Marking Time in Sequence Mining

Carl H. Mooney and John F. Roddick

School of Informatics and Engineering
Flinders University of South Australia,
PO Box 2100, Adelaide, South Australia 5001,
Email: {carl.mooney, roddick}@infoeng.flinders.edu.au

Abstract

Sequence mining is often conducted over static and temporal datasets as well as over collections of events (episodes). More recently, there has also been a focus on the mining of streaming data. However, while many sequences are associated with absolute time values, most sequence mining routines treat time in a relative sense, only returning patterns that can be described in terms of Allen-style relationships (or simpler).

In this work we investigate the accommodation of timing marks within the sequence mining process. The paper discusses the opportunities presented and the problems that may be encountered and presents a novel algorithm, $INTEM_{TM}$, that provides support for timing marks. This enables sequences to be examined not only in respect of the order and occurrence of tokens but also in terms of pace. Algorithmic considerations are discussed and an example provided for the case of polled sensor data.

1 Introduction

Frequent-pattern (sequence) mining from static databases has been conducted for a number of years and algorithms for this form of mining are relatively mature (Pei et al. 2001, Srikant & Agrawal 1996, Wang & Han 2004, Yan et al. 2003). Transaction datasets commonly include a *time-stamp* for each transaction and it is this that can be used, in conjunction with a *transaction_id*, to constrain the mining activity with respect to time.

However, sequence mining is not limited to data stored in transaction-structured datasets and there are other domains where an implicit time-stamp may or may not be included such as web logs, alarm data in telecommunications networks, sensor data, and so on. In such domains, the data can be viewed as a series of events occurring at specific times and therefore the problem becomes a search for collections of events (episodes) that occur frequently together. Solving this problem requires a different approach, and several types of algorithm have been proposed for different domains (Mannila & Toivonen 1996, Mannila et al. 1997, Mooney & Roddick 2004, Spiliopoulou 1999).

Such datasets can also be very similar in nature to, or are themselves, streaming datasets, an area of research that is gaining significant interest at present (Gaber et al. 2005, Giannella et al. 2003, Lin et al. 2003). However, the datasets used in these domains do not always include a *time-stamp* and this reduces the problem to those that occur close to each other in the sequence. This changes the semantics of *frequent* and makes mining more problematic if time constraints are required, or if information relative to the pace of the activity is of interest. However, in some datasets, the passage of time, while not being available as a full time-stamp, may be marked by a token representing a *timing tick*.

In this paper we address this problem by introducing the notion of a ***timing mark*** (or ***timing tick***) to accommodate the passage of time within the sequence mining process. This allows the process not only to provide information relative to order and occurrence of sequences but also the pace at which they occurred.

The remainder of the paper is organised as follow. Section 2 briefly discusses background material on sequence mining and related work. Section 3 introduces the concept of the *timing mark* and discusses the opportunities presented and potential problems that may be encountered. Section 4 deals with algorithmic considerations and presents a novel algorithm $INTEM_{TM}$ that provides support for the concept of *timing marks*. Section 5 provides experimental results resulting from the implementation while Section 6 offers some conclusions and suggestions for future work.

2 Background and Related Work

The sequential pattern mining problem can be viewed from both a static dataset and episodic point of view; the latter being the area most closely related to the mining discussed in this work. We outline below some definitions of the related areas and previous research in sequence, episodic and time series mining is briefly discussed.

2.1 Sequential Pattern Mining

Given a dataset of sequences, where each sequence consists of a list of transactions ordered by transaction time and each transaction is a set of items, the aim of sequential pattern mining is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data-sequences that contain that pattern (Agrawal & Srikant 1995).

The problem of mining sequential patterns can be stated as follows: Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, termed *items*, which comprise the alphabet. An *event* is a non-empty unordered collection of items. It is assumed without loss of generality that items of an event are sorted in lexicographic order. A *sequence* is an ordered list of events. An event is

denoted as $(i_1, i_2, \ldots, i_k)$, where i_j is an item. A sequence α is denoted as $\langle \alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_q \rangle$, where α_i is an event. A sequence with k-items ($k = \sum_j |\alpha_j|$) is called a *k-sequence*. For example, $\langle B \rightarrow AC \rangle$ is a 3-sequence. A sequence $\langle \alpha_1 \rightarrow \alpha_2 \ldots \rightarrow \alpha_n \rangle$ is a *subsequence* of another sequence $\langle \beta_1 \rightarrow \beta_2 \ldots \rightarrow \beta_m \rangle$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $\alpha_1 \subseteq \beta_{i_1}, \alpha_2 \subseteq \beta_{i_2}, \ldots, \alpha_n \subseteq \beta_{i_n}$. For example the sequence $\langle B \rightarrow AC \rangle$ is a subsequence of $\langle AB \rightarrow E \rightarrow ACD \rangle$, since $B \subseteq AB$ and $AC \subseteq ACD$, and the order of events is preserved. However, the sequence $AB \rightarrow E$ is not a subsequence of ABE and vice versa.

The process is provided with a dataset $\mathcal{D}$ of input-sequences where each input-sequence in the dataset has the following fields: sequence-id, event-time and the items present in the event. It is assumed that no sequence has more than one event with the same time-stamp, so that the time-stamp may be used as the event identifier. In general, the *support* or *frequency* of a sequence, denoted $\sigma(\alpha, \mathcal{D})$, is defined as the total number of input-sequences in the dataset $\mathcal{D}$ that contain α[1]. Given a user-specified *minimum support* threshold (denoted *min_supp*), a sequence is said to be *frequent* if it occurs at least *min_supp* times and the set of frequent k-sequences is denoted as $\mathcal{F}_k$. A frequent sequence is deemed to be *maximal* if it is not a subsequence of any other frequent sequence. The task then becomes the discovery of all frequent sequences from a dataset $\mathcal{D}$ of input-sequences and a user supplied *min_supp*.

2.2 Episodic Mining

The first algorithmic framework developed to mine datasets that were episodic in nature was introduced by Mannila et al. (1995). The task was to find all episodes that occur frequently in an event sequence, given a class of episodes and an input sequence of events. In their framework an episode is defined to be:

> *"... a collection of events that occur relatively close to each other in a given partial order*, and ... *frequent episodes as a recurrent combination of events"* (Mannila et al. 1997)

The notation used is as follows.

E is a set of event types and an event is a pair (A, t), where $A \in \mathbf{E}$ is an event type and t is the time (occurrence) of the event. There are no restrictions on the number of attributes that an event type may contain, but the paper only considers single values with no loss of generality. An event sequence **s** on **E** is a triple (s, T_s, T_e), where $s = \langle (A_1, t_1), (A_2, t_2), \ldots, (A_n, t_n) \rangle$ is an ordered sequence of events such that $A_i \in \mathbf{E}$ for all $i = 1, \ldots, n$, and $t_i \leq t_{i+1}$ for all $i = 1, \ldots, n-1$. Further $T_s < T_e$ are integers, T_s is called the starting time and T_e the ending time, and $T_s \leq t_i < T_e$ for all $i = 1, \ldots, n$.

2.3 Time Series Mining

Data mining of time series datasets includes not only sequence mining but also clustering, classification, and association mining (Lin et al. 2003, Das et al. 1998, Guralnik & Srivastava 1999, Höppner 2001, Keogh et al. 1993). As would be expected, the constraints available each time are those appropriate for the form of mining and the rules that emerge from this type of analysis are similarly aligned with the mining method chosen. For the case of sequences, typical rules are based on (*a priori* supplied) calendric, or cyclic patterns and have some similarity to those addressed in this paper.

[1] This general definition has been modified as algorithmic development has progressed and different methods for calculating support have been introduced, see the work of Joshi et al. (1999) for a complete summary of counting techniques.

3 Timing Marks

The concept of *timing marks* introduced here refers to embedded tokens that indicate the passage of time. They are not *time-stamps* in that they do not record absolute time values but rather *ticks* which can be referenced to determine the pace of events [2]. For example, the notion of polled data infers a (fixed) time interval during which the polling occurs. During this interval, it may be possible that not all sensors are read and/or some do not return data. Moreover, many sequences of events have a time-stamp, either inherently in how they are reported, or overlaid by a system that needs to interrogate the data. How this fixed time element is encoded in the data is of interest. In traditional sequence mining, time-series mining and web-log mining each element to be mined has a time-stamp associated with it and therefore encoding an additional *timing mark* is not necessary. With sensor data and other data streams there is usually no time-stamp and therefore it is necessary to include a time-stamp or *timing mark* into the data.

In our recent work on mining interacting episodes we have implicitly assumed (in common with other researchers) that each token (sensor reading) occurred for a fixed period of time and that the time between tokens was zero (or alternatively, that events are instantaneous and the time between tokens was constant). That is, we could view a sequence of n tokens as occurring over n time periods of equal length, no matter what the period/granularity was (Mooney & Roddick 2004). This work relaxes this assumption. That is, although the time between events may remain unchanged – equal length intervals – the number of tokens (events) that occur within that time can vary. To accommodate this assumption we have introduced *timing marks* into the data. These *timing marks* may have different properties depending on the data they are associated with and more generally *timing marks* can be viewed as having the following possibilities[3]:

Timing marks as tokens. One of the polled sensors is used as the timing mark which would mean that all time-referenced sequences would be reported with reference to this sensor. One problem with this option is that the sensor used as the timing mark may not fire regularly and as such any rules that are reported may or may not have value. If however the sensor is firing in every cycle then its usefulness from a reporting standpoint is valuable in the same way as if it were a delimiter.

Timing marks added as delimiters. In this option, timing marks are added as additional tokens to the sequence. This is necessary where all other tokens are sporadic as is the case with many types of sequence. For our purposes, this is our assumed format.

[2] The consensus glossary (Jensen et al. 1998) delineates between two forms of time - absolute and relative. Timing marks are, in many respects, both and neither of these possessing an absolute time period relative to each other but little else. Certainly, little of the current temporal data mining research (Roddick & Spiliopoulou 2002) can handle timing marks.

[3] In the following examples the period '.' is used as the notation of the *timing mark*.

Timing marks as absolute time. In some cases, each token carries with it an absolute time stamp. In this case there is more information that is required for our purposes here and it would be trivial to convert such a sequence to one that contained timing marks as delimiters.

The value of timing marks becomes apparent when queries can be issued and results reported with respect to timing marks. A given sequence, for example, could be deemed as "fast/bursty" or as "slow". For instance, the difference between the two sequences **ABC** and **A...B..C** may be significant even if they occur within the normal lookahead.

More significantly, the semantics of rules using temporal relationships such as $A - during \rightarrow B$ or $A - meets \rightarrow B$ may change depending on the number of timing marks that have been encountered. For instance, to allow for recording latency, two intervals may be deemed to *meet* if they occur within n timing marks.

4 Algorithmic Considerations

Timing marks can be either present or absent in a data stream and as such users should have the opportunity to include or exclude the timing marks in their search for frequent episodes. Consequently, the timing marks feature has necessarily been implemented as a constraint, thus allowing the user to select the token that is the timing mark and, in addition, choose whether to report those episodes that contain exactly the prescribed number of timing marks or all episodes up to and including the prescribed number of timing marks.

Since we provide the user with the choice, it makes sense for the implementation of this constraint to be post episode discovery. To further reinforce this decision, the token used for the timing mark may be one of the data tokens, not one that is orthogonal to the data, in which case the user may not wish to remove the token from those episodes that are reported. In order to facilitate the fact that the timing mark may be one of the data tokens, the input file is scanned upon selection to generate a list of those tokens available. This incurs no overhead since the file has to be read before further processing can be undertaken. Since this is an added constraint, the impact on the existing algorithm is minimal.

The two parameters used are:

- The **lookahead** (or window) parameter used in previous work (Mooney & Roddick 2004) (similar to Mannila *et al.*'s window concept (Mannila et al. 1997)), defines the maximum length episode to be mined) is included, together with
- a *timing mark count* (**tmc**), which defines either the maximum number of timing marks that can be included *or* the exact number of timing marks that should occur in the sequence.

Since both of these measures can be used, for the purpose of "frequent", a sequence up to *lookahead* must also occur within the prescribed number of marks – ie. the cut-off is either the *lookahead* or *timing mark count* whichever is the smaller.

4.1 Timing Mark Pruning

If the user has chosen to include the timing marks in their search then the following will occur after the frequent episodes have been discovered. First, pruning will be conducted on the frequent sequences so that only those that contain the prescribed number of timing marks, see Algorithm 4.1, will remain. If the timing mark is not determined to be one of the tokens in the data, then removal of the timing marks from those remaining sequences will ensue, and finally reassignment of them to the correct output containers, see Algorithm 4.2.

For timing marks to remain unambiguous to the user and therefore be consistent throughout the application then the following convention is adopted:

1) Within one mark means that there are no *timing marks* allowed in the sequence. Algorithmically this can be described by – assuming the timing mark is "." –

   if (tmc = 1 $\wedge$ cand.indexOf(".") $\neq -1$) **then**
   set output to null
   end if
   return

   This also leads to an added pruning technique – i.e. in the case of one timing mark, if we are looking for an x length sequence and the last item in the sequence is a *timing mark*, then the next x sequences are not viable candidates so can be eliminated from the search.

2) Within one or more *timing mark*s. During one timing mark is as described above while n marks indicates that there are n distinct sections in the sequence which would have embedded $n-1$ timing marks.

Algorithm 4.1 Algorithm for imposing timing marks on sequence discovery

Input: A set of frequent sequences that are to be pruned for timing marks.
Output: the collection of frequent sequences according to the timing mark constraints.

```
1: procedure PRUNEFORTIMINGMARKS(ArrayList aList)
2:     for (i := 0; i < aList.size(); i++) do
3:         TreeMap tm := aList.get(i);
4:         TreeMap clTm := tm.clone();
5:         for all (String cand : clTm.keySet()) do
6:             int numMarks = countTimingMarks(cand);
7:             if (exactly_selected) then
8:                 if (numMarks ≠ maxMarks) then
9:                     tm.remove(cand);
10:                end if
11:            else
12:                if (numMarks > maxMarks) then
13:                    tm.remove(cand);
14:                end if
15:            end if
16:        end for
17:    end for
18: end procedure
```

Algorithm 4.2 Removes the timing marks from the frequent sequences and reassigns them to the correct output containers.

Input: a list of frequent episodes that have been pruned for the required number of timing marks.
Output: the required frequent episodes without timing marks.

```
1: procedure REMOVEALLTIMINGMARKS(ArrayList aList)
2:     ArrayList modList := new ArrayList();
3:     for all (TreeMap tmap : aList) do
4:         TreeMap modTree := new TreeMap();
5:         for all (String cand : tmap.keySet()) do
6:             String newCand := removeTimingMarks(cand);
7:             if (!newCand.equals("")) then
8:                 modTree.put(newCand, tmap.get(cand));
9:             end if
10:        end for
11:        modList.add(modTree);
12:    end for
13:    frequentList := reassignEpisodes(modList);
14: end procedure
```

4.2 Rule semantics

Typically rules from sequence discovery are of the type that can described in terms of Allen-style (Allen 1983) relationships (or simpler). This is the case not only for market-basket mining (Agrawal & Srikant 1995, Ayres et al. 2002, Garofalakis et al. 1999, Han et al. 2000), but also episodic mining (Mannila et al. 1997, Mooney & Roddick 2004). In the case of episodic mining both parallel and serial episodes yield these types of rules. When using *timing marks* as delimiters the following possibilities, similar to those of episodic mining, must be considered:

1) if the sequences occur within the interval delimited by a pair of *timing marks*, for example, **.ABCDEF.**, then this is analogous to parallel episodes[4], or

2) if the sequences must occur within a certain number of *timing marks*. For example, **.AB.CDE.F.**, then it is analogous to serial episodes.

In the first case above, the discovered sequences could be treated as transactions (if order is irrelevant) and therefore further processing may be conducted using other data mining methods, such as association rule mining (Ceglar & Roddick 2006). In the second case details about the 'speed' of discovered sequences can be obtained with respect to the number of timing marks that are contained, allowing for a better understanding of the data.

If the *timing mark* is viewed as a fixed length period with no absolute time-stamp associated with it then we can search for sequences that occur under both of the above conditions. For example, given the sequence **.ABCDEFG.HIJ..** with a maximum look ahead of 5:

1) For sequences that occur within one cycle – **ABCDE**, **BCDEF**, and **CDEFG** are all valid while **FGHIJ** is not. This may be useful to determine if certain sensors did not fire during a particular cycle.

2) For sequences that occur over a period of x time cycles – sequence **FGHIJ** occurs over two time cycles.

Given this information, the knowledge of the position of the *timing mark* allows for added semantics to be attached to the sequence – not only can we say that **FGHIJ** occurs over two time cycles but also that first cycle is ended with **FG** and the second is begun with **HIJ**. This may have added interest, depending on the application, to any resulting output that may be derived.

5 Experimental Results

The algorithm was implemented in the $Java^{TM}$ programming language and all experiments were conducted on a 2.6GHz AMD machine running $Windows^{®}\ XP$ with 1Gb of RAM (see Figure 1). The $INTEM_{TM}$ implementation represents an extension to the $INTEM$ (**INT**eracting **E**pisode **M**iner), which also includes graphical output, as well as text, of the discovered sequences. Furthermore, interactions may also be discovered and reported using Allen-style relationships (Allen 1983), Freksa's conceptual neighbourhoods (Freksa 1992) or more fine grained Midpoint relationships (Roddick & Mooney 2005). Since the user has control over the number and method of timing mark inclusion the "speed" of the discovered sequences is known and can be reported easily.

[4]This would be data dependent and would rely on whether the order within the marks is relevant.

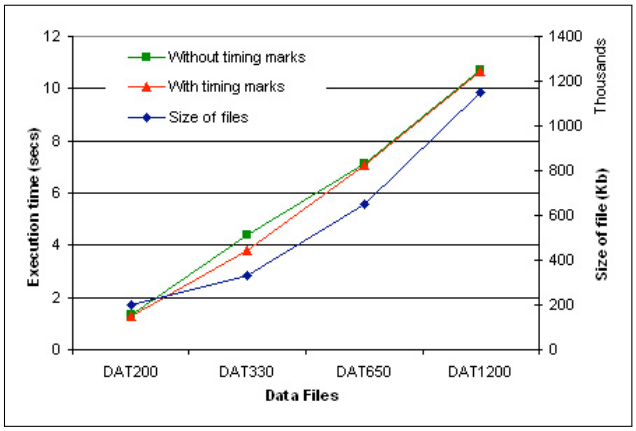

Figure 2: Execution times with and without timing marks for the test files.

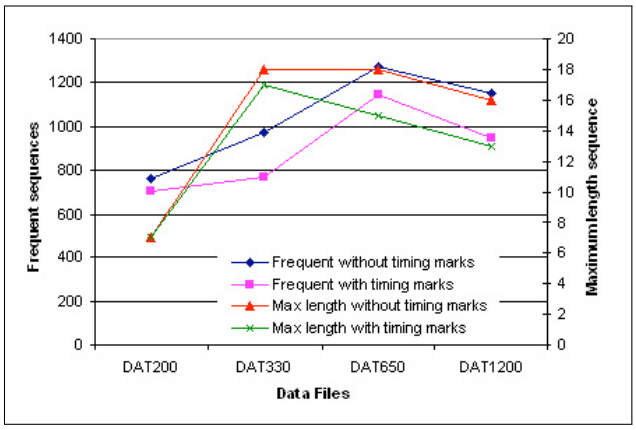

Figure 3: Number of frequent sequences and maximum length sequences with and without timing marks.

All tests were conducted using a low support level (0.005), a *lookahead* value of 20 and, when including timing marks, a *timing mark count* (tmc) of 2 with the reporting option set to exclude the marks from the output. The results show that there is no overhead incurred when using the timing mark option, see Figure 2. Indeed, since the constraint is implemented deeper in the process there is a slight speed increase when looking for sequences containing timing marks. The reason for the speed up can also be seen, (see Figure 3), by e fact there are less sequences discovered with the timing mark option selected and in the majority of cases the maximum length of the discovered sequences is smaller.

6 Conclusions and Future Work

In this paper we have discussed the inclusion of timing marks for dealing with data that have no absolute time attached to the events to be mined. We have shown that the implementation of the algorithm incurs negligible added overhead and that the benefits associated with the rules that may be reported are important in terms of being able to determine the pace of a sequence.

Future research is necessary in this area to accommodate this feature into algorithms that can deal with streaming data – an already complex domain (see (Gaber et al. 2005)) Further research is also needed in the area of rule generation, together with some consideration of the resultant semantics of the

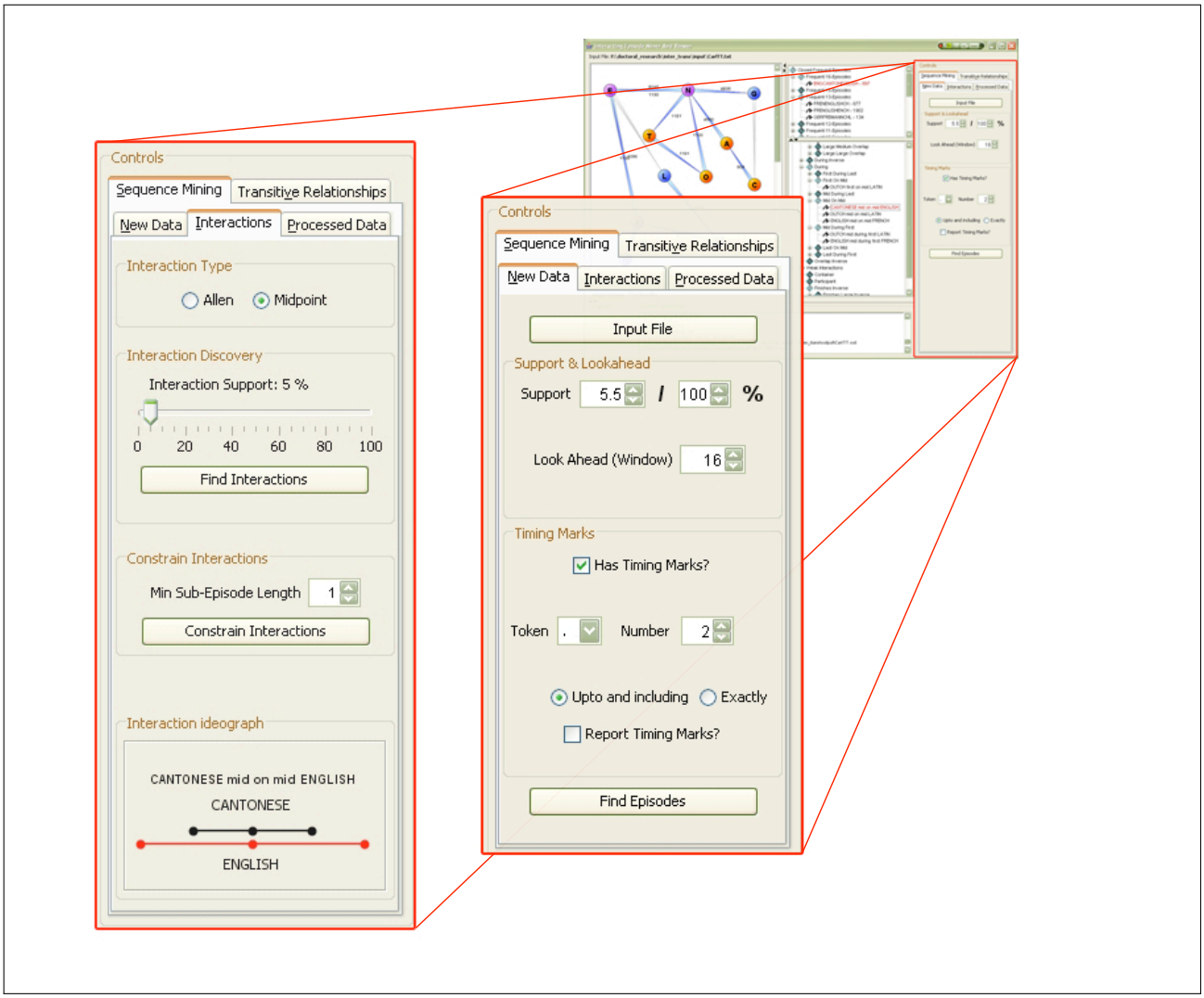

Figure 1: Screenshot of the experimental system.

rules. This latter issue is highly dependent on the data being mined and therefore careful consideration is needed.

References

Agrawal, R. & Srikant, R. (1995), Mining sequential patterns, *in* P. S. Yu & A. S. P. Chen, eds, '11th International Conference on Data Engineering (ICDE'95)', IEEE Computer Society Press, Taipei, Taiwan, pp. 3–14.

Allen, J. (1983), 'Maintaining knowledge about temporal intervals', *Communications of the ACM* **26**(11), 832–843.

Ayres, J., Flannick, J., Gehrke, J. & Yiu, T. (2002), Sequential pattern mining using a bitmap representation, *in* '8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM Press, Edmonton, Alberta, Canada, pp. 429–435.

Ceglar, A. & Roddick, J. F. (2006), 'Association mining', *ACM Computing Surveys* **38**(2).

Das, G., Lin, K.-I., Mannila, H., Renganathan, G. & Smyth, P. (1998), Rule discovery from time series, *in* '4th International Conference on Knowledge Discovery and Data Mining (KDD-98)', AAAI Press.

Freksa, C. (1992), 'Temporal reasoning based on semi-intervals', *Artificial Intelligence* **54**(1-2), 199–227.

Gaber, M. M., Zaslavsky, A. & Krishnaswamy, S. (2005), 'Mining data streams: a review', *SIGMOD Record* **34**(2), 18–26.

Garofalakis, M. N., Rastogi, R. & Shim, K. (1999), SPIRIT: Sequential pattern mining with regular expression constraints, *in* M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik & M. L. Brodie, eds, '25th International Conference on Very Large Data Bases, VLDB'99', Morgan Kaufmann, Edinburgh, Scotland, UK, pp. 223–234.

Giannella, C., Han, J., Pei, J., Yan, X. & Yu., P. (2003), Mining frequent patterns in data streams at multiple time granularities, *in* H. Kargupta, A. Joshi, K. Sivakumar & Y. Yesha, eds, 'Next Generation Data Mining'.

Guralnik, V. & Srivastava, J. (1999), Event detection from time series data, *in* S. Chaudhuri & D. Madigan, eds, '5th International Conference on Knowledge Discovery and Data Mining', ACM Press, San Diego, CA, USA, pp. 33–42.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. & Hsu, M.-C. (2000), Freespan: frequent pattern-projected sequential pattern mining, *in* '6th ACM SIGKDD International Conference

on Knowledge Discovery and Data Mining', ACM Press, Boston, MA, USA, pp. 355–359.

Höppner, F. (2001), 'Discovery of temporal patterns - learning rules about the qualitative behaviour of time series'.

Jensen, C. S., Clifford, J., Elmasri, R., Gadia, S. K., Hayes, P., Jajodia, S., Dyreson, C., Grandi, F., Kafer, W., Kline, N., Lorentzos, N., Mitsopoulos, Y., Montanari, A., Nonen, D., Peressi, E., Pernici, B., Roddick, J. F., Sarda, N. L., Scalas, M. R., Segev, A., Snodgrass, R. T., Soo, M. D., Tansel, A., Tiberio, P. & Wiederhold, G. (1998), A consensus glossary of temporal database concepts - february 1998 version, *in* O. Etzion, S. Jajodia & S. Sripada, eds, 'Temporal Databases - Research and Practice', Vol. 1399, Springer, pp. 367–405.

Joshi, M. V., Karypis, G. & Kumar, V. (1999), Universal formulation of sequential patterns, Technical Report Under Preparation #99-21, Department of Computer Science, University of Minnesota.

Keogh, E., Chu, S., Hart, D. & Pazzani, M. (1993), Segmenting time series: A survey and novel approach, *in* 'Data Mining in Time Series Databases', World Scientific Publishing Company.

Lin, J., Keogh, E., Lonardi, S. & Chiu, B. (2003), A symbolic representation of time series, with implications for streaming algorithms, *in* '8th ACM SIGMOD Workshop on Research issues in Data Mining and Knowledge Discovery, DMKD'03', ACM Press, pp. 2–11.

Mannila, H. & Toivonen, H. (1996), Discovering generalised episodes using minimal occurrences, *in* '2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)', AAAI Press, Menlo Park, Portland, Oregon, pp. 146–151.

Mannila, H., Toivonen, H. & Verkamo, A. I. (1995), Discovering frequent episodes in sequences, *in* U. M. Fayyad & R. Uthurusamy, eds, '1st International Conference on Knowledge Discovery and Data Mining (KDD-95)', AAAI Press, Menlo Park, CA, USA, Montreal, Quebec, Canada, pp. 210–215.

Mannila, H., Toivonen, H. & Verkamo, A. I. (1997), 'Discovery of frequent episodes in event sequences', *Data Mining and Knowledge Discovery* **1**(3), 259–289.

Mooney, C. H. & Roddick, J. F. (2004), Mining relationships between interacting episodes, *in* M. W. Berry, U. Dayal, C. Kamath & D. Skillicorn, eds, '4th SIAM International Conference on Data Mining (SDM'04)', SIAM, Lake Buena Vista, Florida.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. & Hsu, M.-C. (2001), PrefixSpan: Mining sequential patterns efficiently by prefix projected pattern growth, *in* '2001 International Conference of Data Engineering (ICDE'01)', Heidelberg, Germany, pp. 215–226.

Roddick, J. F. & Mooney, C. H. (2005), 'Linear temporal sequences and their interpretation using midpoint relationships', *IEEE Transactions on Knowledge and Data Engineering* **17**(1), 133–135.

Roddick, J. F. & Spiliopoulou, M. (2002), 'A survey of temporal knowledge discovery paradigms and methods', *IEEE Transactions on Knowledge and Data Engineering* **14**(4), 750–767.

Spiliopoulou, M. (1999), Managing interesting rules in sequence mining, *in* 'Principles of Data Mining and Knowledge Discovery', pp. 554–560.

Srikant, R. & Agrawal, R. (1996), Mining sequential patterns: generalisations and performance improvements, *in* P. M. G. Apers, M. Bouzeghoub & G. Gardarin, eds, 'International Conference on Extending Database Technology, EDBT'96', Vol. 1057 of *LNCS*, Springer, Avignon, France, pp. 3–17.

Wang, J. & Han, J. (2004), BIDE: Efficient mining of frequent closed sequences, *in* '20th International Conference on Data Engineering', IEEE Press, pp. 79–90.

Yan, X., Han, J. & Afshar, R. (2003), Clospan: Mining closed sequential patterns in large datasets, *in* '3rd SIAM International Conference on Data Mining (SDM'03)', San Francisco, CA.