IBSM: Interval-Based Sequence Matching

Alexios Kotsifakos* Panagiotis Papapetrou † Vassilis Athitsos *

Abstract

Sequences of event intervals appear in several application domains including sign language, sensor networks, medicine, human motion databases, and linguistics. Such sequences comprise events that occur at time intervals and are time stamped at their start and end time. In this paper, we propose a new method, called `IBSM`, for comparing such sequences. `IBSM` performs *full sequence matching* using a vector-based representation of the original sequence. At each time point an *event vector* is computed; hence, the original sequence is mapped to an ordered set of vectors, which we call *event table*. Given two sequences, their event tables are *resized* using bilinear interpolation, which ensures they are of the same size. The resulting event tables are then compared using the Euclidean distance. In addition, we propose two techniques for reducing the computational cost of `IBSM` when performing nearest neighbor search in a large database. Extensive experiments on eight real datasets show that `IBSM` outperforms existing state-of-the-art methods in terms of nearest neighbor classification accuracy, and by up to two orders of magnitude in terms of runtime.

1 Introduction

Many application domains are characterized by sequences of event intervals, such as sign language [26,27], medicine [14], geo-informatics [30], cognitive science [5], linguistics [6], and music informatics [24]. For example, in sign language, sentences are constructed by events corresponding to occurrences of various grammatical, syntactic, and gestural expressions. Such expressions have a time duration and they may occur concurrently, hence, sequences of event intervals are formed. Moreover, in medicine [14], patients typically undergo a series of diagnostic tests and treatements that have a time duration and may also occur concurrently.

The main advantage of event interval sequences over traditional event sequences, which model series of instantaneous events, is that they incorporate the notion of duration in their event representation. Essentially, sequences of event intervals can be encoded as a collection of labeled events accompanied by their start and end time values. An example of a sequence of five labeled temporal intervals is shown in Figure 1.

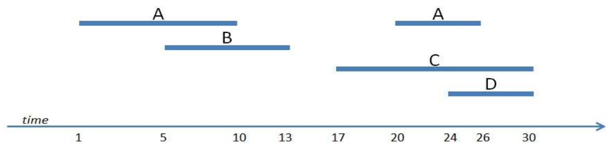


Figure 1: Example of a sequence of five event intervals. Four event labels are used in this example: A, B, C, and D. Note that event A occurs twice.

Knowledge discovery tasks have been so far the main focus of many studies on sequences of event intervals. Such tasks include mining patterns and association rules [2, 13, 21, 27], mining semi-interval partial orders [23], and discovering relationships for classification [29]. Surprisingly, similarity searching and matching has received limited attention [15,16].

Here, we study the problem of *full sequence matching* of sequences of event intervals. Developing robust methods for solving this problem will facilitate a wide range of knowledge discovery tasks (such as classification and clustering) in a wide range of application domains where such sequences occur, like the ones mentioned above. Existing similarity measures on discrete sequences, such as the Levenshtein distance [18], are not directly applicable to sequences of event intervals. As shown by Kostakis et al. [15], mapping a sequence of event intervals to a discrete event sequence, by considering only the start and end points of each event interval and labeling them with the event label that corresponds to that interval, may lead to a large number of false matches.

A recent similarity measure, `Artemis` [15], has been proposed for similarity matching of event interval sequences. Each sequence is represented by the set of temporal relations between all pairs of event intervals. Given two sequences, and using this representation, `Artemis` finds an optimum matching between their pairs of event intervals by mapping the sequences into a

*Department of Computer Science and Engineering, University of Texas at Arlington, USA.

†Department of Computer Science and Information Systems, Birkbeck University of London, UK, and Department of Information and Computer Science, Aalto University, Finland

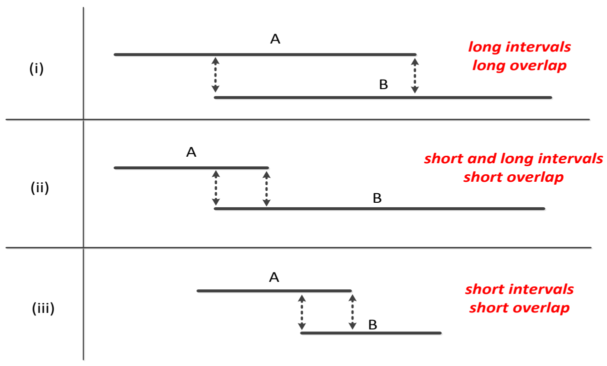


Figure 2: Three sequences of two event intervals A and B. In all sequences the temporal relation between the events is the same (they are overlapping), though the time duration of both the intervals and their overlap is different.

bipartite graph. Then, similarity is inferred using the Hungarian algorithm. One limitation of this method is that it only considers the types of temporal relations in the matching and ignores the actual duration of the events. For instance, consider the three sequences shown in Figure 2. Each sequence consists of two event intervals with the same label and the same temporal relation: *overlap*. Hence, `Artemis` would conclude that the three sequences are identical. Nonetheless, one could argue that these sequences differ substantially: the time durations of the intervals vary among the three sequences and the time duration of their overlap is also different.

In this paper, we address this shortcoming of `Artemis` and propose a novel method, `IBSM` (shorthand for Interval-Based Subsequence Matching), which performs full sequence matching by: (1) transforming the compared event-interval sequences to a vector-based representation, and (2) computing the Euclidean distance of the new representations of the sequences. The key novelty of `IBSM` is that it explicitly takes into account the time durations of the event intervals in the sequences and implicitly their temporal relations.

The main contributions of this paper include:

- a robust vector-based representation of event interval sequences that facilitates full sequence matching,
- a novel method, called `IBSM`, for matching event-interval sequences that exploits the vector-based representation by applying bilinear interpolation and computes the Euclidean distance of the resulting sequence representations,
- two techniques based on sampling and alphabet reduction that speed up the runtime and decrease the memory requirements of `IBSM` when performing nearest neighbor search in a database of event-interval sequences, and
- an extensive experimental evaluation of `IBSM` against two state-of-the art measures on eight real datasets taken from different application domains, where `IBSM` is shown to outperform existing state-of-the-art methods in 7 out of 8 datasets, in terms of nearest neighbor classification accuracy, and by up to two orders of magnitude in terms of runtime.

The remainder of this paper is organized as follows: in Section 2 we provide the necessary background and core definitions, whereas in Section 3 we describe the proposed method. In Section 4, we present two techniques for speeding up nearest neighbor search using `IBSM`. Our experimental evaluation and findings are discussed in Section 5, while Section 6 presents the related work. Finally, Section 7 concludes the paper and discusses directions for future work.

2 Background

Let $\Sigma = \{E_1, \ldots, E_m\}$ be an alphabet of m event labels. An event that occurs over a time interval defines an *event interval* and an ordered multiset of event intervals defines an *event-interval sequence*. Next, we provide a more formal defintion for these two concepts.

Definition 1. **(event interval)** *An event interval is defined as a triple $S = (E, t_{start}, t_{end})$, where $S.E \in \Sigma$ and $S.t_{start}$, $S.t_{end}$ correspond to the start and end time of S, respectively. In general, $S.t_{start} \leq S.t_{end}$, where the equality holds when the event is instantaneous.*

Definition 2. **(e-sequence)** *An event-interval sequence or e-sequence $\mathcal{S}=\{S_1, \ldots, S_n\}$ is an ordered multiset of n event intervals. The temporal order of the event intervals in $\mathcal{S}$ is ascending based on their start time and in the case of ties it is descending based on their end time. In case of further ties, alphabetical order is used.*

The *size* of an e-sequence $\mathcal{S}$, $|\mathcal{S}|$, is the number of event intervals in the e-sequence, whereas the *length* of $\mathcal{S}$ corresponds to the maximum time point in $\mathcal{S}$, i.e., $length(\mathcal{S}) = S_n.t_{end}$. Recalling the example in Figure 1 and following the above definitions, we derive the following e-sequence representation: $\mathcal{S} = \{(A, 1, 10), (B, 5, 13), (C, 17, 30), (A, 20, 26), (D, 24, 30)\}$.

We use Allen's model for temporal logic [3, 4] to define the relations between two event intervals. Given two event intervals A and B, we consider seven temporal relations: *meet*, *match*, *overlap*, *left-contain*, *right-contain*, *contain*, and *follow*. These relations are shown in Figure 3.

The problem we study in this paper is how to assess the similarity between two e-sequences $\mathcal{S}$ and $\mathcal{T}$. A robust distance measure should take into account several common characteristics of $\mathcal{S}$ and $\mathcal{T}$: (1) common event labels, (2) pairs of event intervals with the same temporal relation, (3) the time duration of each event interval as well as the duration of each temporal relation.

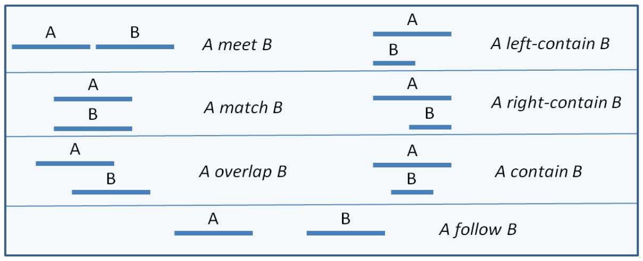


Figure 3: Seven temporal relations between two event-intervals.

3 IBSM: Interval-Based Sequence Matching

We introduce a novel approach, called IBSM (Interval-Based Sequence Matching), for performing e-sequence matching between two e-sequences. Each e-sequence $\mathcal{S}$ is mapped to its corresponding *resized event table* representation, which is computed in two phases: (a) $\mathcal{S}$ is converted to an *event table* $A_{\mathcal{S}}$, and then (b) $A_{\mathcal{S}}$ is resized to yield the final representation of $\mathcal{S}$. Finally, IBSM computes the Euclidean distance of these representations. As shown in Section 3.3, IBSM performs e-sequence matching in time linear to the length of the e-sequences.

3.1 Event table representation Firstly, we assume that at each point in time all events may appear simultaneously. More than that, even multiple instances of one event are allowed to happen concurrently at a specific time point. Hence, given an e-sequence $\mathcal{S}$ and a time point t, some events in Σ may be *active* and some may be not. We use an *event vector* to record this information.

Definition 3. **(active event interval)** *An event interval* $S = (E, t_{start}, t_{end})$ *is active at time point* t *if and only if*

$$S.t_{start} \leq t \leq S.t_{end}$$

Definition 4. **(event vector)** *Given an e-sequence* $\mathcal{S} = \{S_1, \ldots, S_n\}$ *and a time point* t*, the event vector* $\mathcal{V}^t_{\mathcal{S}}$ *of* $\mathcal{S}$ *at time* t *is a vector of size* $|\Sigma|$*, where each value* $\mathcal{V}^t_{\mathcal{S}}(i)$ *is a non-negative integer that records how many event intervals with label* $E_i \in \Sigma$ *are active at time* t*.*

Therefore, for each time point $t \in [1, \ldots, length(\mathcal{S})]$, the corresponding event vector $\mathcal{V}^t_{\mathcal{S}}$ is computed and recorded as a column in the *event table* $A_{\mathcal{S}}$.

Definition 5. **(event table)** *Given an e-sequence* $\mathcal{S} = \{S_1, \ldots, S_n\}$ *its corresponding event table* $A_{\mathcal{S}}$ *is a matrix of size* $|\Sigma| \times length(\mathcal{S})$*, where* $A_{\mathcal{S}}(i,j) = \mathcal{V}^j_{\mathcal{S}}(i)$*, for* $i \in [1, \ldots \Sigma]$ *and* $j \in [1, \ldots length(\mathcal{S})]$*.*

In other words, each e-sequence $\mathcal{S}$ is represented by event table $A_{\mathcal{S}}$, where each cell (i,j) of the table reflects the number of times the i-th event in Σ appears at the j-th time point in $\mathcal{S}$.

3.2 Resizing the event table The next step of our method is to ensure that the tables of the two e-sequences $\mathcal{S}$ and $\mathcal{T}$ under comparison are of the same size. To achieve this, we use a rezising parameter γ. Typically, γ is set to the maximum time point between the two e-sequences, that is

$$\gamma = max\{length(\mathcal{S}), length(\mathcal{T})\}.$$

Note that, in case we are computing pair-wise distances of more than two e-sequences, γ is set to the maximum e-sequence length.

Definition 6. **(γ-resized event table)** *Given an event table* $A_{\mathcal{S}}$ *and a resizing parameter* γ*, the* γ-rezised event table $A^{\gamma}_{\mathcal{S}}$ *is computed by resizing* $A_{\mathcal{S}}$ *to* γ *using* bilinear interpolation.

Consequently, after this step is executed, $\mathcal{S}$ and $\mathcal{T}$ are represented by their γ-resized event tables $A^{\gamma}_{\mathcal{S}}$ and $A^{\gamma}_{\mathcal{T}}$, respectively, which are of the same size $|\Sigma|$ x γ.

The final step of IBSM is to compute the distance of the resulting γ-resized event tables of $\mathcal{S}$ and $\mathcal{T}$. The distance measure used is the Euclidean distance, and it is computed as follows:

$$(3.1) \qquad D(\mathcal{S},\mathcal{T}) = \sqrt{\sum_{i=1}^{|\Sigma|} \sum_{j=1}^{\gamma} (A^{\gamma}_{\mathcal{S}}(i,j) - A^{\gamma}_{\mathcal{T}}(i,j))^2}$$

3.3 Complexity The online computational complexity of IBSM is linear to the length of the e-sequences. This is a significant improvement over the existing state-of-the-art *Artemis* method [15], which is cubic to the e-sequence length. More specifically, given two e-sequences $\mathcal{S}$, $\mathcal{T}$, defined over an event alphabet Σ, and a resizing parameter γ, the online computational time and space complexity of IBSM is $O(|\Sigma| \times \gamma)$. Regarding the pre-processing step, the method requires $O(|\Sigma| \times \mathit{length}(\mathcal{S}))$ and $O(|\Sigma| \times \mathit{length}(\mathcal{T}))$ time and space for computing and resizing the event tables for $\mathcal{S}$ and $\mathcal{T}$, respectively.

4 Nearest neighbor search using IBSM

IBSM can be used for nearest neighbor search in an *e-sequence database*, i.e., a collection of e-sequences. Let DB be an e-sequence database. Given a query e-sequence Q we want to find the nearest neighbor of Q in DB. Despite the linear complexity of IBSM, which already achieves over an order of magnitude speedup compared to the state-of-the-art Artemis method, it depends on the lengths of the e-sequences in DB as well as the alphabet size $|\Sigma|$. Thus, we next introduce two techniques for speeding up IBSM for nearest neighbor search, that can achieve significant speedups by reducing the e-sequence lengths as well as the alphabet size.

4.1 Speedup by sampling The first speedup technique that can be applied is to reduce the number of columns of the γ-resized event tables of the e-sequences in DB. Given the γ-resized event table $A^{\gamma}_{\mathcal{S}}$ of each e-sequence $\mathcal{S} \in DB$, we perform uniform sampling on the columns of $A^{\gamma}_{\mathcal{S}}$ with a sampling period equal to δ. This results in including only columns $1, 1+\delta, 1+2\delta, 1+3\delta, \ldots$ from $A^{\gamma}_{\mathcal{S}}$.

More formally, we consider the following set of columns:

$$\{j+1 | j \in [0, \gamma) \text{ and } \mathit{mod}(j, \delta) = 0\}.$$

Sampling results in a reduction on the columns of $A^{\gamma}_{\mathcal{S}}$ expressed by the *sampling rate* $r \in [0,1]$ given by

$$r = \frac{\lceil \frac{\gamma}{\delta} \rceil}{\gamma}. \tag{4.2}$$

Effectively, the number of columns in $A^{\gamma}_{\mathcal{S}}$ is reduced from γ to $\lceil \frac{\gamma}{\delta} \rceil$, which implies that each γ-resized event table is reduced to $r \times 100\%$ of its original size.

4.2 Speedup by alphabet reduction We pressent a second speedup technique that reduces the number of rows of the γ-resized event tables of the e-sequences in DB. The key idea is to reduce the size of the alphabet Σ. Specifically, given a γ-resized event table $A^{\gamma}_{\mathcal{S}}$ of γ columns, for each event $E_i \in \Sigma$, the fraction of non-zero occurrences of E_i in row i of the table, denoted as $h(E_i, A^{\gamma}_{\mathcal{S}})$, is given by

$$h(E_i, A^{\gamma}_{\mathcal{S}}) = \frac{1}{\gamma} \sum_{j=1}^{\gamma} I(A^{\gamma}_{\mathcal{S}}(i,j)),$$

where $I(\cdot)$ is an indicator function such that

$$I(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$$

DEFINITION 7. **(event density)** *Given a database DB of e-sequences defined over an alphabet Σ, the event density $H(E_i, DB)$ of each $E_i \in \Sigma$ in DB is defined as follows:*

$$H(E_i, DB) = \frac{1}{|DB|} \sum_{\mathcal{S} \in DB} h(E_i, A^{\gamma}_{\mathcal{S}}).$$

In other words, the event density of E_i in DB expresses the average fraction of non-zero occurrences of E_i in the γ-resized event tables of the e-sequences in DB.

The alphabet reduction technique we propose here computes the density of each event $E_i \in \Sigma$ in DB and removes E_i from Σ, if

$$H(E_i, DB) < \epsilon \text{ , where } \epsilon \in [0,1].$$

Practically, ϵ is a threshold placed on the average frequency of event E_i in DB. If E_i appears in less than $\epsilon \times 100\%$ of the columns of the γ-resized event tables in DB, on average, then it is removed from Σ. The intuition behind this technique is that event labels that appear more frequently in active event intervals in DB are those that mostly characterize the e-sequences in DB.

Therefore, the ratio of the reduced alphabet size over the initial size, called the *alphabet reduction rate* $s \in [0,1]$, is given by

$$s = 1 - \frac{|\{E_i \in \Sigma | H(E_i, DB) < \epsilon\}|}{|\Sigma|}. \tag{4.3}$$

Effectively, the number of rows in $A^{\gamma}_{\mathcal{S}}$ is reduced from $|\Sigma|$ to $s|\Sigma|$, which implies that each γ-resized event table is reduced to $s \times 100\%$ of its original size.

In the description of the two previous techniques we assumed that each technique is applied to the original γ-resized event table. To speed up IBSM we apply both techniques and finally compute the distance of the reduced event tables using Equation 3.1. As a result, the overall benefit gained by applying both techniques is a total reduction of each γ-resized event table to $r \times s \times 100\%$ of its original size.

5 Experiments

In this section we present the experimental setup and results evaluating the performance of IBSM.

5.1 Experimental Setup In our experiments we used eight real datasets, and compared `IBSM` with two competitor methods in terms of 1-NN classification accuracy and runtime.

5.1.1 Datasets Our eight real datasets were taken from various application domains. A summary of the statistics for each dataset is shown in Table 1. Below, we describe each dataset in more detail:

- **ASL-BU** [27]. Event labels correspond to grammatical or syntactic forms (e.g., wh-word, wh-question, verb, noun, etc.) as well as facial or gestural expressions (e.g., head tilt right, rapid head shake, eyebrow raise, etc.). An e-sequence is an expression of a sentence using sign language.

- **ASL-BU2**. This is the newest version of ASL-BU. The structure is the same as that of ASL-BU but this dataset contains a large number of new e-sequences and versions of the previous ones that are improved in terms of annotation, where additional labels have been introduced.

- **Auslan** [23]. The e-sequences were derived from the Australian Sign Language dataset available in the UCI repository [1]. Each event interval represents a word like *girl* or *right.*

- **Blocks** [23]. Event labels correspond to visual primitives obtained from videos of a human hand stacking colored blocks and describe which blocks are touched as well as the actions of the hand (e.g., contacts blue or red, attached hand red, etc.). Each e-sequence represents one of eight different scenarios including atomic actions, such as *pickup*, or complete scenarios, such as *assemble.*

- **Context** [23]. Event labels were derived from categoric and numeric data describing the context of a mobile device carried by humans in different situations. Each e-sequence represents one of five different scenarios such as *street* or *meeting.*

- **Hepatitis** [28] The dataset contains information about patients who have either Hepatitis B or Hepatitis C. The event intervals represent the results of 63 regular tests. Each e-sequence describes a series of tests taken by a patient.

- **Pioneer** [23]. This dataset was constructed from the Pioneer-1 dataset available in the UCI repository [2]. Event intervals correspond to the input provided by the robot sensors. Each e-sequence in the dataset describes one of three scenarios: *gripper*, *move*, *turn.*

- **Skating** [23]. Event intervals describe muscle activity and leg position of 6 professional In-Line Speed Skaters during controlled tests at 7 different speeds on a treadmill. Each e-sequence represents a complete movement cycle.

5.1.2 Methods We compared these methods:

- `IBSM`: we experimented with and without sampling and alphabet reduction.

- `Artemis` [15]: the state-of-the-art method for similarity mathing of e-sequences.

- `DTW-based` [15]: the baseline approach which `Artemis` has been compared to. Each e-sequence is mapped to a set of vectors. Each vector keeps track of the number of times each event label appears, and is created only for these time points where a change occurs, i.e., when one or more event intervals become active or inactive. The dimensions are combined using the root mean square.

5.1.3 Evaluation We computed the 1-NN classification accuracy for each method and dataset. For each dataset, each e-sequence was considered to be a query Q and the distance from the remaining e-sequences in the dataset was computed. The class label of the 1-NN e-sequence was compared to that of Q. In case of ties, a majority scheme was followed.

To be more specific, let DB be the set of e-sequences in a dataset. For each query $Q \in DB$, we computed its distance $D(Q, \mathcal{S})$ against each e-sequence $\mathcal{S} \in DB \backslash Q$. Note that $D(\cdot)$ corresponds to the distance function used by each of the three methods. Let $\mathcal{D}_Q = \{D(Q, \mathcal{S}) | \forall \mathcal{S} \in DB \setminus Q\}$ be the set of distances of Q to $DB \setminus Q$ and $NN_Q = min(\mathcal{D}_Q)$ the distance of the 1-NN of Q in $DB \backslash Q$. If there is only one e-sequence in $DB \backslash Q$ with distance NN_Q from Q then we just compare the class labels of the two e-sequences. If there is more than one e-sequence with that distance, then we consider the union of all classes of these e-sequences and report the majority class as the class of the nearest neighbor.

In the results reported in Section 5.2, the classification accuracy for each dataset is defined as the percentage of the total number of e-sequences of the dataset that were correctly classified.

The methods have been implemented in Matlab on an AMD Opteron 8220 SE processor running at 2.8GHz.

[1]http://www.ics.uci.edu/ mlearn/MLRepository.html
[2]http://archive.ics.uci.edu/ml/

Table 1: Dataset Statistics

Dataset	# of	e-sequence size			# of	# of	max e-seq.	interval size			
	e-seq.	min.	max.	average	labels	classes	length	mean	stdev	min	max
ASL-BU	873	3	40	17	216	9	5901	594	590	3	4468
ASL-BU2	1839	4	93	23	254	7	14968	669	808	3	9967
Auslan2	200	9	20	12	12	10	30	20	12	1	30
Blocks	210	3	12	6	8	8	123	17	12	1	57
Context	240	47	149	81	54	5	284	69	81	1	284
Hepatitis	498	15	592	108	63	2	7555	634	1093	1	7555
Pioneer	160	36	89	56	92	3	80	36	21	1	80
Skating	530	27	143	44	41	6	6829	576	672	1	6829

5.2 Experimental Results Next, we present our experimental findings in terms of classification accuracy and runtime.

5.2.1 Classification accuracy We first evaluated the performance of `IBSM` in terms of 1-NN classification accuracy on the eight datasets and compared it to that of `Artemis` and `DTW-based`. For this erxperiment, we did not apply the speedup techniques for `IBSM`, i.e., we set the sampling period $\delta = 1$ and the alphabet reduction threshold $\epsilon = 0$. In Table 2 we can see that `IBSM` is a clear winner in all datasets except for the Pioneer dataset. The results strongly suggest that when comparing e-sequences it is essential to take into account the interval durations along with the relation types (which is what `IBSM` does), since the 1-NN classification accuracy can be substantially improved.

Next, we studied the effect of sampling and alphabet reduction on the accuracy of `IBSM` when performing nearest neighbor search. We first experimented on different sampling rates r keeping the alphabet size fixed to $|\Sigma|$. In Figure 4, we show the performance of `IBSM` in terms of 1-NN classification accuracy for different sampling rates, compared to `Artemis` and `DTW-based`. We observed that for all datasets a sampling rate of $r = 10\%$ suffices to maintain the original 1-NN classification accuracy (i.e., without sampling). Note that for ASL-BU, ASL-BU2, Hepatitis, and Skating the rate was $\approx 0.9\%$.

In addition, we experimented on different alphabet reduction rates while maintaing the sampling rate constant. Using the result of the previous experiment, we fixed the sampling rate to $r = 10\%$, and varied the alphabet reduction rate s. The results are shown in Figure 5. Observe that the effect of s on the 1-NN classification accuracy varies per dataset. Specifically, ASL-BU, Blocks, and Skating can tolerate a reduction down to $s = 40\%$ without significant drop in the accuracy, while for Hepatitis and Pioneer the original accuracy is mantaintaned until s drops down to 30%. The results are quite worse for Context, where $s = 50\%$, and Auslan2, where $s = 70\%$. Finally, ASL-BU2 shows the best performance as it can tolerate an alphabet reduction down to $s = 10\%$ without significant loss in terms of 1-NN classification accuracy.

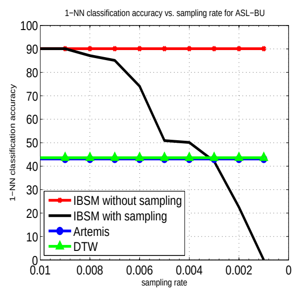
Table 2: 1-NN classification accuracy. For `IBSM` neither sampling nor aplhabet reduction have been applied.

Dataset	`IBSM`	`Artemis`	`DTW`
ASL-BU	**90.1**	79.56	43.58
ASL-BU2	**82.2**	80.53	77.25
Auslan2	**39.5**	28.5	22
Blocks	**100**	99	87
Context	**97.08**	90	89
Hepatitis	**77.91**	72.09	74.03
Pioneer	93.75	**97.5**	93
Skating	**97.74**	84	77

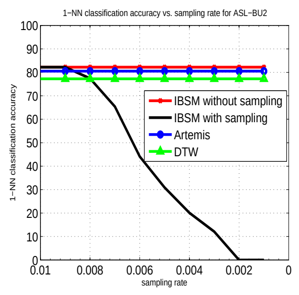
We conclude that in the majority of the datasets applying a sampling rate r of 10% and reducing the alphabet size by 50%, hence reducing the computational cost by a factor of 95%, can still maintain a 1-NN classification accuracy that is higher than that of the competitor methods for 7 out of 8 datasets.

Table 3: Runtime in seconds. We show the average total runtime (including pre-processing and matching) for comparing all pairs of e-sequences within a dataset. For `IBSM` neither sampling nor alphabet reduction have been applied.

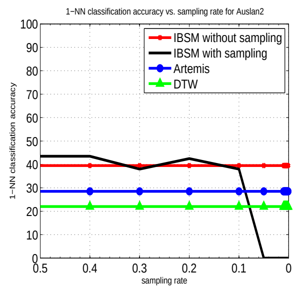
Dataset	`IBSM`	`Artemis`	`DTW`
ASL-BU	**81.34**	1915.97	190.89
ASL-BU2	**659.73**	14018.08	2140.47
Auslan2	**0.94**	9.64	2.73
Blocks	**0.79**	22.20	1.56
Context	**5.53**	121.66	9.56
Hepatitis	**27.96**	537.16	58.23
Pioneer	**8.76**	98.41	25.48
Skating	**14.96**	259.63	39.90



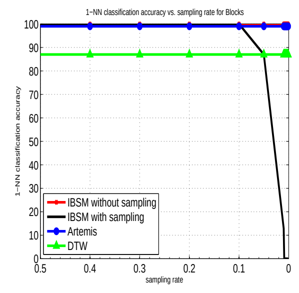
(a) *ASL-BU*



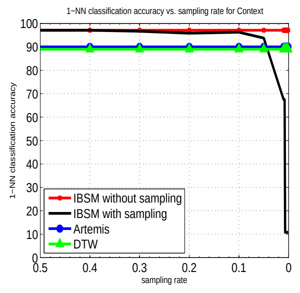
(b) *ASL-BU2*



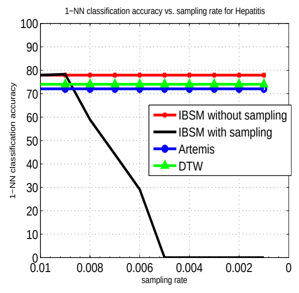
(c) *Auslan2*



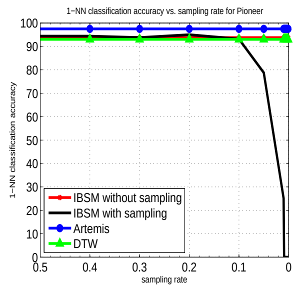
(d) *Blocks*



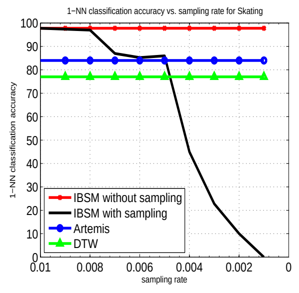
(e) *Context*



(f) *Hepatitis*

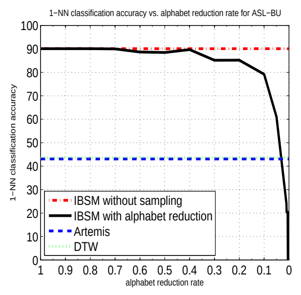


(g) *Pioneer*

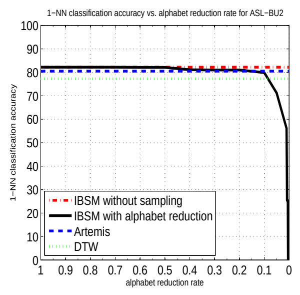


(h) *Skating*

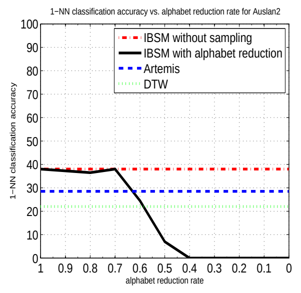
Figure 4: Comparison of `IBSM`, `Artemis`, and `DTW-based` for different sampling rates r. No alphabet reduction was applied ($\epsilon = 0$). The flat lines are used to indicate the 1-NN accuracy of the two competitor methods and `IBSM` without sampling.



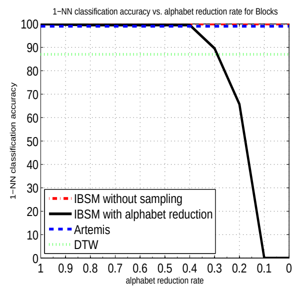
(a) *ASL-BU*



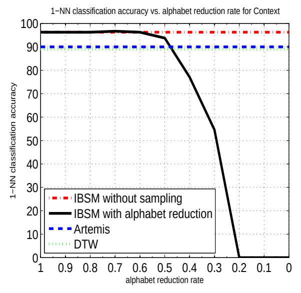
(b) *ASL-BU2*



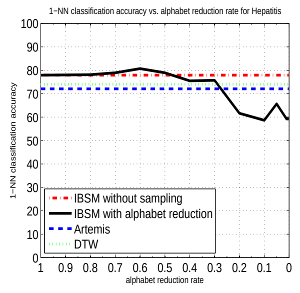
(c) *Auslan2*



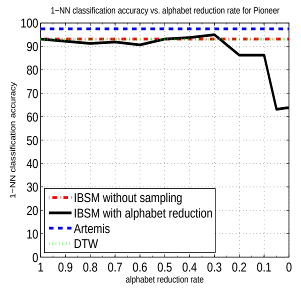
(d) *Blocks*



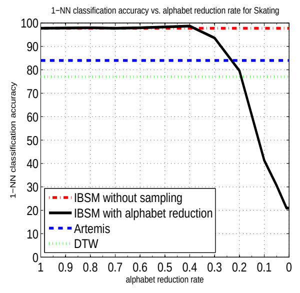
(e) *Context*



(f) *Hepatitis*



(g) *Pioneer*



(h) *Skating*

Figure 5: Comparison of `IBSM`, `Artemis`, and `DTW-based` for different alphabet reduction rates s. Note that the sampling rate for `IBSM` was fixed to $r = 10\%$. The flat lines are used to indicate the 1-NN accuracy of the two competitor methods and `IBSM` without sampling.

5.2.2 Runtime Finally, we benchmarked the methods in terms of runtime. The results are shown in Table 3. Note that for `IBSM` neither sampling nor alphabet reduction have been applied. For each dataset, we show the average total runtime (including pre-processing and matching) for comparing all pairs of e-sequences for each dataset. It turns out that `IBSM` is a clear winner in all cases achieving up to two orders of magnitude speedup against `Artemis`. This speedup becomes significantly higher (by at least another order of magnitude) when applying sampling and alphabet reduction. Due to space limitations we have not included these results in the paper.

6 Related Work

Existing work on temporal interval sequences has so far been focusing merely on frequent pattern and association rule mining. Several approaches [19, 31] consider discovering frequent intervals in databases, where intervals appear sequentially and are not labeled, while others [9] consider temporally annotated sequential patterns where transitions from one event to another have a time duration. A graph-based approach [12] represents each temporal pattern by considering only two types of relations between event-intervals (*follow* and *overlap*). In Ale et al. [2], the lifetime of an item is defined as the time between its first and last occurrence and the support is calculated with respect to this interval.

A large variety of Apriori-based techniques [1, 7, 10, 11, 13, 17, 20] for finding temporal patterns, episodes, and association rules on interval-based event sequences have been proposed. 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 hierarchical representation of interval-based event sequences has been proposed by Patel et al. [33] for frequent pattern mining. In addition, there has been some recent work on mining semi-partial orders of time intervals [23], while an efficient method for mining closed patterns of interval-based events has been proposed [8].

Recent work on *margin-closed* patterns [22, 23] focuses on significantly reducing the number of reported patterns by favoring longer patterns and suppressing shorter patterns with similar frequencies. A unifying view of temporal concepts and data models has been formulated in [21] to enable categorization of existing approaches to unsupervised pattern mining from symbolic temporal data; time point-based methods and interval-based methods as well as univariate and multivariate methods are considered.

A thorough survey of the literature on pattern mining from event-interval sequences is beyond the scope of this paper as the problem studied here is fundamentally different. To the best of our knowledge, the only existing principled method for comparing event-interval sequences is `Artemis` [15]. A baseline approach, `DTW-based`, which is described in Kostakis et al. [15], is based on an event-interval sequence representation that is vector-based, however, due to its contruction, it fails to take into consideration all pair-wise temporal relations [15]. Finally, a matrix-based representation [16] has been proposed for comparing event-interval sequences. Unfortunately this representation is ambiguous, i.e., two different event-interval sequences may have the same matrix representation, and hence it is not considered further in this paper.

7 Conclusions

We have proposed a novel method for full matching of sequences of interval-based events. The novelty of the method against existing approaches is the fact that it considers both temporal relations and duration of the event intervals in the e-sequences. The method converts the original sequences to an event table representation and then computes a Euclidean-based distance between the event tables. Additionally, we have proposed two techniques for speeding up `IBSM` when used for nearest neighbor search in large e-sequence databases. We provided an extensive experimental evaluation of `IBSM` against two state-of-the-art methods on eight real datasets. The performance of our method in terms of 1-NN classification accuracy and runtime is significantly better than the two competitors. Directions for future work include the exlporation of additional speedup techniques as well as the theoretical analysis of the properties of the proposed method.

Acknowledgments

This work was partially supported by grants from the National Science Foundation, IIS-0812601, IIS-1055062, CNS-1059235, and CNS-1035913, and in part by the Academy of Finland ALGODAN Centre of Excellence.

References

[1] T. Abraham and J. F. Roddick. Incremental meta-mining from large temporal data sets. In *ER '98: 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 *Proceedings of the 15th ACM Symposium On Applied Computing*, pages 294–300, 2000.

[3] J. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 1994.

[4] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[5] B. Berendt. Explaining preferred mental models in Allen inferences with a metrical model of imagery. In *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, pages 489–494, 1996.

[6] B. Bergen and N. Chang. Embodied construction grammar in simulation-based language understanding. *Construction grammars: Cognitive grounding and theoretical extensions*, pages 147–190, 2005.

[7] X. Chen and I. Petrounias. Mining temporal features in association rules. In *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 295–300. Springer-Verlag, 1999.

[8] Y.-C. Chen, W.-C. Peng, and S.-Y. Le. CEMiner-an effcient algorithms for mining closed patterns from interval-based data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2011.

[9] F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of temporally annotated sequences. In *Proceedings of the 6th SIAM Data Mining Conference*, volume 124, pages 348–359, 2006.

[10] F. Höppner. Discovery of temporal patterns - learning rules about the qualitative behaviour of time series. In *Proceedings of the 5th European Conference on Principles of Knowledge Discovery in Databases*, pages 192–203, 2001.

[11] F. Höppner and F. Klawonn. Finding informative rules in interval sequences. In *Proceedings of the 4th International Symposium on Advances in Intelligent Data Analysis*, pages 123–132, 2001.

[12] 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.

[13] P. Kam and A. W. Fu. Discovering temporal patterns for interval-based events. In *Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery*, pages 317–326, 2000.

[14] R. Kosara and S. Miksch. Visualizing complex notions of time. *Studies in Health Technology and Informatics*, pages 211–215, 2001.

[15] O. Kostakis, P. Papapetrou, and J. Hollmén. Artemis: Assessing the similarity of event-interval sequences. In *Proceedings of the Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2011)*, pages 229–244, 2011.

[16] O. Kostakis, P. Papapetrou, and J. Hollmén. Distance measure for querying arrangements of temporal intervals. In *Proceedings of Pervasive Technologies Related to Assistive Environments*, 2011.

[17] 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.

[18] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics*, 10(8):707–710, 1966.

[19] J.-L. Lin. Mining maximal frequent intervals. In *Proceedings of the 18th ACM Symposium On Applied Computing*, pages 624–629, 2003.

[20] C. Mooney and J. F. Roddick. Mining relationships between interacting episodes. In *Proceedings of the 4th SIAM International Conference on Data Mining*, 2004.

[21] F. Mörchen. Unsupervised pattern mining from symbolic temporal data. *SIGKDD Exploration Newsletter*, 9:41–55, June 2007.

[22] F. Mörchen. Temporal pattern mining in symbolic time point and time interval data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, KDD '10, pages 2:1–2:1. ACM, 2010.

[23] F. Mörchen and D. Fradkin. Robust mining of time intervals with semi-interval partial order patterns. In *Proceedings of the 10th SIAM International Conference on Data Mining*, pages 315–326, 2010.

[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 *Proceedings of the IEEE ICDM Workshop on Data Mining in Bioinformatics*, 2006.

[26] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Discovering frequent arrangements of temporal intervals. In *Proceedings of 5th IEEE International Conference on Data Mining*, pages 354–361, 2005.

[27] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems*, pages 133–171, 2009.

[28] D. Patel, W. Hsu, and M. Lee. Mining relationships among interval-based events for classification. In *Proceedings of the 28th ACM SIGMOD International Conference on Management of Data*, pages 393–404. ACM, 2008.

[29] D. Patel, W. Hsu, and M. L. Lee. Mining relationships among interval-based events for classification. In *Proceedings of ACM SIGMOD*, pages 393–404, 2008.

[30] N. Pissinou, I. Radev, and K. Makki. Spatio-temporal modeling in video and multimedia geographic information systems. *GeoInformatica*, 5(4):375–409, 2001.

[31] 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.

[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.