Ramex: A Sequence Mining Algorithm Using Poly-trees

Luís Cavique

Departamento de Ciências e Tecnologia, Universidade Aberta, Portugal
lcavique@uab.pt

Abstract. Sequence mining combines the discovery of frequent itemsets and the order they appear in. Most of the sequence pattern discovery techniques present some handicaps like the generation of a huge number of rules and the lack of scalability. In this work the proposed algorithm concerns the analysis of the whole rather than the parts, thus providing a holistic view of the sequences. The algorithm analyzes event logs and allows a non-expert user to understand the sequences using a poly-tree visualization. The scalability associated with condensed data structures, which shrink the data without losing information, allows dealing with the Big Data challenge. Ramex was implemented in different scenarios.

Keywords: pervasive business intelligence, sequence mining, poly-trees.

1 Introduction

The emergence of the Internet has changed the way people interact with computers. People can access information from personal computers or mobile devices (smart phones or tablets) anytime and anywhere. The mobile, or ubiquitous dimension associated with the Web 2.0 and the *Internet of things* generate a large volume of data with an increasing updating velocity. The exponential growth of data, when compared to the linear growth of processing capabilities, leads to a decline in the capacity to extract useful knowledge from the stored data.

In order to understand the activities of complex computer systems and the eventual diagnosis of problems, keeping a record of event logs is an essential task. Accordingly, the storage and analysis of event logs is a pertinent challenge.

Pervasive Information Systems aim to study how information environments affect human interactions. Pervasive spaces go beyond the Human-Computer Interaction (HCI) in order to create socio-technical systems that benefit stakeholders and users. In particular, Pervasive Business Intelligence looks for holistic views which combine information from different latencies [17].

Most of the sequence pattern discovery techniques present three common handicaps: the need of parameters, the huge number of rules that do not permit a global view and the scalability problems:

i) Parameters: The user must specify a minimum support threshold to find the desired patterns. A useless output can be expected by pruning either too many or too few items. The process must be repeated interactively, which becomes very time consuming for large databases.

A. Rocha et al. (eds.), *New Contributions in Information Systems and Technologies,*
Advances in Intelligent Systems and Computing 354, DOI: 10.1007/978-3-319-16528-8_14

ii) Number of Rules: The association rules' systems that support the itemset and sequence mining usually generate a huge number of rules, and therefore, it is difficult for the user to decide which rules to use.

iii) Scalability: Since most of the existing algorithms use a lattice structure in the search space and need to scan the database more than once, they are not compatible with very large databases.

In order to create holistic environments in data mining, micro-patterns and macro-patterns must be differentiated [7]. The micro-patterns correspond to small percentages of data; for instance, in association rules, it is usual to have a measure of support that includes support values $\geq 5\%$, with high confidence rules being chosen. On the other hand, the macro-patterns involve a large percentage of data, for example in the regression model all data elements are used. The micro-patterns are characterized by high confidence, while macro-patterns are characterized by high support. There are other examples of micro-patterns: in sequence mining a support $\geq 1\%$ is frequent; in the classification problem, by using decision trees, each branch of the tree corresponds to a small percentage of the data; in the classification problem using the k-nearest neighbor, the comparisons are made using a reduced number of k elements. Finally, regarding macro-patterns in techniques such as regression, hypothesis testing, clustering or reduction of attributes, all data are taken into account.

The proposed algorithm was coined with the Latin name, Ramex, meaning “branch of a tree”. Ramex introduces a new vision for classic problems of sequence mining, considering the accumulation of events, and allowing the search of macro-patterns, instead of searching only for micro-patterns.

Ramex provides a comprehensive view of the sequences, providing the user the visualization of the data sequences with a special kind of tree, a poly-tree, which shows all the items, but only the most relevant sequences retrieving the x-ray of the dataset.

Ramex has been implemented in different scenarios: web mining [6] and financial studies [16], [20]. This paper also includes part of the work of [8].

In Section 2, the related work with sequence and process mining is presented. In Section 3, the proposed algorithm is reported and a numeric example is presented. In Section 4, computational results are reported using the IBM Quest Synthetic datasets generator. Finally, in Section 5, we draw some conclusions.

2 Related Work

In this section concepts are presented and definitions are established in order to be reused in the following sections. Sequence Mining is referred, Process Mining is introduced and the definition of Poly-tree is established.

2.1 Sequence Mining

The problem of pattern discovery is to extract interesting patterns from the data. It is difficult to define what the ingredients of an interesting pattern are. The temporal data mining can be divided into four different approaches: Periodic Patterns [22], Sequential Discovery [4], Frequent Episodes [15] and Markov Chain Models [5].

The first approach to this problem was given by the KMP string matching algorithm [14], where the aim is to find a short pattern in a long text. The concept of meta-pattern was presented in order to enable the representation of a set of basic patterns in a concise way [22]. The meta-pattern model works in a structured framework similar to a grammar structure, where the meta-pattern may be composed of patterns or meta-patterns.

Sequence Mining Discovery involves a sequence of steps in time that can be exemplified with a customer transaction event log. The frequent temporal pattern might express buying patterns that many customers exhibit. Most of the Frequent Sequence Mining algorithms use lattice structures in the search space. These algorithms include for the breadth-first search, the AprioriAll algorithm [4] and the GSP (Generalized Sequential Pattern) [19] and for the depth-first search the SPADE (Sequential PAttern Discovery using Equivalence classes) [23].

Another approach to discover temporal patterns in sequential data is the frequent episode discovery presented by [15]. In this sequential pattern framework, a set of events is given and the aim is to discover frequent sequences, or frequent episodes, that occur in the timeline. The data referred here as an event sequence are denoted by E={(e(1),t(1)), (e(2),t(2)), …(e(n), t(n))} where e(i) takes values from an event-type set and t(i) is an integer denoting the time stamp of the i^{th} event.

The Markov chain is an acyclic graph with a set of states associated with a set of transitions between states. In the market basket each state corresponds to an item and in the user web navigation each state is a page. Markov models have been used to represent and analyze users' web navigation data [5]. At each time interval the system can change from a current state to the next state. The transition between states is quantified using probabilities. For each state in the transition matrix P(i,j) is the probability of moving from state *i* to state *j* in one step, and the sum of the probabilities of each state is equal to one. The first-order chain, is usually represented by a square matrix of the probabilities of the transitions from the current states to the following states. A second-order Markov chain takes into account the previous and the current state probability of the transition to the next state. The nth-order matrix, with n>1, grows into new dimensions ceasing to be square, as the first-order matrix. The higher-order chains consider sets of previous states leading to ordered sequences of states, while expanding the dimension of the original matrix and also the complexity of the problem.

2.2 Process Mining

Process Mining [1] is a recent approach that aims to create a bridge between Data Mining and Business Process Modelling (BPM) by discovering paths in event log data.

The Petri nets technique is an established tool for modelling processes that is often used in Process Mining. Petri nets like the Markov chain allows an aggregate view of the process workflow.

The Process Mining Manifesto [3] is supported by 53 organizations, 77 expert consultants and is available in 13 languages. The manifesto presents six guiding principles and eleven challenges, clarifying the aim of this new concept.

2.3 Graphs, Trees and Poly-trees Concepts

Given a connected undirected graph, a spanning tree of the graph is a sub-graph that connects all the vertexes. A minimum weight spanning tree is the spanning tree with a weight that is lower than or equal to the weight of every other spanning tree. This problem is easily solved using a greedy algorithm. The algorithms proposed by Kruskal and Prim are well-known examples [9]. In Kruskal's algorithm, the edges are chosen without worrying about the connections to previous edges, but avoiding the cycles. In Prim's algorithm the tree grows from an arbitrary root.

An oriented tree is a direct acyclic graph with a node in-degree equal to one, with the exception of the root, where the in-degree is equal to zero.

A poly-tree is a relaxed oriented tree, with one (and only one) path between any pair of nodes, where the in-degree of a node can be greater than one. The term poly-tree was coined by [18] and the poly-tree algorithm is well-known for inference in Bayesian Networks.

A tree, with a node in-degree equal to one, a poly-tree, with a node in-degree greater than one and a direct acyclic graph, with two or more paths between a pair of nodes are shown in Figure 1.

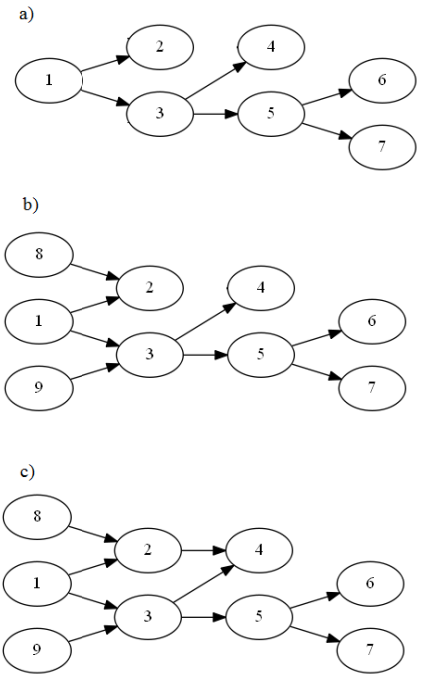

Fig. 1. (a) tree, (b) poly-tree and (c) direct acyclic graph

In a connected directed graph G, an acyclic sub-graph B is a branch in G if the in-degree of each vertex is at the most 1. Let w(B) be the sum of the weighted arcs in branch B. The maximum weight branching problem is the optimization problem that finds the largest possible branch B and was proposed in [10] known as Edmonds' branching algorithm. The algorithm is described in two steps: the condensation process, to remove the cycles, and the unraveling process where the branch is created.

Fulkerson [11] presents the same problem with an additional constraint, the branch must start in a vertex called the root. His algorithm for the maximum packing rooted directed cuts in a graph is equal to the weight of the minimum spanning tree directed away from the root. The algorithm is also described in two steps.

Both algorithms [10] and [11] generated trees, with in-degree zero or one. To find the maximum weighted poly-tree, there are few or inexistent bibliographic references.

3 Ramex Algorithm

The aim of this approach is to create a poly-tree of events, with as many branches as needed to visit all the vertexes. The Ramex algorithm uses a Poly-tree Sequence model with two phases: the transformation of the problem into a network and the search of the sequences.

Algorithm 1. Ramex, Two Phase Algorithm to get the Poly-tree Sequence Model

Input: raw data (#id, event stream);
Output: a poly-tree of events;
1) Problem transformation, by creating next-events and accumulating into the network
2) Search for the most weighed Poly-tree sequence of events

3.1 Problem Transformation

The event log is represented in a structure with two columns (#id, event stream). Our approach uses a two-phase algorithm: the transformation of a database into a network and the search of the maximum weighted poly-tree.

For each line in the table, starting with the first event, a new attribute next-event is created, and continues until the last event. Then a network, i.e. a graph G with a source and a sink is created, where each sequence has an initial node called source and a final node called sink. The network, where cycles are allowed, condenses the information of the database by incorporating all the event sequences. In network G each state corresponds to an event and each transition represents the sequence from one event to the next-event. The weight of each arc corresponds to the number of times that one item precedes the next-item.

The transformation of a database into a network is identical in the Markov Chain or Petri nets approaches. As the patterns that occur in nature, the accumulation of elements produces a specific shape. For example, a dune, created by wind is composed by several layers of sand and has a recognized configuration. The advantage is getting a macro view of the system, instead of extracting rules from a small subset of data, as sequence/episode mining algorithms do.

The Poly-tree Sequence approach has strong connections with the Markov Chain Models. In the Markov Chain Model each state is an item and in the transition matrix, each P(i,j) is the probability of moving from state *i* to state *j* in one step. This approach, like the Markov Chain Models, also presents a global view since all the items are taken into consideration. In our approach instead of the relative frequencies, the absolute frequencies are used. Another difference between this approach, using poly-trees and the Markov Chain is that it can handle cyclic graphs while the Markov Chain deals only with acyclic ones. As has already been mentioned, we use heuristics based on the Prim algorithm in order to reach a good scalability.

3.2 Searching Heuristics

In this paper, two heuristics are shown: The Forward Heuristic, that generates a tree of items, and the Back-and-Forward Heuristic that is able to create poly-trees.

In our approach, we can find an identical tree by using the Forward Heuristic, Algorithm 2, based on the Prim algorithm.

Algorithm 2. Forward Heuristic

```
Input: Network G;
Output: Tree S;
Initialize S;
For each vertex in G
   For each edge in G
        x = arg_max(weighted forward-vertex not visited in G and connected with S)
   End-for;
   Update solution S with x;
End-for;
```

For the Back-and-Forward mode, Algorithm 3, we developed the following heuristic also based on the Prim algorithm.

Algorithm 3. Back-and-Forward Heuristic

```
Input: Network G;
Output: Poly-tree S;
Initialize S;
For each vertex in G
   For each edge in G
        x = arg_max(weighted forward-vertex not visited in G and connected with S;
                    weighted back-vertex not visited in G and connected with S)
   End-for;
   Update solution S with x;
End-for;
```

Prim algorithm finds the minimum or maximum spanning-tree in undirected graphs. Back-and-Forward heuristic finds the Maximum Weighted Poly-tree by applying the same technique to directed graphs. The time complexity of the Back-and-Forward heuristic is equal to the Prim algorithm, which is $\Theta(E + V. \log V)$ with E edges and V vertexes.

3.3 Numeric Example

To exemplify the different approaches a numeric example is presented. We are going to use an event log created by [2] and presented in Table 1.

In Figure 2, a numeric example is shown. Figure 2.a shows the original cyclic network provided by the first phase of the algorithm. For the second phase two approaches are possible. Figure 2.b shows the maximum weighted tree that results from the application of the Forward heuristic. In Figure 2.c the maximum weighted poly-tree is shown. Note that in vertex 4 the in-degree is two and the edge (4, 5) was found using the maximum weighted back-vertex mode. Note also, that the sum of the weights is greater than the previous one, showing that the poly-tree structure always represents more weighted patterns.

Table 1. Event Log order by frequency

event stream	*#count*		*event stream*	*#count*
acdeh	455		acdefbdeg	14
abdeg	191		acdefdbeg	11
adceh	177		adcefcdeh	9
abdeh	144		adcefdbeh	8
acdeg	111		adcefbdeg	5
adceg	82		acdefbdefdbeg	3
adbeh	56		adcefdbeg	2
acdefdbeh	47		adcefbdefbdeg	2
adbeg	38		adcefdbefbdeh	1
acdefbdeh	33		adbefbdefdbeg	1
			adcefdbefcdefdbeg	1

Different applications can use the Forward or Back-and-Forward heuristic. The Forward heuristic must be chosen if a starting node is given. Node ‘a’ was chosen in the example. For instance, most of the web mining problems start in a root node, so we suggest this first mode [6]. If there is no information about the starting node, the best edge should be chosen, and the Back-and-Forward heuristic was applied in previous works [16] and [20].

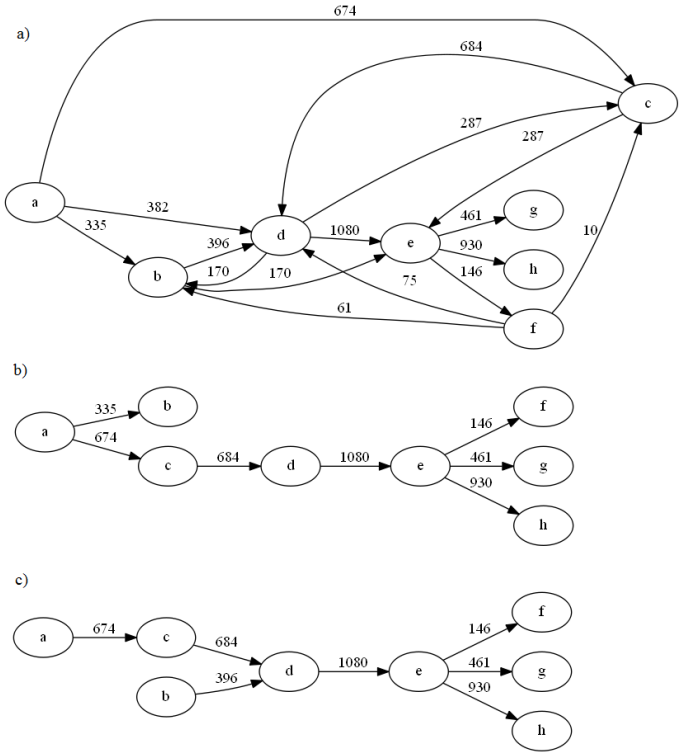

Fig. 2. (a) Original cyclic network, (b) Forward Heuristic provides a tree solution, (c) Back-and-forward Heuristic provides a poly-tree solution

4 Computational Results

In the validation of an algorithm some choices must be made such as the computational environment, the datasets and the performance measures. The performance measures discussed are the computational time and the quality of the solutions.

4.1 Computational Time

The computer programs were written in C language and the Dev-C++ compiler was used. The computational results were obtained using a 2.53GHz Intel Core-2Duo processor with 4.00 GB of main memory running under the Windows Vista operating system.

The artificial datasets were generated by the IBM Quest Synthetic [4] [13]. Table 2 shows the subset of the parameters used in the generator.

Table 2. Parameters for IBM Data Generator

Symbol	*Meaning*
T	average number items per stream
N	number of different items in thousands

We tested the datasets by varying the parameters of the IBM data generator as shown in Table 3. The dataset T?N1 shows how the algorithms perform by varying average number of items per stream and the dataset T10N? is used to study the variation of numbers of different items in thousands.

Table 3. Varying parameters of the datasets

T?N1	*average number of items per stream*	*CPU Time (second)*
t2n1	2	13,3
t4n1	4	13,1
t6n1	6	13,2
t8n1	8	13,1
t10n1	10	13,3
T10N?	*number of different items in thousands*	*CPU Time (second)*
t10n1	1	13,3
t10n2	2	15,9
t10n4	4	16,3
t10n6	6	16,7
t10n8	8	16,6

Our algorithm performed very well in all the experiments with a running time around 15 seconds, showing an excellent scalability. The robustness of the Back-and-Forward Heuristic is based on the use of condensed data and takes advantage of polynomial algorithms that are able to find poly-trees in networks.

4.2 Data Visualization

The output of Ramex uses DOT language in order to be visualized by [12] and [21].

In Figure 3 a poly-tree with 7472 nodes and 7471 edges is provided from the t10n8 dataset. Note that there is a subset of inner nodes in the graph highlighted by the chars ‘a’, ‘b’, ‘c’ and ‘d’. The fork “abcd” is the origin of the radial poly-tree.

Given the holistic view of the dataset, Tulip software allows the user to zoom in on the radial poly-tree and discover micro-sequences.

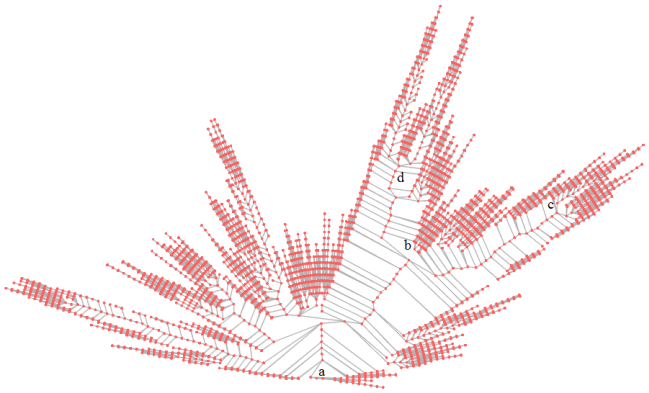

Fig. 3. Poly-tree with 7472 nodes and 7471 edges of the t10n8 dataset

5 Conclusions

In this paper we presented a new Sequence Mining algorithm called Ramex in order to identify branches in poly-trees. The algorithm has two phases: the transformation of the problem into a network, followed by the search of sequences.

The poly-tree model uses condensed data structures. The first phase of the algorithm shrinks the data without losing information. In the second phase, two heuristics were presented to find trees and poly-trees, the Forward Heuristic and the Back-and-Forward Heuristic both inspired in the Prim algorithm. The simplicity in the visualization is obtained using the poly-tree structure that shows all the events and also all of the most important event paths, without overlays, by-passes or cycles.

We believe Ramex is very useful in Pervasive Information Systems. Firstly, it provides a holistic view of the system, by retrieving the x-ray of the input data, with visualization benefits for the end-user. The accumulation of data produces stable results and avoids the *concept drift* as is common in micro-patterns approaches. Secondly, the algorithm performance enables its implementation in Big Data environments, with high velocity in the updates. To deal with data stream the online implementation of Ramex allows the accumulation of data in the network, avoiding the storage of the large event logs. In the meantime, the tree generation could run over an instance of the network, according to the specifications of the user.

References

1. van der Aalst, W.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011) ISBN 978-3-642-19344-6
2. van der Aalst, W.: Mine Your Own Business: Using Process Mining to Turn Big Data into Real Value. In: Keynote 21st European Conference on Information Systems, ECIS, Utrecht, The Netherlands (2013)

3. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
4. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the 11th International Conference Data Engineering, ICDE, pp. 3–14. IEEE Press (1995)
5. Borges, J., Levene, M.: Evaluating Variable-Length Markov Chain Models for Analysis of User Web Navigation Sessions. IEEE Trans. Knowl. Data Eng. 19(4), 441–452 (2007)
6. Cavique, L.: A Network Algorithm to Discover Sequential Patterns. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) EPIA 2007. LNCS (LNAI), vol. 4874, pp. 406–414. Springer, Heidelberg (2007)
7. Cavique, L.: A new taxonomy in Data Science. Maximus Report, section IV, pp. 92–93 (2014) (in Portuguese)
8. Cavique, L., Coelho, J.: Sequential Pattern Discovery Using Oriented Trees. Revista de Ciências da Computação (3), 12–22 (2008) (in Portuguese)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press and McGraw-Hill (2009) ISBN 0-262-03384-4
10. Edmonds, J.: Optimum branchings. J. Research of the National Bureau of Standards 71B, 233–240 (1967)
11. Fulkerson, D.R.: Packing rooted directed cuts in a weighted directed graph. Mathematical Programming 6, 1–13 (1974)
12. GraphViz (2014), http://www.graphviz.org/ (accessed June 9, 2014)
13. IBM Almaden Research Center, Synthetic data generation code for associations and sequential patterns (2006), http://www.almaden.ibm.com/software/quest/
14. Knuth, D.E., Morris, J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM Journal on Computing 6(1), 323–350 (1977)
15. Mannila, H., Toivonen, H., Verkamo, I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–289 (1997)
16. Marques, N.C., Cavique, L.: Sequential pattern mining of price interactions. In: EPIA 2013, 16th Portuguese Conference, Advances in Artificial Intelligence, Local Proceedings, Angra do Heroísmo, Açores, Portugal, pp. 314–325 (2013)
17. McKendrick, J.: Pervasive Business Intelligence means BI for the masses. Informatica (2008), http://blogs.informatica.com/perspectives/2008/08/31/pervasive-business-intelligence-means-bi-for-the-masses/#fbid=k4I_3ZvQSUp (accessed December 15, 2014)
18. Rebane, G., Pearl, J.: The recovery of causal poly-trees from statistical data. In: Proceedings of Uncertainty in Artificial Intelligence, pp. 222–228 (1987)
19. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
20. Tiple, P.S.: Tool for Discovering Sequential Patterns in Financial Markets. Dissertação para obtenção do Grau de Mestre em Engenharia Informática, na Faculdade de Ciências e Tecnologia da Universidade Nova Lisboa (2014)
21. Tulip, Better Visualization Through Research (2014), http://tulip.labri.fr/TulipDrupal (accessed December 2014)
22. Wang, W., Yang, J., Yu, P.: Meta-Patterns: revealing hidden periodic patterns. In: IEEE International Conference on Data Mining (ICDM), pp. 550–557 (2001)
23. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 42, 31–60 (2001)