

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

Due to the probabilistic nature of the BoW and the fact that it contains discrete features, it is suitable to use in naive Bayes classifiers. In the other end, the TF-idf is typically used with linear Support Vector Machines (SVM) classifiers [56].

2.7.3 Text Pattern Search

More formal do not use cntrl+F...

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

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

In order to understand the way that regex work, some of the most used characters and regex primitives are presented as follows [22]:

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

Table 2.1: Main regex operators and meta-characters. Each is presented with a simple example of a good match for a possible regex. A description is also made for complementary understanding.

2.8 Performance and Validation Measures

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

2.8.1 Classification Problems

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.8.2 Event Detection

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

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

Additionally, the distance of the TP events from the ground-truth events can be calculated with several distance-based metrics, namely the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the Mean Signed Error (MsE):

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

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

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

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

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

2.8.3 Evaluating Expressiveness

As we are introducing novel ways of performing more expressive query-based searches with regex, we are measuring the legibility and difficulty in generating a query. In the programming field, *Halstead* measures are typically used for this purpose. These measures describe the complexity of the script directly from the source code, based on a set of metrics calculated with the number of distinct operators (oprt) and operands (oprd), and the total number of operators (Toprt) and total number of operands (Toprd). These metrics are the [36]:

Vocabulary

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

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

Length

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

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

Calculated Length

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

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

Volume

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

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

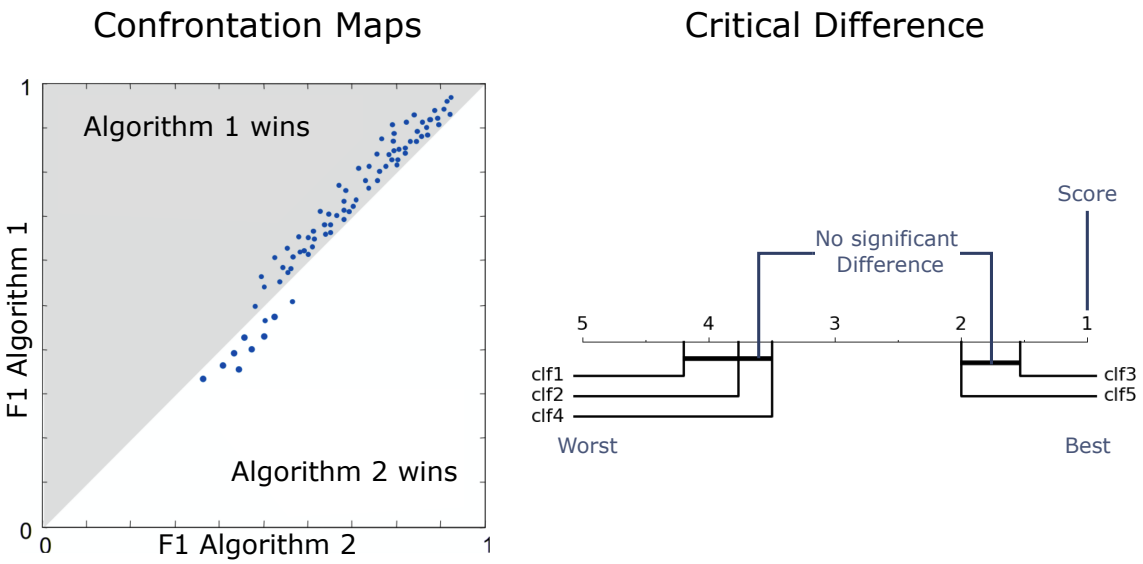

Figure 2.7: On the left are confrontation maps, used to compare classification algorithm performances. On the right is a critical difference map, which shows a statistical difference in the performance of algorithms for general purposes.

Difficulty

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

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

Effort

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

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

2.8.4 Comparing Algorithms

In classification problems, a good procedure is to compare the proposed algorithm with the existing state-of-the-art solutions, so that the reader can understand how the results presented are unbiased from the data. In this work, for each strategy proposed in the domains of classification and event detection, we will compare it with existing solutions. There are two typical ways of displaying this comparison: *confrontation maps* and *critical difference maps*, with examples displayed on Figures 2.7.left and 2.7.right, respectively.

2.8.4.1 Confrontation Maps

When the proposed algorithm is applied to a dataset and compared with another algorithm, we might be tempted to display an overwhelming quantity of information in a table. Although this is a valid approach that should be made to give a full picture to the reader,

there should also be a straightforward way of displaying the same information, but reading it at a glance. With confrontation maps, we can compare the performance of two algorithms just to very intuitively understand if there is a significant difference in their performance. Typically, this map is a scatter plot comparing the *F1-score* or *accuracy* of algorithm 1 *VS* algorithm 2. Figure 2.7.left displays an example of it taken from [37]. Each dot of the plot is a dataset. When it is above the diagonal, it is better classified by algorithm 1, while if below, it is better classified by algorithm 2. In this example, algorithm 1 is better in most datasets.

2.8.4.2 Critical Difference

Critical difference maps are a way of comparing the performance of multiple algorithms or variations of the same algorithm. The plot is a representation of a statistical test over the performance result of each algorithm. The test evaluates if the difference in the performance is significant (critical difference) or not. For instance, on Figure 2.7.right, the plot compares 5 different classifiers (*clf1* to *clf5*) and highlights that the difference in performance is not significant between *clf1, 2* and *4*, neither between *clf3 and 5*. However, *clf3* and *5* have a much better performance than the other classifiers. The closer the classifiers are from the right (1), the better they are. The bold bar connects the classifier with performances that are not significantly different.

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