Temporal sequence alignment and agglomerative clustering for the analysis of medical longitudinal data

Kishan Rama
kishan.rama@tecnico.ulisboa.pt

***Abstract*—Clustering of patients is a very important task in the medical area because it can help physicians treat different groups of patients in a personalized way. Little prior work has been done in clustering based on sequence alignment with longitudinal data that comes from clinical practice. The literature mostly describes techniques that perform sequence alignment and clustering of biological sequences, such as protein sequences. In this work, we propose to apply the Temporal Needleman-Wunsch (TNW), a modified Needleman-Wunsch (NW) algorithm commonly used in Bioinformatics, which incorporates transition times between events of a sequence. The proposed method, named AliClu, tries to output the correct number of clusters by combining TNW with agglomerative hierarchical clustering. The method starts by creating a set of temporal sequences. Pairwise alignment of the sequences is then performed with TNW, which outputs a similarity matrix that is used in agglomerative clustering. In order to find the appropriate number of clusters and assess the stability of each cluster, a resampling technique is applied. Synthetic data was generated to validate the proposed method which allowed us to demonstrate that separation of sequences based on the temporal information is possible. Furthermore, the results with the real dataset, from the Portuguese Society of Rheumatology (Reuma.pt), showed that successful separation of the patients can be achieved. However, our method does not perform so well when the number of clusters increase and long sequences are used, hence, further investigation is required in tuning the parameters.**

***Index Terms*—temporal sequence alignment, clustering, bootstrap, clustering indices.**

I. Introduction

A new exciting era is emerging in HealthCare that will change completely our understanding of medicine and medical practice. The huge amount of data produced within HealthCare holds important knowledge to be gained. Machine Learning and data mining techniques are being used in order to extract this knowledge. Data collected in HealthCare come in many forms and from different sources. Nowadays this data is being stored in Electronic Medical Records (EMRs) that contains large amounts of longitudinal data that tell us the clinical history of the patients. The objective of personalized medicine is to use this information to create individual treatment plans and even predict patient's response to targeted therapies [1]. An interesting way of performing this precision medicine is to use genomic data to create specific gene-based therapies. However, patient's genetic data is only acquired for certain diagnoses and not in every medical center [2].

Imagine a scenario where a doctor is treating a patient and can have immediate access to a list of patients with similar clinical histories to the one being assessed at the moment. This could help the doctor to choose therapies that worked on these other patients and not some common standard therapy, improving the efficiency of treatments. Hence, in this work an alternative way of performing personalized medicine is proposed that could be easily incorporated in clinical use to assist medical doctors. We use longitudinal data present in EMRs of a database from the *Sociedade Portuguesa de Reumatalogia* (Reuma.pt [3]) to find similarities between patients.

A promising research area is sequence alignment. In the 1970s an important alignment method was developed by Needleman and Wunsch (NW) [4] to tackle the difficulty of finding similarities between biological sequences such as protein sequences. This approach is used to assess homology between molecular sequences, however, several adapations have been made to diversify its use in other applications. The temporal information present in EMRs can be exploited in alignment methods to find, for example, similarities between patients based on their medical histories. In this work, we try to cluster patients from Reuma.pt database based on sequence similarity, by using the Temporal Needleman-Wunsch Algorithm (TNW) [5], a modified NW approach that also takes into account time information between events.

The main objective of this study is to find groups of patients with similar temporal characteristics on EMRs. To accomplish this goal we propose to: study several sequence alignment methods mainly the NW and TNW algorithm; study clustering algorithms, mainly hierarchical ones; implement a clustering validation method; create a method that combines sequence alignment and hierarchical clustering to output an appropriate number of clusters; create synthetic data; run the proposed method in synthetic and real data (Reuma.pt).

II. Sequence alignment

For a better comprehension of the TNW algorithm it is a good approach to understand first the NW algorithm.

A. The Needleman-Wunsch Algortihm

The NW algorithm is a global sequence algorithm method commonly used in Bioinformatics to assess similarity between molecular sequences [4]. This method is mathematically proven to find the optimal alignment between two sequences.

It is based on dynamic programming where the basic idea is to solve the problem by dividing into smaller problems; solve the smaller problems optimally and then use the sub-solutions to construct an optimal solution for the original problem.

For a pair of sequences, $X = x_1, ..., x_i, ..., x_m$ and $Y = y_1, ..., y_j, ..., y_n$ where x_i and y_j for $i \in \{1, ..., m\}$ and $j \in \{1, ..., n\}$ are the elements of sequences X and Y with size m and n, respectively; the algorithm uses two matrices score H and traceback T that considers all possible pairs of letters from the two sequences to build the alignment. The NW algorithm consists of three steps:

1) Initialisation of the score matrix H:

$$H_{r0} = -rg; H_{0c} = -cg \quad \forall r \in \{0, ..., m\}, c \in \{0, ..., n\}. \quad (1)$$

2) Calculation of the scores and filling the traceback matrix. The remaining entries of the score matrix are defined as:

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S(x_i, y_j) \\ H_{i-1,j} - g \\ H_{i,j-1} - g \end{cases}, \forall i \in \{1, ..., m\}, j \in \{1, ..., n\}. \quad (2)$$

3) Deducing the alignment from the traceback matrix T.

In the equations (1) and (2), $S(x_i, y_j)$ is a user pre-defined scoring schema that measures the similarity between sequence elements x_i and y_j while g is a constant gap penalty chosen by the user that allows us to penalize the alignment score whenever a gap is inserted.

B. The Temporal Needleman-Wunsch Algorithm

The TNW algorithm is a modified version of the NW method that incorporates the transition times between elements of a sequence. Temporal alignment methods are vastly used for time-series alignment, however, these methods do not impose a penalty for missing events neither use the relative time between them. First, we need to create these temporal sequences as a pre-processing step. Given two consecutive events A and B, and the transition time t between them, then there are two possible encodings: 1) Suffix - encoded (SE): A.t , B.0 and 2) Prefix- ecoded (PE): 0.A , t.B

Memoryless version: The goal of this method is to assess similarity between two sequences by considering the appropriate transition times of events in the two sequences. Consider two sequences: $X = x_1.t_{x_1}, x_2.t_{x_2}, ..., x_m.0$ and $Y = y_1.t_{y_1}, y_2.t_{y_2}, ..., y_n.0$, where t_{x_i} and t_{y_j} are the transition times for elements x_i and y_j, respectively, for $i \in [1, m-1]$ and $j \in [1, n-1]$. The simplest method is to use the transition times of the sequence elements that are being compared in computing the score matrix H. In this memoryless version the H matrix is computed using equations (1) and (3) :

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S(x_i, y_j) - f(t_{x_i}, t_{y_j}) \\ H_{i-1,j} - g \\ H_{i,j-1} - g \end{cases}, \forall i \in \{1, ..., m\}, j \in \{1, ..., n\}. \quad (3)$$

The only difference between equations (2) and (3) is the introduction of a term $f(t_{x_i}, t_{y_j})$ that is a user defined temporal penalty function. The aim of this function is to reduce the similarity $S(x_i, y_j)$ by an amount that depends on t_{x_i} and t_{y_j}. In [5] the following temporal penalty function was used:

$$f(t_{x_i}, t_{y_j}) = T_p \frac{|t_{x_i} - t_{y_j}|}{\max(t_{x_i}, t_{y_j})}, \quad (4)$$

where T_p is some constant factor that will impose the maximum penalty on $S(x_i, x_j)$. This penalty function computes a percentage discrepancy between times t_i and t_j of two events x_i and x_j that are compared. If the events being compared have the same transition times associated to both of them that means that they are similar and the penalty function will be zero. But if the transition times are different then a penalty that depends on the times will be imposed. The memoryless version presents some limitations such as ignoring the transition times for events that align to gaps in score computations, and the choice of encoding affects the overall score and alignment of sequences.

Full TNW version: The Full TNW algorithm can compute the temporal penalties with the total transition time between consecutive match pairs, something not accomplished by the memoryless version, by using PE sequences and two auxiliary matrices TR and TC that *accumulate* these transition times for events that align with gaps associated with the first X and second sequence Y, respectively. The score matrix H is now calculated using:

$$H_{i,j} = \max \begin{Bmatrix} H_{i-1,j-1} + S(x_i, y_j) - f(t_{x_i} + TR_{i-1,j-1}, t_{y_j} + TC_{i-1,j-1}) \\ H_{i-1,j} - g \\ H_{i,j-1} - g \end{Bmatrix}, \forall i \in \{1, ..., m\}, j \in \{1, ..., n\}, \quad (5)$$

where TR and TC are given by:

$$TR_{i,j} = \begin{cases} 0, & \text{if } T(i,j) = \text{“diag”} \\ TR_{i,j-1}, & \text{if } T(i,j) = \text{“left”} \\ TR_{i-1,j} + t_{xi}, & \text{if } T(i,j) = \text{“up”} \end{cases} \quad (6)$$

$$TC_{i,j} = \begin{cases} 0, & \text{if } T(i,j) = \text{“diag”} \\ TC_{i,j-1} + t_{yj}, & \text{if } T(i,j) = \text{“left”} \\ Tc_{i-1,j}, & \text{if } T(i,j) = \text{“up”}. \end{cases} \quad (7)$$

The Full TNW algorithm is a temporal sequence alignment method that can be used for pattern discovery and matching sequences where the timing information between events is relevant. In this work, we will test the Full TNW algorithm with the Reuma.pt dataset.

III. CLUSTERING

Clustering is the task of grouping similar objects together. It is considered an unsupervised learning technique since there is no label assigned to any object, so it has hard to evaluate the quality of any given method. A great amount of clustering algorithms can be found in the literature. The reason for this is due to the fact of not having a clear notion of a cluster and similarity. In this work we will focus on hierarchical clustering, more specifically, on agglomerative clustering. The reason for choosing this algorithm is because it can work directly with a distance matrix D, that will be obtained in this work when performing pairwise sequence alignment.

This approach requires a distance metric over clusters that measure the cluster distance $D(c_i, c_j)$. In this work, we will work with five variants of the agglomerative clustering: single link, complete link, average link, centroid link and ward's method.

A. Clustering Indices

The use of various clustering algorithms with the same dataset can yield different results. Several clustering indices were proposed by researchers to measure clustering results.

Let us introduce the contingency table (CT) and mismatch matrix. Let X be a set of N data points $\{x_1, ..., x_N\}$. If two clustering algorithms are applied to this data, two different clustering results can be obtained: $A = \{A_1, ..., A_R\}$ with R clusters and $B = \{B_1, ..., B_C\}$ with C clusters. The CT of size $R \times C$ shown in Table I contains the information on cluster overlap between A and B where each entry n_{ij} indicate the number of elements that are common to clusters A_i and B_j.

		Partition B				
	Class	B_1	B_2	$\cdots$	B_C	Sums
Partition A	A_1	n_{11}	n_{12}	$\cdots$	n_{1C}	$n_{1\bullet}$
	A_2	n_{21}	n_{22}	$\cdots$	n_{2C}	$n_{2\bullet}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	A_R	n_{R1}	n_{R2}	$\cdots$	n_{RC}	$n_{R\bullet}$
	Sums	$n_{\bullet 1}$	$n_{\bullet 2}$	$\cdots$	$n_{\bullet C}$	N

TABLE I
CONTINGENCY TABLE.

The information in the CT can be transformed into a mismatch matrix shown in Table II that focus on the pairwise agreement between clusters.

		Partition B		
	Number of pairs	In the same cluster	In different clusters	Sums
Partition A	In the same cluster	a	b	$a+b$
	In different clusters	c	d	$c+d$
	Sums	$a+c$	$b+d$	M

TABLE II
MISMATCH MATRIX.

The entries a, b, c and d represent counts of pairs among the $\binom{N}{2}$ distinct pairs in four possible configurations. The measures a and d are usually interpreted as agreements between the two clusterings A and B whereas b and c represent disagreements. The following clustering indices were used in this work:

a) ***Rand Index****[6]:*

$$R = \frac{a+d}{a+b+c+d}. \tag{8}$$

b) ***Adjusted Rand Index****[7]:*

$$ARI = \frac{\sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} - \left[\sum_{i=1}^{R}\binom{n_{i\bullet}}{2}\sum_{j=1}^{C}\binom{n_{\bullet j}}{2}\right]/\binom{N}{2}}{\frac{1}{2}\left[\sum_{i=1}^{R}\binom{n_{i\bullet}}{2} + \sum_{j=1}^{C}\binom{n_{\bullet j}}{2}\right] - \left[\sum_{i=1}^{R}\binom{n_{i\bullet}}{2}\sum_{j=1}^{C}\binom{n_{\bullet j}}{2}\right]/\binom{N}{2}}. \tag{9}$$

c) ***Fowlkes and Mallows****[8]:*

$$FM = \frac{a}{\sqrt{(a+b)(a+c)}}. \tag{10}$$

d) ***Jaccard Index****:*

$$Jaccard = \frac{a}{a+b+c}. \tag{11}$$

e) ***Wallace's coefficients****[9]:*

$$W_{A->B} = \frac{a}{a+b}, \qquad W_{B->A} = \frac{a}{a+c}. \tag{12}$$

f) ***Adjusted Wallace [10]****:*

$$AW_{A->B} = \frac{W_{A->B} - W_{i(A->B)}}{1 - W_{i(A->B)}}, \tag{13}$$

where $W_{i(A->B)}$ is the expected Wallace coefficient under independence and is computed as:

$$W_{i(A->B)} = 1 - SID_B, \tag{14}$$

where SID_B is the Simpson's index of diversity of the clustering B given by:

$$SID_B = 1 - \frac{1}{N(N-1)}\sum_{j=1}^{C} n_{\bullet j}(n_{\bullet j} - 1). \tag{15}$$

B. Validation of Hierarchical Clustering

Clustering validation is one of the most challenging tasks in the area of clustering. The main question that arises in clustering validation is *"how many clusters?"*. In this section the focus will be on validation of hierarchical clustering. The challenge here is to decide where to cut the tree.

We present an automatic validation of hierarchical clustering based on resampling techniques proposed in [11]. This general validation tool consists of a three level assessment of stability.

How many clusters? The criteria used here to find the appropriate number of clusters is based on random resampling. For each Bootstrap sample, a hierarchical clustering is performed. Then, measures of correspondence between partitions like the ones presented in subsection III-A are computed between the clustering of original objects and the clustering of bootstrap samples. More concretely, the summary statistics (average, standard deviation, etc) of M, Rand, AR, FM and Jaccard values is computed for each number number of clusters $k = 2, 3, ..., K$. The value of M indicates the number of bootstrap samples and K is the maximum number of clusters that we want to analyse. Each index value is computed between the unique partition of clustering with all objects and a partition of clustering with three fourths randomly selected objects (bootstrap sample). Here, sampling without replacement is used.

The final decision about the number of clusters k is made by a consensus of all the clustering indices used. Hence, we will look at which k all these indices yield the maximum average values. Furthermore, we can analyse the standard deviation to find where there are sudden decreases.

Cluster Validation In this level the stability of the clusters found before is assessed. The author of [11] considers that

stable clusters from a general statistical point of view are clusters that can be confirmed and reproduced to a high degree. To measure stability of individual clusters three measures of correspondence (Jaccard, rate of recovery and Dice) between a cluster A and a cluster B are used:

$$\tau(A,B)=\frac{|A\cap B|}{|A\cup B|},\quad \gamma(A,B)=\frac{|A\cap B|}{|A|},\quad \eta(A,B)=\frac{|A\cap B|}{|A|+|B|}. \tag{16}$$

After eliciting the number of clusters k in the previous level a clustering of the original sample $X=\{x_1,x_2,...,x_N\}$ into a collection of k clusters $\{A_1,...,A_j,...,A_k\}$ is obtained. Let A_j be one individual cluster whose stability has to be assessed. The same bootstrap method applies here: clustering of a bootstrap sample randomly drawn from the set of all original objects into a collection of k clusters $\{B_1,...,B_k\}$. Then, the definition of stability of cluster A_j using measure γ, for instance, is based on the most similar cluster:

$$\gamma_j^*=\max_{B_i\in\{B_1,...,B_k\}}\gamma(A_j,B_i). \tag{17}$$

By repeating resampling and clustering many times, the stability of the cluster A_j can be assessed, for instance, by computing the average of the corresponding values of γ_k^*. Again, we are looking for higher values of these averages and low standard deviation values, since it is difficult to fix an appropriate threshold to consider a cluster stable.

Reliability In this last level, the reliability of the cluster membership of each individual object is assessed. This step can be performed at the same time the cluster stability is evaluated. For every bootstrap sample, a hierarchical clustering is performed and a dendrogram is obtained. This dendrogram is cut to obtain our correct number of clusters k. Then, the correct classifications of the objects into the clusters of the original sample size can be counted and, lastly, the reliability is this count divided by the total number of simulations.

IV. Proposed Method

The main steps of the proposed approach are now explained in simple terms. The initial step consists pre-processing the raw data to obtain temporal sequences. Then, on the second step, pairqise sequence alignment is performed and a similarity matrix is obtained. The third step consists on converting this similarity matrix into a distance matrix. Agglomerative clustering is then performed with this distance matrix. Furthermore, validation of the clustering results is accomplished.

A. Data Pre-Processing

The available data consists of observations of several variables during time. The distinction between the experiments that will be performed is on the observed variable used to create the temporal sequences. In order to build them we need three variables: $id\_patient$, $event$ and $time$. The $id\_patient$ refers to the unique identifier of the patient, $event$ is the observed variable and $time$ is the variable that gives the temporal information. More concretely, the observed variable $event$ can be, for example, symptoms of a patient, blood pressure, results of medical exams, etc. Thus, it can be a categorical or a numerical variable. If it is a numerical variable then a conversion to categorical must be done. Later, a conversion made in one of the experiments will be explained in more detail. The variable $time$ can appear formatted as a date or just as a number in any time unit (seconds, minutes, days, etc.). Depending on the type of the $time$ variable we can have two types of pre-processing.

a) ***Pre-Processing I:*** $time$ ***is a number****:* In this pre-processing the $time$ variable is just a number. For each patient with an $id\_patient$, the main pre-processing steps are:

1) Eliminate repeated ($id\_patient$, $event$, $time$) pairs.
2) Eliminate rows with NA values.
3) If $event$ is a categorical variable: convert numerical values of $event$ into strings in alphabetical order, i.e $event$ = 1 $\rightarrow$ A, $event$ = 2 $\rightarrow$ B and so on. If the $event$ already appear as strings this step is not necessary. If $event$ is a numerical variable: convert numerical variable into a categorical variable.
4) Create the prefix-encoded temporal sequences.

b) ***Pre-Processing II:*** $time$ ***is a date****:* In this pre-processing the $time$ variable appears in date format. The pre-processing procedure here is quite similar to the previous one with an additional step. The main pre-processing steps are:

1) Eliminate repeated ($id\_patient$, $event$, $time$) rows.
2) Eliminate rows with NA values.
3) This Step is equal to Step 3 of pre-processing I.
4) For each row, expect for the first row for each patient, that will be zero, the relative time interval is given by the difference of $time$ variables between the current row and the previous one. These time intervals are associated with the corresponding row events.
5) Create the prefix-encoded temporal sequences.

After the pre-processing step, all patients are fully characterized by the temporal sequences.

B. Sequence alignment

After creating the temporal sequences in the pre-processing step, it is possible to perform alignment between all patient pairs using the TNW algorithm. The information of all the alignments can be summarized into a $N\times N$ similarity matrix Sm where N indicates the number of patients in our data. In this matrix scores at position (i,j) represents the score of the alignment of patients with $id\_patient\ i$ and $id\_patient\ j$. The alignment of a patient sequence with himself gives a perfect alignment and, hence, the diagonal values of the similarity matrix should be very high. However, the diagonal values are not going to be used in the clustering step and, since the matrix is symmetric, we only need to compute its upper triangle part. These similarity scores can change drastically depending on the choice of the parameters used in the TNW algorithm.

C. Distance matrix

Before using the agglomerative clustering algorithm we need to convert the similarity matrix Sm obtained in the

previous step into a distance matrix D. First, we negate the similarity scores and then in a second step, a shift is made to all scores by adding the maximum similarity score in matrix Sm. The shift is made in order to make all scores greater or equal to zero and can be defined mathematically by Eq. (18). The two steps are combined together in Eq. (19) which gives us the final formula to compute the distance matrix.

$$a = \max_{i,j=1,\dots,N} Sm_{ij}. \tag{18}$$

$$D = -Sm + a\left(\mathbf{1}\cdot\mathbf{1}^T\right), \qquad \mathbf{1} = \begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix} \in \mathbf{R}^N. \tag{19}$$

D. Agglomerative Clustering & Validation

After the conversion made in the previous section, a distance matrix is obtained that can be used in agglomerative clustering.

Since the objective of the work is to retrieve patient clusters a decision has to be made regarding the number of clusters, i.e where to cut the resulting dendrogram. To answer this question, the automatic validation of hierarchical clustering proposed in [11] was applied in this step. The pseudo-code of this procedure is given in Algorithm 1. As parameters, the algorithm needs: a distance metric for the agglomerative clustering algorithm, the number of bootstrap samples M, the maximum number of clusters K and a distance matrix D.

The algorithm begins by performing agglomerative clustering on distance matrix D in Step 1 and by defining the parameters M and K in Step 2. The number of bootstrap samples is set to $M = 250$, the same number used in [11]. This procedure to find the number of clusters is computationally expensive, hence, we decided to not increase the value of M. The maximum number of clusters K to be analysed is defined in terms of the number of patients to be clustered. For example, if there are 100 patients it does not make sense to set $K = 50$ which would result in clusters with small number of patients. Therefore the value of K is set according to the number of patients in the data. Then, an outer loop starts in Step 3 (bootstrap method) for each bootstrap sample. From Step 4 to 6, a bootstrap sample is formed and agglomerative clustering is performed on it. Then, an inner loop is performed to compute the clustering indices between a clustering of the original patients and a clustering of the bootstrap samples (Steps 7-12). In steps 9 and 10, we cut the dendrogram Z and Z', respectively. After running the outer loop M times, the statistics of the clustering indices are computed (Step 13). Finally, in Steps 14 and 15 the decision regarding the number of clusters is made. We choose the k that yield more maximum average values of clustering indices. In this choice, Rand measure is disregarded, since Adjusted Rand is already a "corrected for chance" version of it. Furthermore, to corroborate the previous decision, the standard deviation of the clustering indices for each k is also analysed. This analysis is not automatic. Note that the Algorithm can output automatically the number of clusters if Step 15 is excluded. Otherwise, a non-automatic decision is made by evaluating both the averages and standard deviations of the different clustering indices.

Algorithm 1 Unravelling the number of clusters

1: Perform agglomerative clustering on distance matrix D which gives a dendrogram Z.
2: Let M be the number of bootstrap samples and K the maximum number of clusters.
3: **Repeat** M times:
4: - Boostrap sample – randomly select $\frac{3}{4}$ patients from the original data.
5: - Create a new distance matrix D' for the bootstrap sample.
6: - Perform hierarchical clustering on D' – outputs dendrogram Z'.
7: - Let $k = 2$.
8: **Repeat** until $k = K$:
9: - Cut the dendrogram Z in order to obtain k clusters.
10: - Cut the dendrogram Z' in order to obtain k clusters.
11: - Compute Rand, AR, FM, Jaccard and AW between the original partition and bootstrap partition.
12: - $k = k + 1$.
13: Compute statistics of the M computations for each analysed k.
14: Perform consensus decision on number of clusters.
15: Corroborate the previous decision by analysing the standard deviation of clustering indices for each k.

After the correct number of clusters k is found then the second level proposed in [11] can be assessed regarding the stability of each individual cluster. The Algorithm 2 is followed to understand the stability of the obtained clusters. The structure of this algorithm is identical to Algorithm 1, since the same bootstrap method is applied here.

Algorithm 2 Cluster stability assessment

1: Define k as the number of clusters found with Algorithm 1.
2: Obtain a collection of k clusters $\{A_1, ..., A_j, ..., A_k\}$ by cutting the dendrogram Z.
3: **Repeat** M times:
4: - Bootstrap sample – randomly select $\frac{3}{4}$ patients from the original data.
5: - Create a new distance matrix D' for the bootstrap sample.
6: - Perform hierarchical clustering on D' – outputs a dendrogram Z'.
7: - Obtain a collection of k clusters $\{B_1, ..., B_k\}$ by cutting the dendrogram Z'.
8: - Let $j = 1$.
9: **Repeat** until $j = k$:
10: - Let $\tau_j^* = \max_{B_i \in \{B_1,\dots,B_k\}} \tau(A_j, B_i)$.
11: - Let $\gamma_j^* = \max_{B_i \in \{B_1,\dots,B_k\}} \gamma(A_j, B_i)$.
12: - Let $\eta_j^* = \max_{B_i \in \{B_1,\dots,B_k\}} \eta(A_j, B_i)$.
13: - $j = j + 1$.
14: Compute statistics of the M computations for each analysed cluster.

The algorithm starts by defining k as the number of clusters found with Algorithm 1 in Step 1, and from that, a collection of k clusters $\{A_1, ..., A_j, ..., A_k\}$ is obtained by cutting the dendrogram Z. The bootstrap method is then applied again, for each bootstrap sample that is created a dendrogram Z' is obtained by performing agglomerative clustering on it (Steps 4-6). Then, another collection of k clusters $\{B_1, ..., B_k\}$ of the bootstrap sample is obtained by cutting the dendrogram Z' (Step 7). From Step 8 to 13, three different measures, τ_j^* (Jaccard) , γ_j^* (rate of recovery) and η_j^* (Dice) are computed for each cluster from the collection $\{A_1, ..., A_j, ..., A_k\}$. This indices give a measure of similarity between a found cluster

and the most similar cluster from the collection $\{B_1, ..., B_k\}$. Finally, in Step 14, the stability of the found clusters can be assessed by computing the average values of τ_j^*, γ_j^* and η_j^* and by analysing the standard deviations. As discussed in [11], it is difficult to fix an appropriate threshold to consider a cluster as stable. Therefore, we considered stable clusters the ones that yield high average values (close to one) and low standard deviation values of τ_j^*, γ_j^* and η_j^*.

E. Tuning the parameters

Choosing appropriate parameters of the TNW depends on the application. In this work, all the experiments use the same scoring schema and user-defined temporal penalty function. A simple scoring schema is used where pairs that match get a score of 1 and mismatches get a score of -1.1. Note that in this work the scoring schema could be modified to reflect similarities between events based on medical knowledge. The user-defined temporal penalty function in Eq. (4) at page 2 was implemented which requires a user defined value for T_p.

After setting the scoring schema and the temporal penalty function, the main question that arises is what value of gap penalty, g, and temporal penalty constant, T_p, should be used. We concluded that there is a limited amount of variation that we can perform on the parameters. It was seen that by increasing the gap penalty, alignment of events that do not match can be obtained or by decreasing to a certain amount total misalignments can also be obtained. These type of alignments are not desired in this work. The main type of alignments wanted is aligning similar events and introducing gaps when there are mismatch events, with the final score obtained being affected by the temporal information. Hence, a value of $T_p = 0.25$ is used during the work and the gap penalty is varied from -1 to 1. The reason for this interval is that usually the alignments of the sequences do not change drastically inside this range. In Algorithm 3 it is presented the complete method followed in this work to obtain clusters.

Algorithm 3 AliClu

1: Perform pre-processing I or pre-processing II on the raw data.
2: Set parameters of the TNW algorithm: pre-defined scoring system S defined with a score of 1 for pairs that match and -1.1 for mismatches; user-defined temporal penalty function (Eq. (4)) with $T_p = 0.25$.
3: Let $g = -1$.
4: **Repeat** until $g = 1$:
5: - Perform pairwise alignment using TNW algorithm with g.
6: - Convert similarity matrix S_m into a distance matrix D (Eq. (19)).
7: - Run Algorithm 1 to unravel the number of clusters.
8: - $g = g + 0.1$.
9: Perform consensus decision on the number of clusters.
10: Run Algorithm 2 to assess cluster stability.

The initial step of the algorithm is to perform pre-processing I or pre-processing II, accordingly, on the raw data to obtain temporal sequences (Step 1). Then, the parameters of the TNW algorithm are set (Step 2). The gap penalty used in the TNW algorithm is varied from -1 to 1, in steps of 0.1. For each value of the gap penalty, pairwise alignment using TNW is performed, which outputs a similarity matrix Sm (Step 5). This matrix is converted into a distance matrix D by using Eq. (19) (Step 6). The number of clusters is then found by running Algorithm 1 (Step 7). After running the cycle in Step 4, usually, there are twenty results, one for each value of the gap penalty, each of them with the choice of k and the average values of the clustering indices for that choice. In Step 9, the final number of clusters k is obtained from these twenty different results. This step is decisive if we want the algorithm to output the number of clusters automatically or not. To automate the whole process and in order to obtain a final and definite answer for the number of clusters, we define the final number of clusters as the k that occurs more often on the twenty results. This means that most of the different alignments and clusterings agree on the same number of clusters. In this automated way, the chosen gap penalty is the one that yield best average values of the clustering indices for the final number of clusters. However, when working with real data, most often this final answer might not be correct or make sense at all. In those cases, it is more important if we analyse the twenty different results separately and follow a non-automated way. In this approach, we analyse twenty different results that are composed by the resulting dendrograms, average and standard deviations values of the clustering indices obtained as it will be seen in the next chapter when applying this approach to the Reuma.pt dataset. Finally, in Step 10, the stability of the clusters is assessed by running Algorithm 2. If the results obtained with the Algorithm 3 are not satisfactory, one can try running with different values for the temporal penalty constant Tp.

V. Results

In this section the results obtained with the proposed method are presented. The results are divided into two subsections. The first subsection present the results obtained with the synthetic dataset whereas the second subsection shows the results obtained with the Reuma.pt dataset.

A. Synthetic dataset

In order to have a more controlled manner of testing the proposed method, synthetic data was generated. By having synthetic data, the true labels of the clusters are known a priori which help us in understanding if the method outputs the correct clusters or not. The synthetic dataset consists of temporal sequences generated by continuous-time Markov chains (CTMC). A CTMC $X(t)$ is defined by two components. First component is a discrete-time Markov chain, called the *jump chain* that consists of a countable set of states $E \subset \{0, 1, 2, ...\}$ along with transition probabilities. Second, a set of holding time parameters λ_i is defined to control the amount of time spent in each state.

Q-matrices The matrix $Q = (q_{ij}), i, j \in E$ also referred as the generator matrix allows us an alternative way of specifying a CTMC. This matrix has the following properties:

1) $q_{ii} \leq 0$ for all $i \in E$;
2) $q_{ij} \geq 0$ for all $i, j \in E$ such that $i \neq j$;
3) $\sum_{j \in E} q_{ij} = 0$ for all $i \in E$.

An adequate way to present the information contained in a Q-matrix is by means of a transition rate diagram. In this diagram, the values q_{ij} are shown on the edges whereas the values of q_{ii} are not usually shown because they are implied by the other values

The jump matrix Π The first component of a CTMC is a discrete-time Markov chain defined by a jump matrix Π that gives us the transition probabilities π_{ij}, i.e the same as p_{ij} in an ordinary discrete-time Markov chain. For a given Q-matrix Q, a stochastic matrix Π is associated. The entries of Π are defined as:

$$\pi_{ij} = \begin{cases} \frac{q_{ij}}{q_i} & \text{if } q_i \neq 0 \text{ and } j \neq i, \\ 0 & \text{if } q_i \neq 0 \text{ and } j = i, \\ 0 & \text{if } q_i = 0 \text{ and } j \neq 0, \\ 1 & \text{if } q_i = 0 \text{ and } j = i, \end{cases} \tag{20}$$

where $q_i = \sum_{i \neq j} q_{ij}$. Beside the Q-matrix Q and the Π matrix already presented, we also need an initial probability distribution vector α for the states. Once all the elements are defined then the temporal sequences can be generated with Algorithm 4. The algorithm starts again by initializing the CTMC with an initial state i being drawn. Then, in Steps 2-3, the initial state i is converted to a string i' and the temporal sequence is initialized as $tseq =$‘$0.i'$’. Since, in our experiments, we never used more than five states, a simple conversion of the states into strings in alphabetical order is made, i.e a state $i = 1$ is converted to $i' = A$, state $i = 2$ to $i' = B$ and so on. In Step 4, the time t' of the jump of the CTMC to another state is simulated and then the new state j is simulated (Step 5). Again, after simulating the new state j a convertion to string j' is made and the temporal sequence is update as $tseq = tseq+$ ‘,$t'.j'$’. Finally, we consider the existence of an absorbing state in order to mark an end to the sequences, hence, the simulation continues (Step 6) until an absorbing state is found.

Algorithm 4 Temporal Sequence Generation

1: Initialize the CTMC at $t = 0$ with initial state i drawn from the initial distribution α.
2: Convert the initial state i to a string i'.
3: Initialize the temporal sequence: $tseq =$‘$0.i'$’.
4: Call the current state i; simulate the time of the next event, t', as an $Exponential\,(q_i)$ random variable.
5: Simulate the new state j:
- If $q_i = 0$, set $j = i$ and stop;
- If $q_i \neq 0$, simulate a discrete random variable with probability distribution given by the i-th row of the Π-matrix, i.e, $\frac{q_{ij}}{q_i}, j \neq i$;
- Convert j to a string j';
- Update the temporal sequence: $tseq = tseq+$ ‘,$t'.j'$’.
6: Return to Step 4.

Now that an algorithm is provided to generate temporal sequences, two relevant experiments that were performed are shown. In all the experiments, the Algorithm 3 is used with the exception of the first step. Since the temporal sequences are generated already on the prefix-encoded format, there is no need for the pre-processing step. Also because the true labels of the clusters are known a priori on the experiments, an additional analysis is made where clustering indices are computed between the true labels and the clusters that are going to be found with the method.

1) Generated sequences - Two events: In this experiment the transition diagram for the Q-matrix that specifies the CTMC is shown in Fig. 1.

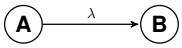

Fig. 1. Transition rate diagram I

Several variations of parameters are made in this experiment in order to test our method against different conditions. The generated dataset can contain a number of clusters $n_{clusters}$. Each cluster i is defined by a CTMC with parameter λ_i and is composed by a number of sequences $n_{sequences}$. The goal is to separate correctly the $n_{clusters}$ clusters of the generated dataset. For each choice of number of sequences per cluster $n_{sequences}$ and linkage method, the experiments are repeated 25 times in order to obtain the percentage of correct decisions, which is defined as the number of times the proposed method outputs the correct number of clusters divided by the total number of experiments. In Figure 2, we present the results of the percentages of correct decisions made by the proposed method when the generated dataset contains two clusters where the CTMCs have parameters $\lambda_1 = 1000$ and $\lambda_2 = 1$. The number ($n_{sequences}$) is varied from 5 up to 100.

Fig. 2. Percentage of correct decisions on the numbers of clusters – 2 clusters ($\lambda_1 = 1000$, $\lambda_2 = 1$).

It is possible to observe different behaviours by the different linkage methods. The Ward's distance metric performed quite well, specially when the number of sequences is 50 and 100 where it attained a 100% correct decision on the number of clusters. Overall, it can be said that the Ward's distance has the best performance among all linkage methods. Table III presents statistics of the clustering indices between the original partitions and the found clusters for the Ward's distance. The averages are very high and the standard deviations remain quite low. All the measures agree, i.e they present values close to each other and therefore it is possible to conclude that the found clusters are equal to the original ones in the majority of the experiments.

2) Generated sequences - Mixture of all type of sequences: A more difficult experiment was performed with a dataset composed of 4 clusters that are generated by the CTMCs presented in Figure 3.

		Number of sequences per cluster				
		5	15	25	50	100
Average	Rand	0.982	0.984	0.967	0.978	0.980
	AR	0.963	0.967	0.933	0.956	0.959
	FM	0.980	0.983	0.966	0.978	0.980
	Jaccard	0.967	0.969	0.939	0.957	0.961
	AW	0.966	0.968	0.936	0.957	0.960
Standard Deviation	Rand	0.059	0.036	0.054	0.023	0.026
	AR	0.112	0.072	0.107	0.045	0.052
	FM	0.065	0.037	0.054	0.023	0.026
	Jaccard	0.106	0.067	0.093	0.043	0.046
	AW	0.110	0.069	0.100	0.044	0.049
Median	Rand	1	1	1	0.980	0.990
	AR	1	1	1	0.960	0.980
	FM	1	1	1	0.980	0.990
	Jaccard	1	1	1	0.960	0.980
	AW	1	1	1	0.960	0.980

TABLE III

AVERAGE, STANDARD DEVIATION AND MEDIAN OF FIVE CLUSTERING INDICES WHEN USING WARD'S METHOD ($\lambda_1 = 1000$, $\lambda_2 = 1$).

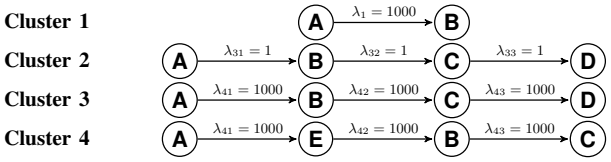

Fig. 3. Final synthetic dataset with 4 clusters.

Similarly to the previous experiments, the number of sequences per cluster is varied. In this experiment, our proposed method is not able to find the 4 clusters from the dataset when tested for 5, 15, 25 and 50 sequences per cluster. We plot the percentages of correct decision for each value of gap penalty. Furthermore, we run our method for different values of temporal penalty constant used in the TNW algorithm, beside $Tp = 0.25$ that is predefined in the method. In Figures 4, 5, 6 and 7, we present the percentage of correct decisions when using $Tp = 0.25$, $Tp = 1$, $Tp = 1.5$ and $Tp = 2$, respectively.

Ward's method is used in the agglomerative algorithm of our method in all these tests. The small negative bars plotted in the figures indicate that the percentage of correct decisions are zero. In Figure 4 it is possible to observe that the percentages of the correct of decisions is zero for the majority of gap penalty values and also for the different choices of number of sequences per cluster, which justifies why our method could not output the correct number of clusters. By increasing the parameter Tp, the percentages of correct decisions increase, mainly for positive gap values, which is observed in Figure 5 where $Tp = 1$. Overall, good performance is verified when $g = 0.3$ and $g = 0.4$ with the latter achieving the highest percentages for all choices of number of sequences per cluster.

In Figure 6, with $Tp = 1.5$, the method performs better from $g = 0.6$ to $g = 1$. The best performance is verified for $g = 0.8$ with 100% of correct decisions on the number of clusters for all choices of number of sequences per cluster. A deterioration of the performance is verified again in Figure 7 where $Tp = 2$. Again, the percentages of correct decisions are mainly zero for the majority of gap values as in the case when we used $Tp = 0.25$. Finally, it is verified by the clustering indices between the found clusters and the original ones, for the cases where the method performed best in Figures 5 ($g \in [0.3\ 0.5]$) and 6 ($g \in [0.7\ 0.9]$), that the correct clusters are obtained. The medians and averages of the clustering indices are always one or almost one whereas the standard deviations are zero or almost zero.

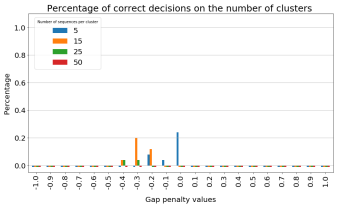

Fig. 4. Percentage of correct decisions for $Tp = 0.25$.

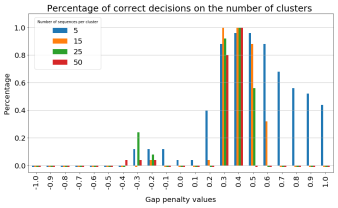

Fig. 5. Percentage of correct decisions for $Tp = 1$.

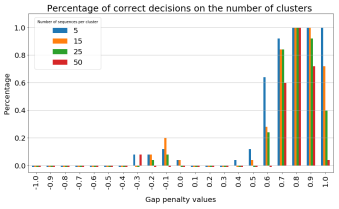

Fig. 6. Percentage of correct decisions for $Tp = 1.5$.

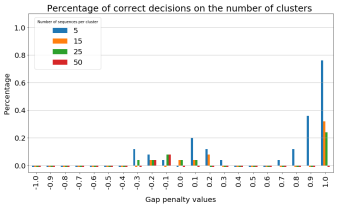

Fig. 7. Percentage of correct decisions for $Tp = 2$.

In this experiment, it was shown that by increasing the parameter Tp, good performances can be obtained. The analysis suggests that the range of gap values used in the proposed method should be changed, for instance, to only use positive gap values. In this manner, the method could correctly separate the clusters in this dataset.

B. Reuma.pt dataset

In this section, two experiments are set to study the application of the proposed method with the Reuma.pt data. The first experiment consists on performing alignment with sequences created with the patient treatment history whereas the second experiment is meant to study the progression of a very important feature in rheumatoid arthritis, the patient's DAS28 evolution.

1) Experiment I - Biological sequences: In this first experiment, the sequences that are created are based on the patients treatment history. There is information about therapies followed by 424 patients from a total of 426. After creating the temporal sequences, the remaining steps of Algorithm 3 can be applied. Regarding the different parameters on the method, the parameters of TNW algorithm, initially, are the same as presented in the previous chapter. The parameters of Algorithm 1 are the number of bootstrap samples, $M = 250$, as introduced before, and the maximum number of clusters K to be analysed is dependent on the number of patients to be clustered. Since there are 426 patients, a value of $K = 20$ is reasonable to use as default for all experiments.

The most relevant result is obtained by using the Ward's distance metric on the agglomerative clustering. For gap values bigger or equal to zero, the correct number of clusters found is usually small. More concretely, for gap values between 0 and 0.8 the number of clusters found is 2 whereas for gap values of 0.9 and 1 the analysis indicated 6 as the correct number of clusters. Since we are not interested in small number of clusters, several values for the temporal penalty constant Tp of the TNW algorithm are tested in combination with different values of the gap penalty. Figure 8 shows the dendrogram obtained when using a gap value of 0.3 and temporal penalty constant of 2.

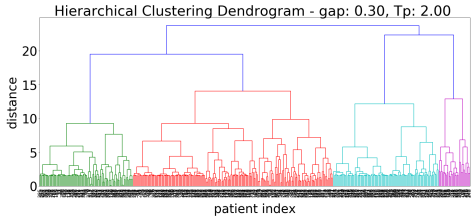

Fig. 8. Dendrogram of Ward's method hierarchical clustering with $g = 0.3$ and $Tp = 2$.

The dendrogram suggests the existence of distinct groups of patients. The average values of the five clustering indices obtained with Algorithm 1 are presented in Table IV. In this case not all average values point to the same number of clusters k and therefore a more careful analysis is required. Three of the measures, AR, FM and Jaccard indicate $k = 3$ as the correct number of clusters. The AW measure favours $k = 18$ whereas the Rand index which is not as relevant as the AR points to $k = 19$ and $k = 20$. Considering only these values, a final decision would be $k = 3$ as the correct number of clusters, since it is the option where most of the measures agree. However, the 3 clusters were inconclusive. A distinct analysis is that the AR and AW for $18 \leq k \leq 20$ also have relatively high values and close to the ones for $k = 3$. Hence, AR and AW also favours the decision for $18 \leq k \leq 20$, which is indeed more realistic given the inherent putative clinical strata. To complement this analysis, let us analyse the standard deviation of the AR presented in Figure V-B1. The minimum AR standard deviation value is achieved for $k = 18$.

Therefore, by combining the information provided by Table IV and Figure V-B1, we choose $k = 18$. Note that we are not interested in small number of clusters, since it was already verified that they do not provided us relevant information. The complete patient's 18 clusters found are fully presented in https://github.com/KishanRama/AliClu/tree/master/Results.

The first observation on the found clusters is that they are balanced. Separation based on the events is achieved, hence, each cluster has markedly distinct sequences. Additionally, the clusters are divided based on the timing information as well. As an example, there several clusters "AZ" successfully divided based on the temporal information. The clusters 7, 12, 13, 15, 16, 17, 18 contain sequences of type "AZ" (not exclusively) where the median of the intervals of time of the

k	Rand	AR	FM	Jaccard	AW
2	0.806	0.596	0.842	0.736	0.639
3	0.857	**0.716**	**0.859**	**0.768**	0.678
4	0.874	0.709	0.802	0.678	0.689
5	0.86	0.622	0.716	0.562	0.686
6	0.872	0.633	0.717	0.564	0.642
7	0.889	0.646	0.715	0.561	0.651
8	0.895	0.631	0.695	0.537	0.609
9	0.905	0.608	0.664	0.501	0.631
10	0.916	0.611	0.66	0.494	0.657
11	0.926	0.628	0.671	0.508	0.657
12	0.937	0.661	0.697	0.54	0.686
13	0.942	0.671	0.704	0.549	0.68
14	0.946	0.679	0.709	0.554	0.684
15	0.949	0.685	0.714	0.56	0.68
16	0.95	0.685	0.713	0.559	0.661
17	0.952	0.684	0.712	0.556	0.647
18	0.959	0.707	0.73	0.578	**0.703**
19	**0.961**	0.71	0.731	0.579	0.697
20	**0.961**	0.707	0.729	0.576	0.684

TABLE IV
AVERAGE VALUES OF FIVE CLUSTERING INDICES FOR THE DENDROGRAM OF FIGURE 8.

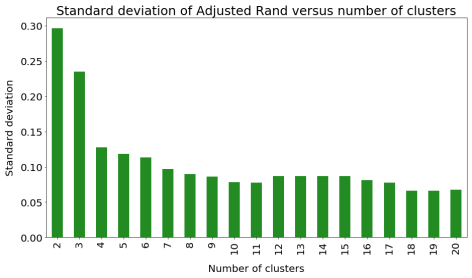

Fig. 9. Standard deviation of AR versus number of clusters for the dendrogram of Figure 8.

sequences between "A" and "Z" is given by 1109, 2288, 106, 362, 1745, 676.5 and 3605, respectively. Another interesting observation is that different type of sequences (not only "AZ") mix up with these new clusters. For instance, in cluster 17, patient with id 69330487 has sequence "0.A,504.B,246.Z" and appears in this cluster because the time from "A" to "Z" is in the time interval correspondent to the sequences of this cluster. This patient can be compared to others on the cluster that took a medicine "A" during 500 days, more or less, and that now might switch to other medicine as observed on the sequence. Basically, if there other patients similar to patient with id 69330487 based on some other characteristics and that are taking only medicine "A", it can be interesting too study why there was a switch of medicine at this particular patient and if the same can be done to others. Regarding clusters containing other type of sequences, the separation is also noted. Hence, in this experiment, separation based on the timing information is well succeeded.

Lastly, the stability of the 18 clusters is assessed through the median, average and standard deviation of the three measures presented in Table V. The median, average and standard deviation values of η^* are always smaller than of τ^* and γ^*, as expected. For several clusters the medians and averages of the three measures are not as high for what we expect for stable clusters. Another observation is that the median and averages of τ^* and γ^* do not agree in all clusters as was observed in other results. In Clusters 1, 2, 3, 10, 16 and 18, those values

agree and they are relatively high to consider them as stable.

Cluster Number (# patients)	τ^* median	η^* median	γ^* median	τ^* average	η^* average	γ^* average	τ^* std	η^* std	γ^* std
1 (2)	1	0.5	1	0.596	0.313	0.74	0.454	0.213	0.314
2 (9)	0.667	0.4	0.667	0.53	0.312	0.685	0.333	0.161	0.189
3 (9)	0.667	0.4	0.667	0.663	0.392	0.668	0.17	0.066	0.168
4 (14)	0.474	0.321	0.643	0.479	0.315	0.635	0.167	0.078	0.15
5 (16)	0.333	0.25	0.5	0.344	0.25	0.506	0.122	0.067	0.127
6 (16)	0.353	0.261	0.438	0.372	0.263	0.443	0.143	0.074	0.132
7 (18)	0.424	0.298	0.722	0.457	0.307	0.723	0.15	0.066	0.109
8 (19)	0.5	0.333	0.579	0.503	0.324	0.598	0.184	0.084	0.17
9 (19)	0.407	0.289	0.526	0.423	0.289	0.542	0.159	0.078	0.148
10 (20)	0.75	0.429	0.75	0.72	0.416	0.72	0.107	0.038	0.107
11 (23)	0.423	0.297	0.478	0.439	0.297	0.497	0.159	0.077	0.134
12 (23)	0.597	0.374	0.739	0.559	0.348	0.727	0.193	0.088	0.12
13 (24)	0.458	0.314	0.458	0.484	0.321	0.496	0.129	0.054	0.126
14 (25)	0.519	0.341	0.56	0.509	0.332	0.57	0.137	0.064	0.127
15 (25)	0.569	0.363	0.64	0.558	0.351	0.629	0.165	0.07	0.14
16 (48)	0.648	0.393	0.719	0.643	0.389	0.714	0.104	0.04	0.08
17 (55)	0.496	0.331	0.509	0.517	0.337	0.533	0.125	0.053	0.126
18 (59)	0.618	0.382	0.627	0.608	0.375	0.621	0.104	0.042	0.103

TABLE V
STABILITY OF 18 CLUSTERS FOUND WHEN USING WARD'S METHOD WITH $g = 0.3$ AND $Tp = 2$.

2) Experiment II - DAS28 sequences: In the second experiment we use the clinical history of the DAS28 measurements to create temporal sequences. The temporal sequences created in this experiment are usually longer than the previous experiment. The results obtained by running our proposed method on this longer sequences were inconclusive. In order to obtain interpretable results the longer sequences are compressed. This avoids events from appearing more than one time in a row.

After obtaining the compressed temporal sequences, the rest of the steps of Algorithm 3 are performed.

We run several tests, which we do not present here, where the different parameters of the algorithm were varied, however, he interpretation of the clusters found was much harder than in the previous experiment with biological therapy sequences. The previous experiment had sequences that were directed. In this experiment, since we are creating sequences based on DAS28 values, it may occur that a patient starts with high values of DAS28 then it changes to a good state of the disease with low values of DAS28 followed again by high values. Hence, the type of sequences that can appear in this experiment are much more complex than in the previous experiment.

Overall, the patients are split based on the type and length of the sequence and it is difficult to assess if there is an effectively separation of the sequences based on the temporal information.

VI. CONCLUSIONS AND FUTURE WORK

We proposed a method, named AliClu, that combines temporal sequence alignment and agglomerative clustering with a validation technique to find clusters in longitudinal data.

Synthetic datasets were generated to test the performance of the proposed method. For the datasets that contained 2 clusters generated by continuous-time Markov chains with very discrepant parameters, i.e. well separated clusters, our method successfully found the correct clusters with percentages of correct decisions above or equal to 80%. Throughout the experiments with synthetic data, it was found that ward's method is the linkage method of choice for the agglomerative clustering. The method was further evaluated with a real dataset, the Reuma.pt dataset. We run our method in a non-automatic manner for this dataset, where the results of the alignments and clusterings were analysed individually for each value of the gap penalty of the TNW algorithm. Also, several values of the temporal penalty constant were tested to obtain the best results. We discovered 18 clusters in the experiment with biological sequences where successful separation based on the events and temporal information between them was achieved. However, the experiment with DAS28 did not yield clusters that could be interpreted easily, which might have to do with the existence of long sequences.

Regarding future work, further investigation is required in tuning the parameters of the method. The synthetic experiment with 4 clusters clearly suggests that instead of varying the gap penalty from -1 to 1, as in the original proposed method, we should compress that interval and use only positive gap values, for instance. We also conclude that using a temporal penalty constant of 0.25 only works for very simple cases. Despite cluster analysis not being usually automatic, but more an iterative process of trial and error, there is room for improvements to make the method fully automatic. This would mean finding clusters automatically in any dataset, where the sequences in each group are similar in terms of events and temporal information between them. The proposed method has the potential to be an useful and effective clinical tool for unravelling patterns from longitudinal data collected in medical appointments.

ACKNOWLEDGMENT

This research was supported by the FCT project NEUROCLINOMICS2 (PTDC/EIA-EIA/111239/2009) and Instituto de Telecomunicações.

REFERENCES

[1] J. Davis, E. Lantz, and D. P. et al., "Machine learning for personalized medicine: Will this drug give me a heart attack," 2008. Proceedings of International Conference on Machine Learning (ICML).

[2] G. H. Fernald, E. Capriotti, and R. D. et al., "Bioinformatics challenges for personalized medicine," *Bionformatics*, vol. 27, pp. 1741–1748, May 2011.

[3] H. Canho, A. Faustino, and F. M. et al., "Reuma.pt - The Rheumatic Diseases Portuguese Register," *Acta Reumatologica Portuguesa*, vol. 36(1), pp. 45–56, Jan. 2011.

[4] S. B. Needleman and C. D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology*, vol. 48, pp. 443–453, 1970.

[5] H. Syed and A. K. Das, "Temporal Needleman-Wunsch," *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Dec. 2015. doi:10.1109/DSAA.2015.7344785.

[6] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846–850, Dec. 1971.

[7] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, Dec. 1985.

[8] E. B. Fowlkes and C. Mallows, "A method for comparing two hierachical clusterings," *Journal of The American Statistical Association*, vol. 78, pp. 553–569, Sep. 1983.

[9] D. L. Wallace, "A method for comparing two hierachical clusterings: Comment," *Journal of The American Statistical Association*, vol. 78, pp. 569–576, Sep. 1983.

[10] A. Severiano, F. Pinto, M. Ramirez, and J. Carrio, "Adjusted wallace coefficient as a measure of congruence between typing methods," *Journal of clinical microbiology*, vol. 49, pp. 3997–4000, Sep. 2011.

[11] H.-J. Mucha, "On validation of hierarchical clustering," in *Advances in Data Analysis* (R. Decker and H. J. Lenz, eds.), (Berlin, Heidelberg), pp. 115–122, Springer Berlin Heidelberg, 2007.