ARTEMIS: Assessing the Similarity of Event-Interval Sequences

Orestis Kostakis, Panagiotis Papapetrou, and Jaakko Hollmén

Department of Information and Computer Science, Aalto University, Finland.
Helsinki Institute for Information Technology, Finland.

Abstract. In several application domains, such as sign language, medicine, and sensor networks, events are not necessarily instantaneous but they can have a time duration. Sequences of interval-based events may contain useful domain knowledge; thus, searching, indexing, and mining such sequences is crucial. We introduce two distance measures for comparing sequences of interval-based events which can be used for several data mining tasks such as classification and clustering. The first measure maps each sequence of interval-based events to a set of vectors that hold information about all concurrent events. These sets are then compared using an existing dynamic programming method. The second method, called `Artemis`, finds correspondence between intervals by mapping the two sequences into a bipartite graph. Similarity is inferred by employing the Hungarian algorithm. In addition, we present a linear-time lower-bound for `Artemis`. The performance of both measures is tested on data from three domains: sign language, medicine, and sensor networks. Experiments show the superiority of `Artemis` in terms of robustness to high levels of artificially introduced noise.

Keywords: Event-interval sequence, distance measure, Dynamic Time Warping, Hungarian algorithm

1 Introduction

Sequences of temporal intervals exist in many application domains, such as human motion databases, sign language, human activity monitoring, and medicine. Their main advantage over traditional sequences, which model series of instantaneous events, is that they incorporate the notion of duration in their event representation scheme. Due to this, they are used in a broad range of fields such as geo-informatics [29], cognitive science [4], linguistic analysis [5], music [24], and medicine [12]. Essentially, a sequence of event-intervals corresponds to a collection of labelled events accompanied by their start and end time values. We will call such sequence, *event-interval sequence*, and each labelled interval, *event-interval*. An example of such sequence—containing five event-intervals—is shown in Figure 1.

So far, most studies on event-interval sequences have been focusing on the aspect of knowledge discovery, such as mining patterns and association rules that

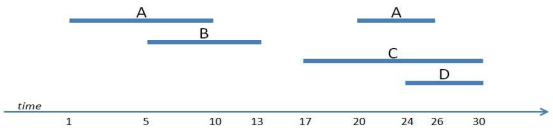


Fig. 1: An example of a sequence of interval-based events. This representation corresponds to the following sequence of event-intervals: $\mathcal{S} = \{(A, 1, 10), (B, 5, 13), (C, 17, 30), (A, 20, 26), (D, 24, 30)\}$.

might be of interest to domain experts [2, 11]. Surprisingly, very limited attention has been given on assessing the similarity of event-interval sequences [13]. Robust similarity measures for comparing such sequences would enable the utilization of existing clustering and classification methods, introduce new index structures for similarity search in event-interval sequence repositories, and would facilitate the implementation of recommendation systems and assistive applications.

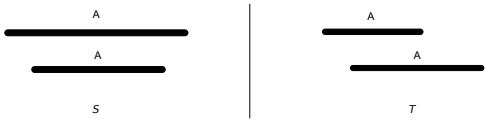


Fig. 2: Two different sequences of interval-based events where the mapping to a sequence of instantaneous events produces the same representation for both.

Existing similarity measures on symbolic sequences or time series are not directly applicable to event-interval sequences [13]. One could, for example, convert a sequence of event-intervals to a sequence of instantaneous events by only considering the start and end points of each event interval, and associating each of the two points with the same event label. This would result in a simplification of the representation of these sequences as they would be mapped to traditional sequences of instantaneous events. Thus, the solution to the problem would reduce to applying an existing distance/similarity measure for sequence matching, such as edit distance [16]. Nonetheless, this solution suffers from several shortcomings. Firstly, the size of the sequences and the alphabet (i.e., set of all possible event labels) would double since each event-interval label would be mapped to two instantaneous event labels. Secondly, crucial information about the pairwise temporal relations between the event-intervals in the sequence will be lost. Consider the example shown in Figure 2, where each sequences consists of two event-intervals with the same label. Obviously, the mapping for both sequences is the same, i.e., $\{A_{start}, A_{start}, A_{end}, A_{end}\}$. The relation between the two event intervals, however, is different. Hence, we can deduce that in order to provide a robust similarity measure for such sequences, their representation should include additional information about the relations between the event-intervals.

Our contributions. In this paper, we formulate the problem of comparing sequences of event-intervals and show that solving the problem by directly mapping it to string matching fails to capture temporal dependencies between the interval-based events. In addition, we propose two distance measures to solve this problem: the first one maps the event-interval sequence to a sequence of vectors that can also be seen as a multi-dimensional time series. The second one attempts to identify correspondence between intervals of the two event-interval sequences, by taking into account the temporal relations that may occur between the events, and then employing minimum-weight bipartite matching to infer a similarity score for the sequences. Moreover, we propose a lower bound for the second method that can achieve significant speed-ups during similarity search. Finally, we present an extensive experimental evaluation on real datasets from three different domains. The methods are benchmarked with respect to their robustness to noise, nearest neighbor classification accuracy, and scalability. In addition, we study the pruning power and tightness of the proposed lower bound.

2 Event-Interval Sequences

Sequences of interval-based events can be represented in many ways. In this paper, we use the *e-sequence* [26, 27] representation, which explicitly considers the start and end time stamps of each event.

Definition 1 ***(E-Sequence)*** *Given an alphabet σ of event labels, an* event-interval sequence *or* e-sequence $\mathcal{S}=\{S_1, \ldots, S_n\}$ *is an ordered set of events occurring over time intervals. Each $S_i = (E, t_{start}, t_{end})$ is called* event-interval*, where $S_i.E \in \sigma$, and $S_i.t_{start}$, $S_i.t_{end}$ denote the start and end time of $S_i.E$, respectively. Note that, we use $S_i.X$ to denote element X of S_i.*

Note that an event-interval corresponds to an event that occurs over a time interval. The temporal order of the event-intervals in an e-sequence is ascending based on their start time and in the case of ties it is descending based on their end time. If ties still exist, alphabetical ordering is applied based on labels. An example of an e-sequence is shown in Figure 1, and it corresponds to:

$$\mathcal{S} = \{(A, 1, 10), (B, 5, 13), (C, 17, 30), (A, 20, 26), (D, 24, 30)\}.$$

For simplicity, in this paper, we are only interested in the types of temporal relations between event-intervals in an e-sequence, and not in the actual duration of each event-interval. Let $\mathcal{I} = \{r_1, \ldots, r_{|\mathcal{I}|}\}$ be the set of all legal temporal relations that can exist between any pair of event-intervals. Based on Allen's model for interval temporal logic [3], we define $\mathcal{I}$ by considering seven types of relations between two event-intervals: *meet*, *match*, *overlap*, *contain*, *left-contain*, *right-contain*, *follow*. These relations are shown in Figure 3 and are described in detail by Papapetrou et al. [27]. Hence, $\mathcal{I} = \{meet, match, overlap, contain, left\text{-}contain, right\text{-}contain, follow\}$.

According to our formulation the same event label can occur many times within an e-sequence. Effectively, this allows two event-intervals of the same

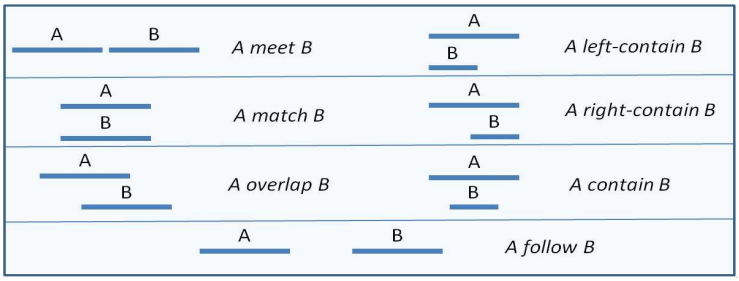


Fig. 3: The seven temporal relations between two event-intervals that are considered in this paper.

label to overlap, which is a perfectly acceptable scenario, for example, in the case of nested for-loops in programming languages.

Using the above definitions, the problem studied in this paper can be formulated as follows:

Problem 1. **(E-sequence Distance)** Given two e-sequences $\mathcal{S}$ and $\mathcal{T}$, define a distance measure D, such that $\forall \mathcal{S}, \mathcal{T}$ it holds: $D(\mathcal{S}, \mathcal{T}) \geq 0$, $D(\mathcal{S}, \mathcal{S}) = 0$, and $D(\mathcal{S}, \mathcal{T}) = D(\mathcal{T}, \mathcal{S})$.

The degree to which $\mathcal{S}$ and $\mathcal{T}$ differ should be reflected in the value of $D(\mathcal{S}, \mathcal{T})$ and should be in accordance with the knowledge obtained from domain experts.

3 Distance Measures

In this section, we define two distance measures for comparing e-sequences and propose a lower bound for the second measure.

3.1 The Vector-based DTW Distance

The first method employs a vector-based representation of e-sequences. For each e-sequence, a set of vectors is defined, where each vector indicates which and how many event-interval labels are active at specific time points in the e-sequence. The selected time points are all the instances in which an event-interval begins or ends.

Definition 2 ***(Event Vector)*** *Given an e-sequence $\mathcal{S}$ defined over an alphabet σ, an* event vector $V^t = (V_1^t, \ldots, V_{|\sigma|}^t)$ *consists of integer values, where each V_i^t records the number of occurrences of event E_i at time stamp t in $\mathcal{S}$.*

Hence, an e-sequence $\mathcal{S}$ can be mapped to an ordered set of event vectors $\mathcal{V}_\mathcal{S} = \{V^{t_0}, V^{t_1}, \ldots, V^{t_m}\}$. The set of time stamps $\{t_1, \ldots, t_m\}$ includes all time points in $\mathcal{S}$ where the "status" of at least one event-interval changes, i.e., an

event-interval starts or ends. V^{t_0} is the null vector which denotes the initial condition at t_0 where no event takes place.

Example. Consider the e-sequence $\mathcal{S}$ shown in Figure 1. Given that $|\sigma| = 4$, the size of each event vector for $\mathcal{S}$ is also 4. The set of event vectors of $\mathcal{S}$ is defined as follows: $\mathcal{V}_\mathcal{S} = \{(0,0,0,0),(1,0,0,0),(1,1,0,0),(0,1,0,0),(0,0,0,0),(0,0,1,0),(1,0,1,0),(1,0,1,1),(0,0,1,1),(0,0,0,0)\}$.

Given two e-sequences $\mathcal{S}$ and $\mathcal{T}$, their vectors can also be seen as $|\sigma|$-dimension al time series; thus, their distance can be computed using Dynamic Time Warping (DTW) [14]. Vectors are compared using the $\mathcal{L}_1$ norm.

Despite the simplicity of this method, it only takes into account the temporal ordering of the event-intervals but does not explicitly consider any temporal relations. This may cause ambiguities in the representation by mapping two different e-sequences to the same set of event vectors.

Complexity. The sets of event vectors $\mathcal{V}_\mathcal{S}$ and $\mathcal{V}_\mathcal{T}$ can be computed in linear time. The time complexity of this DTW computation is $O(|\mathcal{V}_\mathcal{S}||\mathcal{V}_\mathcal{T}||\sigma|)$. Significant speedups in multidimensional time series similarity search under DTW can be achieved using existing lower bounding techniques [31].

3.2 `Artemis`: A Bipartite-based Matching Distance

The second method, called `Artemis`is based on determining *correspondence* between pairs of event-intervals to infer the overall similarity of e-sequences. Given two e-sequences, correspondence is determined by the fraction of common relations between event-intervals in the e-sequences. The overall similarity score is derived from the sum of pairwise scores using the Hungarian algorithm (also known as Kuhn-Munkres algorithm) [23]. `Artemis` consists of two main steps: (a) the mapping step and (b) the matching step.

The mapping step. The first step of `Artemis` is to map each e-sequence $\mathcal{S}$ to an ordered set of temporal relations between event-intervals. Given the set of legal temporal relations between event-intervals $\mathcal{I}$ and two event-intervals S_i and S_j, $r(S_i, S_j)$ is the *event-interval relation* between S_i and S_j, with $r(S_i, S_j) \in \mathcal{I}$.

More specifically, for each event-interval $S_i \in \mathcal{S}$ we record the set of relations of S_i with $S_j \in \mathcal{S}, \forall j \neq i$. Three sets of relations are computed:

- $r_L(S_i) = \{r(S_j, S_i) | 1 \leq j < i\}$: which contains the temporal relations of S_i with all event-intervals located on the left side of S_i in $\mathcal{S}$,
- $r_R(S_i) = \{r(S_i, S_j) | i < j \leq |S|\}$: which contains the temporal relations of S_i with all event-intervals located on the right side of S_i in $\mathcal{S}$, and
- $r_\varnothing(S_i) = \{r(\varnothing, S_i)\}$: which is a singleton with a *follow* relation between $\varnothing$—an extra symbol such that $\varnothing \notin \sigma$—and S_i.

In addition, let $r_{\varnothing L}(S_i) = r_\varnothing(S_i) \cup r_L(S_i)$. Note that symbol $\varnothing$ is introduced so that event labels are also taken into account: e-sequences that differ in event-interval relations but share similar event labels are assigned with smaller distance values than e-sequences that differ in both event labels and event-interval relations.

The matching step. Given two e-sequences $\mathcal{S}$ and $\mathcal{T}$, the matching step of `Artemis` computes a distance value for each pair of event-intervals $S_i \in \mathcal{S}$ and $T_j \in \mathcal{T}$. The key idea behind this computation is to count the number of common relations between $r_{\varnothing L}(S_i)$ and $r_{\varnothing L}(T_j)$—the relations of S_i and T_j with all event-intervals located on their left side, including $\varnothing$—and the common relations between $r_R(S_i)$ and $r_R(T_j)$—the corresponding relations on their right side.

More formally, the *event-interval distance*, denoted as d_m, between two event-intervals $S_i \in \mathcal{S}$ and $T_j \in \mathcal{T}$ is defined as follows:

$$d_m(S_i, T_j) = \begin{cases} \dfrac{\max\{|\mathcal{S}|, |\mathcal{T}|\} - |r_{\varnothing L}(S_i) \cap r_{\varnothing L}(T_j)| - |r_R(S_i) \cap r_R(T_j)|}{\max\{|\mathcal{S}|, |\mathcal{T}|\}} & \text{, if } S_i.E = T_j.E \\ 1 & \text{, if } S_i.E \neq T_j.E \end{cases}$$

Let $D_{\mathcal{S},\mathcal{T}}$ be an $|\mathcal{S}| \times |\mathcal{T}|$ matrix, with $D(i,j) = d_m(S_i, T_j)$, $S_i \in \mathcal{S}$ and $T_j \in \mathcal{T}$. We call $D_{\mathcal{S},\mathcal{T}}$ the *event-interval distance matrix* of $\mathcal{S}$ and $\mathcal{T}$. Problem 1 reduces to the following optimization problem:

Problem 2. **(The Assignment Problem)** Given $\mathcal{S}$, $\mathcal{T}$, and $D_{\mathcal{S},\mathcal{T}}$, assign each event-interval in $\mathcal{S}$ to exactly one event-interval in $\mathcal{T}$ so that the total assignment cost is minimized.

Problem 2 can be solved by the Hungarian algorithm. Let the output of the algorithm be the following set $\mathcal{H}(\mathcal{S},\mathcal{T}) = \{h(S_1), \ldots, h(S_{|\mathcal{S}|}\}$ with an assignment cost $C(\mathcal{S},\mathcal{T})$. Each $h(S_i) \in \mathcal{H}(\mathcal{S},\mathcal{T})$ denotes the event-interval in $\mathcal{T}$ that $S_i \in \mathcal{S}$ is matched to by the Hungarian algorithm. The assignment cost $C(\mathcal{S},\mathcal{T})$ corresponds to the distance—called *Artemis Distance*—between $\mathcal{S}$ and $\mathcal{T}$.

Definition 3 ***(Artemis Distance)*** *Given $\mathcal{S}$, $\mathcal{T}$, $D_{\mathcal{S},\mathcal{T}}$, and $\mathcal{H}(\mathcal{S},\mathcal{T})$, the* `Artemis` *distance of $\mathcal{S}$ and $\mathcal{T}$ is defined as follows:*

$$\texttt{Artemis}(\mathcal{S},\mathcal{T}) = \sum_{i=1}^{\max\{|\mathcal{S}|,|\mathcal{T}|\}} D(S_i, h(S_i)),\ S_i \in \mathcal{S},\ h(S_i) \in \mathcal{H}(\mathcal{S},\mathcal{T}). \quad (1)$$

To handle the case of e-sequences of different size, "dummy" event-intervals are added with distance 1 *from all other event-intervals.*

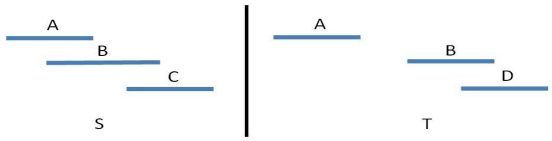


Fig. 4: Two e-sequences $\mathcal{S}$ and $\mathcal{T}$ used as an example for `Artemis`.

Example. Figure 4 shows two e-sequences $\mathcal{S}$ and $\mathcal{T}$. At the *mapping step*, for each event-interval $S_i \in \mathcal{S}$ we compute $r_{\varnothing L}(A) = \{follow(\varnothing, A)\}$, $r_R(A) =$

$\{overlap(A,B),\ follow(A,C)\}, r_{\varnothing L}(B) = \{follow(\varnothing,B),\ overlap(A,B)\}, r_R(B) = \{overlap(B,C)\}$, and $r_{\varnothing L}(C) = \{follow(\varnothing,C),\ follow(A,C),\ overlap(B,C)\}$. For $\mathcal{T}$, we compute $r_{\varnothing L}(A) = \{follow(\varnothing,A)\}, r_R(A) = \{follow(A,B),\ follow(A,D)\}$, $r_{\varnothing L}(B) = \{follow(\varnothing,B),\ follow(A,B)\}$, $r_R(B) = \{overlap(B,D)\}$, and $r_{\varnothing L}(D) = \{follow(\varnothing,D),\ follow(A,D),\ overlap(B,D)\}$. At the *matching step*, the Hungarian algorithm gives $\mathcal{H}(\mathcal{S},\mathcal{T}) = \{A,B,D\}$. Finally, $Artemis(\mathcal{S},\mathcal{T}) = (2/3 + 2/3 + 1) = 7/3$.

Complexity. Let $m = max(|\mathcal{S}|,|\mathcal{T}|)$. Then, at the mapping step, $O(m^2)$ relations are enumerated, while the complexity of computing $D(\mathcal{S},\mathcal{T})$ using hash-tables is $O(m^3)$. The cost of applying the cubic Hungarian algorithm to the two event-interval relation sets results to a total time complexity of $O(m^3)$. A lower bound for speeding up the computation of `Artemis` is described next.

3.3 Lower Bounding `Artemis`

The proposed lower bound can be computed in linear time and is based on the comparison of event label counts. By knowing the number of labels in which two e-sequences differ, we can determine a lower bound for their `Artemis` distance.

Given an e-sequences $\mathcal{S}$, we define an $|\sigma|$-dimensional vector $v^{\mathcal{S}}$, that stores, for each event label in σ, the count of event-intervals in $\mathcal{S}$ that share that label.

Theorem 1 *Given $\mathcal{S}$ and $\mathcal{T}$, the lower bound of $Artemis(\mathcal{S},\mathcal{T})$ is defined as*

$$Artemis_{LB}(\mathcal{S},\mathcal{T}) = \frac{k}{2} + \left(m - \frac{k}{2}\right)\left(\frac{k}{2m}\right) = k - \frac{k^2}{4m}, \tag{2}$$

where $k = ||v^{\mathcal{S}} - v^{\mathcal{T}}||_1$ and $m = max(|\mathcal{S}|,|\mathcal{T}|)$.

Proof. Knowing that $||v^{\mathcal{P}} - v^{\mathcal{Q}}||_1 = k$ we can be sure that the distance is at least $k/2$; those k event-intervals are matched with each other giving a score of 1 for each of the $k/2$ pairs. If $k/2$ is equal to the length of the e-sequences, then it is also their distance. However, if the differences refer to only a subset of all event-intervals, then these differences are reflected in the matching scores of the rest of the event-intervals. So, given that $m = max(|\mathcal{S}|,|\mathcal{T}|)$ and $m > \frac{k}{2}$, the rest of the $m - k/2$ event-intervals would have at least $k/2$ non-common relations. Thus, yielding an additional distance of $(m - k/2) \cdot (k/2m)$.

The proposed lower bound focuses on label counts and not on relations of event-intervals. When the differences of two e-sequences are restricted to event-interval labels, the lower bound is equal to the distance obtained by *Artemis*. On the other hand, when the e-sequences share the same event labels and differ only in the type of event-interval relations, then the lower bound yields zero score. The tightness and pruning power of the lower bound is studied on three datasets in Section 4.

4 Experiments

The performance of the proposed methods has been benchmarked on three real datasets. We studied their robustness to noise, classification accuracy, and scalability.

4.1 Experimental Setup

The proposed methods have been benchmarked on three real datasets (Table 1):

- ***Hepatitis***[28] The dataset contains information about 498 patients who have either Hepatitis B or Hepatitis C. The intervals represent the results of 25 regular tests.
- ***ASL***[27] The dataset contains 873 transcriptions from videos of American Sign Language expressions provided by Boston University. Each e-sequence in the dataset is one utterance expressed using American Sign Language.
- ***Pioneer***[22] This dataset was constructed from the Pioneer-1 dataset available in the UCI repository[1]. It contains time series sensor readings of the Pioneer-1 mobile robot. The data is segmented into event-intervals in which the robot takes action for some period of time.

The proposed methods were evaluated with respect to robustness against two types of artificial noise, k-NN classification accuracy, and scalability. In addition, the efficiency of the lower bound was tested by computing its *tightness* and its *pruning power* against 1-NN queries.

Dataset	# of	# of	e-sequence size			# of	# of
	e-sequences	intervals	min.	max.	average	labels	classes
ASL	873	15675	4	41	18	216	5
Hepatitis	498	53921	15	592	108	147	2
Pioneer	160	8949	36	89	56	92	3

Table 1: Dataset Statistics.

Robustness. The robustness of the proposed methods was tested on two types of artificial noise: *shifts* and *swaps*. In the first case, each interval within an e-sequence is shifted, with a certain *shift probability* value p, by an offset back or forth in time. Given a *distortion level* d as a percentage of the length of the whole sequence, a random value under the uniform distribution is chosen in that integer interval to determine the offset. An event-interval has equal probability to be shifted either back or forth in time, while interval durations remain unaffected. This type of artificial noise attempts to simulate noisy sources or recording devices. Real-world cases for this could include humans who learn sign language or who rehabilitate from brain injuries. Both p and d were set between 0.2 and 1, with step 0.2.

[1] http://archive.ics.uci.edu/ml/

One drawback of employing such artificial noise is that shifting intervals could result to semantic invalidity, e.g., "robot walks forward" overlaps with "robot walks backwards". To avoid such cases, we further experiment with artificial noise which is based on *swaps* of event-interval labels, while the durations and relations of the event-intervals remain unaffected. Given an event-interval, the *swap probability* parameter determines if its label is swapped with the label of another event-interval which is chosen uniformly at random from the whole e-sequence. The swap probability parameter values ranged between 0.2 and 1 with step 0.2. Note that, noise is added off-line and that both methods were tested on exactly the same distorted sequences; to rule out score differences caused by the randomness factor.

Using the above two techniques, noise is inserted into e-sequences which then serve as the queries for 1-NN search. Given a database of e-sequences, a copy of an e-sequence is distorted, based on the parameter values, and then its nearest neighbor is found by scanning the database. This is performed for each e-sequence in the database. Ideally, we would like each noisy e-sequence (query) to be matched to the e-sequence from which it originated. We compared the DTW vector-based measure and `Artemis` in terms of: *retrieval accuracy* (the fraction of noisy queries for which the originating e-sequence is retrieved) and *rank of nearest neighbor* (for each query, the number of database e-sequences with distance less than or equal to that of the originating counterpart).

Classification. For the k-NN classification experiments, we studied the efficiency of the proposed methods under 1-NN and 3-NN classifiers. Due to the fact that the e-sequences in the *ASL* dataset can belong to up to 5 classes (wh-question, etc) simultaneously, and some elements do not have any label (simple affirmative phrases), we had to modify the algorithm. The *ASL* classifier returns a score in $[0, 1]$ denoting the average ratio of common class-labels with its k nearest neighbors. For example, a phrase with labels $\{a\}$ and neighbors with label sets $\{a\}$, $\{a, b\}$, and $\varnothing$ respectively, would yield $(1 + 0.5 + 0)/3$. For the *ASL* dataset, the result is the average sum.

We designed an additional experiment specifically for the *ASL* dataset. There exist in total 288 interval sequences in the dataset which portray exactly the same phrase in English with one or more other sequences. We studied what ratio of the 288 equivalent sequences was retrieved by examining the k nearest neighbors of every sample, for all possible values of k. Two values were monitored: the first corresponds to the ratio of the 288 sequences for which an identical phrase can be detected within their k nearest neighbors; the second is the ratio of the total sum of identical phrases detected within the k-nearest neighbors over the total number of pairs of identical labels. The two values would be the same if any phrase in English were to be represented only by at most two e-sequences in the dataset. This experiment was designed to investigate the suitability of the methods in the field of sign language and relevant application domains.

Scalability. In the attempt to evaluate the scalability of the proposed algorithms, we embedded time-monitoring functions in our implementations. For each e-sequence, we counted the time needed to map the e-sequence to the ap-

propriate representation (set of event vectors for DTW and event-interval relation set for `Artemis`) and then to compare it against the whole database. The dataset, serving as the database, had already been transformed to the appropriate form. For the case of `Artemis`, we did not implement the use of hash-tables, thus the complexity is $O(n^4)$ w.r.t. the size of the e-sequences. Our experiments were implemented in Java and were performed on a PC running Ubuntu Linux, equipped with Intel Core 2 Duo 2GHz CPU and 4GB RAM.

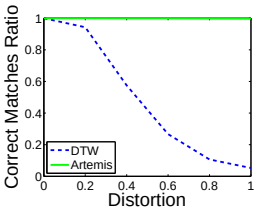
Lower bounding. To assess the quality of the lower bound, we computed its pruning power for 1-NN queries in the database and its tightness. The pruning power is defined as the ratio of the number of pruned e-sequences, using $Artemis_{LB}$, over the total number of comparisons that would have been required if the database were to be scanned sequentially. The tightness is defined as the average ratio of the lower bound distance over the distance given by `Artemis`.

4.2 Results

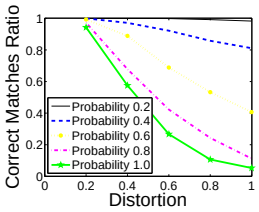
Robustness. Testing the two measures for robustness against noise, `Artemis` was a clear winner for both types of noise insertion. The DTW vector-based approach displayed a significant decline in performance with the increase in the value of the probability and distortion level parameters. `Artemis` was able to maintain a very high rate of finding the noisy queries' originating counterparts. For almost all the datasets and experiments its retrieval accuracy was over 98%. These observations are shown in Figure 5, for the case of *shifts*, and in Figure 6, for the case of *swaps*. `Artemis` displayed a decline in performance only for *Pioneer* and the case of swaps. Note that in Figure 5 we do not show the performance of `Artemis` for *Pioneer* with respect to retrieval accuracy vs. the noise parameters since it always gave a retrieval accuracy of 100%.

Classification. In Table 2 we see the results of the k-NN classification experiment. For the *Hepatitis* and *ASL*, the DTW vector-based approach performs marginally better, while for *Pioneer* `Artemis` is superior. Figure 7 depicts the results of the additional classification experiment conducted on *ASL*, in which the k nearest neighbors of each e-sequence are scanned to determine if they correspond to the same phrase in English. Figure 7a shows the ratio (of the e-sequences which do have equivalents) for which at least one identical phrase can be found within the k nearest neighbors. Figure 7b shows the ratio of the total sum of existing identical phrases that have been found. For both cases, `Artemis` gives a better grouping of semantically related e-sequences. Furthermore, empirical results showed that the groupings provided by `Artemis` are more meaningful for humans. Thus, `Artemis` proves more suitable for applications related to sign language.

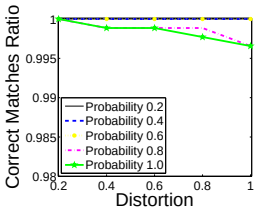
Lower bounding. Table 3 summarizes the assessment of $Artemis_{LB}$. The second column shows the tightness of the lower bound while the third shows the pruning power. The higher values are observed on *ASL*, contrary to *Pioneer* which yields the lowest scores. The latter consists of unequally represented



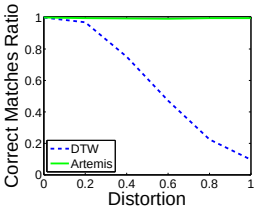
(a) *ASL*: retrieval accuracy for $p = 1.0$



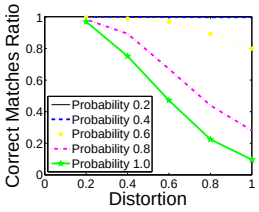
(b) *ASL*: DTW retrieval accuracy



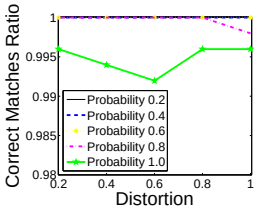
(c) *ASL*: `Artemis` retrieval accuracy



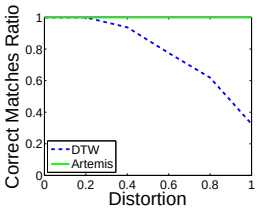
(d) *Hepatitis*: retrieval accuracy for $p = 1.0$



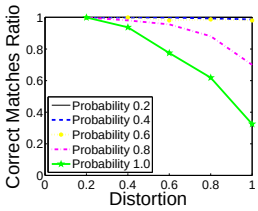
(e) *Hepatitis*: DTW retrieval accuracy



(f) *Hepatitis*: `Artemis` retrieval accuracy



(g) *Pioneer*: retrieval accuracy: $p = 1.0$



(h) *Pioneer*:DTW Retrieval Accuracy

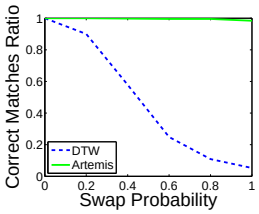
Fig. 5: Robustness experiment when noise is added in the form of shifts.

Dataset	`Artemis` 1-NN	`Artemis` 3-NN	DTW 1-NN	DTW 3-NN
HepData	0.7209	0.7811	0.7403	0.8072
Pioneer	0.9750	0.9750	0.9375	0.9375
ASL	0.4307	0.4013	0.4358	0.4188

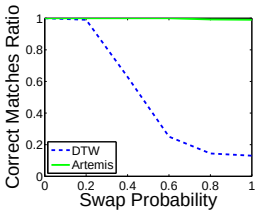
Table 2: k-NN classification results, for $k = 1, 3$.

classes; one of the three classes contains 102 out of 160 samples in total. Thus, its e-sequences are highly homogeneous and thereby render the lower bound technique unable to prune many of them.

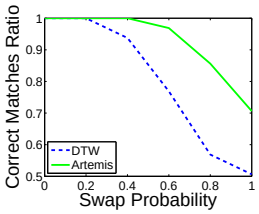
Scalability. The results of the scalability experiment can be seen in Figure 8. For *Pioneer* we observe that `Artemis` is slower than the DTW vector-based approach, as the complexity analysis of Section 3 would suggest. The runtime of `Artemis` blows up in the case of *Hepatitis*. For e-sequences of large size, `Artemis`'



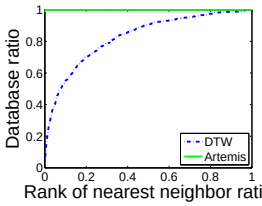
(a) *ASL*: retrieval accuracy for $p = 1.0$



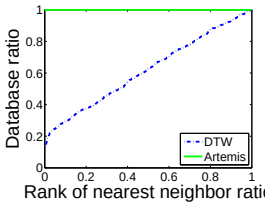
(b) *Hepatitis*: retrieval accuracy for $p = 1.0$



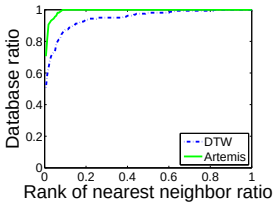
(c) *Pioneer*: retrieval accuracy for $p = 1.0$



(d) *ASL*: ranks of NN for swap probability 1.0

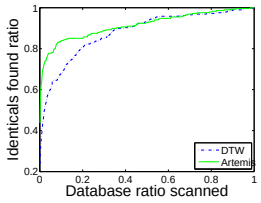


(e) *Hepatitis*: ranks of NN for swap probability 1.0

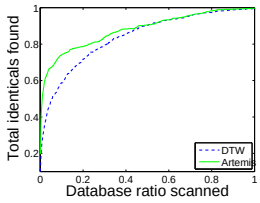


(f) *Pioneer*: ranks of NN for swap probability 1.0

Fig. 6: Robustness experiment when noise is added in the form of swaps.



(a) Identical phrases found



(b) Total sum of identical phrases found

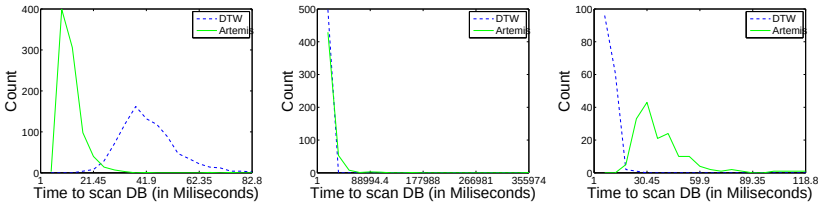
Fig. 7: Identical phrase experiment: *ASL*.

Dataset	Tightness	1-NN pruning power
ASL	0.8837	0.7931
Hepatitis	0.7166	0.7012
Pioneer	0.6189	0.4855

Table 3: Tightness and pruning power of $Artemis_{LB}$ for the 1-NN classification experiment and for the three datasets.

$O(n^4)$ complexity (without the use of hash-tables) significantly affects its performance. On the other hand, we observe that for *ASL* `Artemis` is faster than DTW. The reason for that is the large alphabet size, $|\sigma|$. A way to overcome this would be to prune the alphabet during the comparisons and keep only the union of the labels present in the compared e-sequences $\mathcal{S}, \mathcal{T}$. The alphabet

size would then be at most $(|\mathcal{V}_S| + |\mathcal{V}_T|)$, and the new complexity of DTW $O(|\mathcal{V}_S||\mathcal{V}_T| \cdot (|\mathcal{V}_S| + |\mathcal{V}_T|))$.



(a) *ASL* (b) *Hepatitis* (c) *Pioneer*

Fig. 8: Histograms of the time (in milliseconds) required to compare each element against the whole dataset.

4.3 Lessons Learned

There was no clear winner, between the two proposed methods, in our k-NN classification experiments. The insight we acquired suggests that the choice must be application dependant. The additional experiment on *ASL* supports the thesis that, if semantic information resides in the relations between intervals, `Artemis` has an advantage over the DTW approach. This is further supported by the results of the noise-robustness experiments. In particular, the DTW vector-based approach displayed a deterioration in performance in accordance with the increase of noise. `Artemis` proved highly robust against the types of noise that we experimented with. The advantage of `Artemis` is that a correspondence amongst event-intervals is determined based on the relations of the event-intervals within the e-sequences; contrary to the DTW approach which examines the e-sequences point-by-point and out of their context. The correspondence among event-intervals allows `Artemis` to easily identify the originating counterpart of noisy e-sequences; e-sequences with the same count of each event label yield in the majority of cases lower distance scores than others with different event labels and size.

The scalability experiments showed that DTW outperforms `Artemis`, which suffers from a computational blow up for large e-sequences. On the other hand, DTW becomes slower when the alphabet size is significantly larger than the size of the e-sequences. To speed up search using `Artemis`, our lower bound technique proved significantly tight; the average tightness was measured at 61.8% for *Pioneer*, and up to 88% for *ASL*. This translates to a pruning power of 48.5% to 79.3% over the brute force sequential scan of the database.

5 Related Work

Existing work on interval-based sequences has so far been focusing merely on frequent pattern and association rule mining.

Several approaches [17, 30] consider discovering frequent intervals in databases, where intervals appear sequentially and are not labelled, while others [7] consider temporally annotated sequential patterns where transitions from one event to another have a time duration. A graph-based approach [10] represents each temporal pattern by a graph considering only two types of relations between events (*follow* and *overlap*). An approach for mining sequences of interval-based events in a database is discussed in Fu et al. [11], however it is limited to certain forms of patterns.

A generalized interval-based framework [15] improves support counting techniques for mining interval-based episodes; nonetheless, no temporal relations are considered between events. Apriori-based techniques [8, 9, 19, 6, 1] for finding temporal patterns and association rules on interval-based event sequences have been proposed, some [9] also applying interestingness measures to evaluate the significance of the findings.

BFS-based and DFS-based approaches [25–27, 32] apply efficient pruning techniques, thus reducing the inherent exponential complexity of the mining problem, while a non-ambiguous event-interval representation is defined [33] that considers start and end points of e-sequences and converts them to a sequential representation. Finally, there has been some recent work on mining semi-partial orders of time intervals [22].

Moreover, in Ale et. al [2], the lifetime of an item is defined as the time between the first and the last occurrence and the temporal support is calculated with respect to this interval. Finally, Lu et. al [18] study inter-transaction association rules by merging all itemsets within a sliding time window inside a transaction.

Recent work on *margin-closed* patterns [21, 22], focuses on significantly reducing the number of reported patterns by favouring longer patterns and suppressing shorter patterns with similar frequencies. The extracted set of margin-closed patterns may include a significantly smaller set of patterns compared to the set of closed patterns while retaining the most important information about the data. A unifying view of temporal concepts and data models has been formulated [20] to enable categorization of existing approaches for unsupervised pattern mining from symbolic temporal data; Time point-based methods and interval-based methods as well as univariate and multivariate methods are considered.

6 Summary and Conclusions

We have defined the problem of comparing sequences of interval-based events and presented two polynomial-time algorithms. The first reduces the problem to matching vectors, while the second, `Artemis`, is based on determining correspondence among intervals and maps the problem to the assignment problem. The two measures were tested for their robustness against two types of artificial noise, their k-NN classification power, and, finally, their scalability. `Artemis` demonstrates a remarkable robustness against noise, while there exist cases for both measures in which they perform faster than the other. Additionally, we

developed a linear-time lower-bounding technique for `Artemis`, which we tested in terms of tightness and pruning power.

Directions for future work include studying the applicability of the proposed methods to other application domains and real-world scenarios. Furthermore, we plan on investigating alternative problems to assess the similarity of event-interval sequences, such as the Maximum Common Subsequence and the Maximum Contiguous Subsequence. Devising algorithms for these problems would allow the direct application of event-interval sequences to domains such as anomaly detection, profiling of executable files, network monitoring, and many others.

Acknowledgements: This work was supported in part by the Academy of Finland ALGODAN Centre of Excellence. We would like to thank D. Patel for providing us with the *Hepatitis* dataset, and F. Mörchen for the *Pioneer* dataset.

References

1. T. Abraham and J. F. Roddick. Incremental meta-mining from large temporal data sets. In *Proceedings of the Workshops on Data Warehousing and Data Mining*, pages 41–54, 1999.
2. J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *Proc. of the SAC*, pages 294–300, 2000.
3. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.
4. B. Berendt. Explaining preferred mental models in Allen inferences with a metrical model of imagery. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pages 489–494, 1996.
5. B. Bergen and N. Chang. Embodied construction grammar in simulation-based language understanding. *Construction grammars: Cognitive grounding and theoretical extensions*, pages 147–190, 2005.
6. X. Chen and I. Petrounias. Mining temporal features in association rules. In *Proc. of PKDD*, pages 295–300, London, UK, 1999. Springer-Verlag.
7. F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of temporally annotated sequences. In *SDM*, volume 6, pages 346–357, 2006.
8. F. Höppner. Discovery of temporal patterns - learning rules about the qualitative behaviour of time series. In *Proc. of PKDD*, pages 192–203, 2001.
9. F. Höppner and F. Klawonn. Finding informative rules in interval sequences. In *Proc. of IDA*, pages 123–132, 2001.
10. S.-Y. Hwang, C.-P. Wei, and W.-S. Yang. Discovery of temporal patterns from process instances. *Computers in Industry*, 53(3):345–364, 2004.
11. P. Kam and A. W. Fu. Discovering temporal patterns for interval-based events. In *DaWaK*, pages 317–326, 2000.
12. R. Kosara and S. Miksch. Visualizing complex notions of time. *Studies in Health Technology and Informatics*, pages 211–215, 2001.
13. O. Kostakis, P. Papapetrou, and J. Hollmén. Distance measure for querying arrangements of temporal intervals. In *Proc. of ACM Pervasive Technologies Related to Assistive Environments (PETRA)*, 2011.
14. J. B. Kruskal and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In *Time Warps*. Addison-Wesley, 1983.
15. S. Laxman, P. Sastry, and K. Unnikrishnan. Discovering frequent generalized episodes when events persist for different durations. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1188–1201, 2007.

16. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics*, 10(8):707–710, 1966.
17. J.-L. Lin. Mining maximal frequent intervals. In *Proc. of SAC*, pages 624–629, 2003.
18. H. Lu, J. Han, and L. Feng. Stock movement prediction and n-dimensional inter-transaction association rules. In *Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 12:1–12:7, 1998.
19. C. Mooney and J. F. Roddick. Mining relationships between interacting episodes. In *Proc. of SDM*, 2004.
20. F. Mörchen. Unsupervised pattern mining from symbolic temporal data. *SIGKDD Explor. Newsl.*, 9:41–55, June 2007.
21. F. Mörchen. Temporal pattern mining in symbolic time point and time interval data. In *Proc. of ACM SIGKDD*, 2010.
22. F. Mörchen and D. Fradkin. Robust mining of time intervals with semi-interval partial order patterns. In *SDM*, pages 315–326, 2010.
23. J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.
24. F. Pachet, G. Ramalho, and J. Carrive. Representing temporal musical objects and reasoning in the MusES system. *Journal of new music research*, 25(3):252–275, 1996.
25. P. Papapetrou, G. Benson, and G. Kollios. Discovering frequent poly-regions in dna sequences. In *Proc. of the IEEE ICDM Workshop on Data Mining in Bioinformatics*, pages 94–98, 2006.
26. P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Discovering frequent arrangements of temporal intervals. In *Proc. of IEEE ICDM*, pages 354–361, 2005.
27. P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems (KAIS)*, 21:133–171, 2009.
28. D. Patel, W. Hsu, and M. Lee. Mining relationships among interval-based events for classification. In *Proc. of ACM SIGMOD*, pages 393–404, 2008.
29. N. Pissinou, I. Radev, and K. Makki. Spatio-temporal modeling in video and multimedia geographic information systems. *GeoInformatica*, 5(4):375–409, 2001.
30. R. Villafane, K. A. Hua, D. Tran, and B. Maulik. Knowledge discovery from series of interval events. *Intelligent Information Systems*, 15(1):71–89, 2000.
31. M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multidimensional time-series. *The VLDB Journal*, 15:1–20, January 2006.
32. E. Winarko and J. F. Roddick. Armada - an algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, 63(1):76–90, 2007.
33. S.-Y. Wu and Y.-L. Chen. Mining nonambiguous temporal patterns for interval-based events. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):742–758, 2007.