A TIME & MEMORY EFFICIENT METHOD FOR EXTRACTING SEQUENTIAL PATTERNS AFTER DATA REDUCTION FROM ORIGINAL DATA SET

[1]Mr. Vishal Arun Pawar, [2]Dr. Bhupesh Gour, [3]Mr. Kamlesh Chandravanshi

[1]Research Scholar, [2]Professor, [3]Associate Professor
TIT, Bhopal

Abstract: **Tremendous amount of data being collected is increasing speedily by computerized applications round the globe. Concealed in the vast data, the valuable information is attracting researchers of multiple disciplines to study effective approaches to derive useful knowledge from within. Amid different data mining equitable, the mining of frequent patterns has been the focus of knowledge discovery in databases. This paper tends to find the efficient algorithm for mining sequential patterns. Mining sequential patterns with time constraints, like time gaps and sliding time-window, may reinforce the accuracy of mining findings. However, the competence to extract the time-constrained patterns was previously available only within Apriori framework. Modern studies show that pattern-growth methodology could speed up sequence mining. Current algorithms use a generate-candidate-and-test approach that may generate a large amount of candidates for phlegmatic datasets. Many candidates don't resemble in the database. Therefore we are introducing a more efficient algorithm for sequential pattern mining. The space & time consumption of proposed algorithm will be lesser in comparison to previous algorithm.**

I. Introduction

Data mining, which is also referred to as knowledge discovery in databases, has been recognized as the process of extracting non-trivial, implicit, previously unknown, and potentially useful information from data in databases . The database used in the mining process generally contains large amounts of data collected by computerized applications. For example, bar-code readers in retail stores, digital sensors in scientific experiments, and other automation tools in engineering often generate tremendous data into databases in a very fast speed. Not to mention the natively computing-centric environments like Web access logs in Internet applications. These databases thus serve as rich and reliable sources for knowledge generation and verification. Meanwhile, the large databases present challenges for effective approaches for knowledge discovery.

The discovered knowledge can be used in many ways in corresponding applications. For example, identifying the frequently appeared sets of items in a retail database can be used to improve the decision making of merchandise placement or sales promotion. Discovering patterns of customer browsing and purchasing (from either customer records or Web traversals) may assist the modeling of user behaviors for customer retention or personalized services. Given the desired databases, whether relational, transactional, spatial, temporal, or multimedia ones, we may obtain useful information after the knowledge discovery process if appropriate mining techniques are used.

A typical process of knowledge discovery in databases is illustrated in Fig. 1-1.

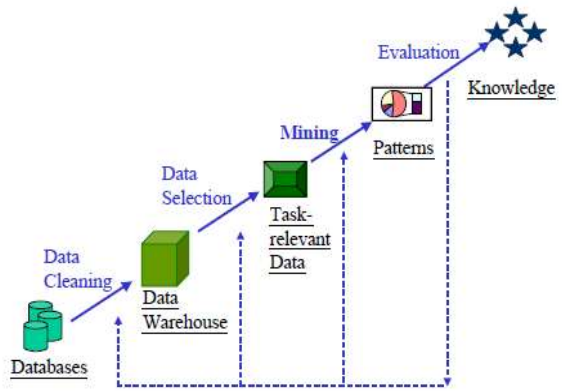

Figure: The process of knowledge discovery in databases

Having the databases, relevant prior knowledge, and the goals of the application domain, the target data set is created by selecting the data required. The data cleaning in Fig. 1-1 may removes those 'dirty' data, e.g. data with incomplete fields, missing or wrong values, in the preprocessing stage. The 'clean' data is then reduced and/or transformed so that the data is represented by the useful features and actionable dimensions. To find the patterns of interest, the users perform the required mining functions, which include summarization/generalization of data characteristics, classification/clustering of data for future prediction, association finding for data correlation, trend and evolution analysis, etc. The discovered patterns are evaluated and presented as knowledge. The process may iterate and contain certain loops between any two steps.

II. Background

Frequent itemsets[1,2] and association rules focus on transactions and the items that appear there. Databases of transactions usually have a temporal information. Sequential pattern or sequential rules exploit this temporal information.
Example data:

- Market basket transactions
- Web server logs
- Tweets
- Workflow production logs

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7

Table: A Sequence Data Base

III. Formal Definition of a Sequence

A sequence is an ordered list of elements (transactions). Each element contains a collection of events (items). Each element is attributed to a specific time or location. Length of a sequence, |s|, is given by the number of elements of the sequence
A **sequential rule is an implication of the form** 1=>2 .It has following two associated terms:

- **Support (1=>2) = F(1 => 2)/(N)**

Where F(1=>2) is the number of transactions in which 2 comes after 1. N is the total number of transactions.

- **Confidence (1=> 2) = F(1 => 2)/ F(1)**

Where F(1=>2) is the number of transactions in which 2 comes after 1. F(1) is the number of transactions containing 1.
Sequential Rule Mining finds all rules whose support and confidence is greater than the minimum support threshold and minimum confidence threshold respectively

IV. Literature Survey

Sequential rule mining has been applied in several domains such as stock market analysis (Das et al., 1998; Hsieh et al., 2006), weather observation (Hamilton & Karimi, 2005) and drought management (Harms et al, 2002; Deogun & Jiang, 2005).
The most famous approach for sequential rule mining is that of Mannila et al. (1997) and other researchers afterward that aim at discovering partially ordered sets of events appearing frequently within a time window in a sequence of events. Given these "frequent episodes", a trivial algorithm can derive sequential rules respecting a minimal confidence and support (Mannila et al., 1997). These rules are of the form X⇒Y, where X and Y are two sets of events, and are interpreted as "if event(s) X appears, event(s) Y are likely to occur with a given confidence afterward". However, their work can only discover rules in a single sequence of events. Other works that extract sequential rules from a single sequence of events are the algorithms of Hamilton & Karimi (2005), Hsieh et al. (2006) and Deogun & Jiang (2005), which respectively discover rules between several events and a single event, between two events, and between several event
Contrarily to these works that discover rules in a single sequence of events, a few works have been designed for mining sequential rules in several sequences (Das et al., 1998; Harms et al., 2002). For example, Das et al. (1998) discovers rules where the left part of a rule can have multiple events, yet the right part still has to contain a single event. This is a serious limitation, as in real-life applications, sequential relationships can involve several events. Moreover, the algorithm of Das et al. (1998) is highly inefficient as it tests all possible rules, without any strategy for pruning the search space. To our knowledge, only the algorithm of Harms et al. (2002) discovers sequential rules from sequence databases, and does not restrict the number of events contained in each rule. It searches for rules with a confidence and a support higher or equal to user-specified thresholds. The support of a rule is here defined as the number of times that the right part occurs after the left part within user-defined time windows
However, one important limitation of the algorithms of Das et al., (1998) and Harms et al. (2002) comes from the fact that they are designed for mining rules occurring frequently in sequences. As a consequence, these algorithms are inadequate for discovering rules common to many sequences. We illustrate this with an example. Consider a sequence database where each sequence corresponds to a customer, and each event represents the items bought during a particular day. Suppose that one wishes to mine sequential rules that are common to many customers. The algorithms of Das et al. (1998) and Harms et al. (2002) are inappropriate since a rule that appears many times in the same sequence could have a high support even if it does not appear in any other sequences. A second example is the application domain of this paper. We have built an intelligent tutoring agent that records a sequence of events for each of its executions. We wish that the tutoring agent discovers sequential rules between events, common to several of its executions, so that the agent can thereafter use the rules for prediction during its following execution.
In general, we may categorize the mining approaches into the generate-and-test framework and the pattern-growth one, for sequence databases of horizontal layout. Typifying the former approaches [1,2 , 3], the GSP (**G**eneralized **S**equential **P**attern)

algorithm [3] generates potential patterns (called candidates), scans each data sequence in the database to compute the frequencies of candidates (called supports), and then identifies candidates having enough supports as sequential patterns. The sequential patterns in current database pass become seeds for generating candidates in the next pass. This generate-and-test process is repeated until no more new candidates are generated. When candidates cannot fit in memory in a batch, GSP re-scans the database to test the remaining candidates that have not been loaded into memory. Consequently, GSP scans at least k times of the on-disk database if the maximum size of the discovered patterns is k, which incurs high cost of disk reading. Despite that GSP was good at candidate pruning, the number of candidates is still very huge that might impair the mining efficiency.

The PrefixSpan (Prefix-projected Sequential pattern mining) algorithm [4], representing the pattern-growth methodology [5, 4, 6], finds the frequent items after scanning the sequence database once. The database is then projected, according to the frequent items, into several smaller databases. Finally, the complete set of sequential patterns is found by recursively growing subsequence fragments in each projected database. Two optimizations for minimizing disk projections were described in [4]. The bi-level projection technique, dealing with huge databases, scans each data sequence twice in the (projected) database so that fewer and smaller projected databases are generated. The pseudo-projection technique, avoiding physical projections, maintains the sequence-postfix of each data sequence in a projection by a pointer-offset pair. However, according to [4], maximum mining performance can be achieved only when the database size is reduced to the size accommodable by the main memory by employing pseudo-projection after using bi-level optimization. Although PrefixSpan successfully discovered patterns employing the divide-and-conquer strategy, the cost of disk I/O might be high due to the creation and processing of the projected sub-databases.

Besides the horizontal layout, the sequence database can be transformed into a vertical format consisting of items' id-lists [7, 8, 9]. The id-list of an item is a list of (sequence-id, timestamp) pairs indicating the occurring timestamps of the item in that sequence. Searching in the lattice formed by id-list intersections, the SPADE (Sequential PAttern Discovery using Equivalence classes) algorithm [9] completed the mining in three passes of database scanning. Nevertheless, additional computation time is required to transform a database of horizontal layout to vertical format, which also requires additional storage space several times larger than that of the original
sequence database.

With rapid cost down and the evidence of the increase in installed memory size, many small or medium sized databases will fit into the main memory. For example, a platform with 256MB memory may hold a database with one million sequences of total size 189MB. Pattern mining performed directly in memory now becomes possible. However, current approaches discover the patterns either through multiple scans of the database or by iterative database projections, thereby requiring abundant disk operations. The mining efficiency could be improved if the excessive disk I/O is reduced by enhancing memory utilization in the discovering process.

In order to reduce the number of iterations, an efficient bidirectional sequential pattern mining approach namely Recursive Prefix Suffix Pattern detection, RPSP [12] algorithm is proposed. The RPSP algorithm first finds all Frequent Itemsets (FI‟s) according to the given minimum support and transforms the database such that each transaction is replaced by all the FI‟s it contains and then finds the patterns. The pattern further is detected based on ith projected databases, and constructs suffix and prefix databases based on the apriori property. RPSP will increase the number of frequent patterns by reducing the minimum support and vice versa. Recursion is terminated when the detected FI set of prefix or suffix projected database of parent database is null. All the patterns that correspond to a particular ith projected database of transformed database are formed into a set, which is disjoint from all other sets. The union of all the disjoint subsets is the resultant set of frequent patterns. The proposed algorithm was tested on the hypothetical data and results obtained were found satisfactory. Thus, RPSP algorithm can be applicable to many real world sequential data sets.

V. Conclusion

We have performed a systematic study on mining of sequential patterns in large databases and developed a pattern-growth approach for efficient and scalable mining of sequential patterns. It is found that the most of the methods like a priori-like methods candidate generation-and-test approach.

References:

[1] Agrawal, R., Imielminski, T., and Swami, A. Mining Association Rules Between Sets of Items in Large Databases, In Proc. SIGMOD Conference, (Washington D.C., USA, May 26-28, 1993) 207-216.

[2] Mannila, H., Toivonen and H., Verkano, A.I. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1, 1 (1997), 259-289

[3] Das., G., Lin, K.-I., Mannila, H., Renganathan, G., and Smyth, P. Rule Discovery from Time Series. In Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (New York, USA, August 27-31, 1998), 16-22.

[4] Harms, S. K., Deogun, J. and Tadesse, T. 2002. Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences. In Proc. 13th Int. Symp. on Methodologies for Intelligent Systems (Lyon, France, June 27-29, 2002), pp. .373-376.

[5] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," Proceedings of the 5th International Conference on Extending Database Technology, Avignon, France, pp. 3-17, 1996. (An extended version is the IBM Research Report RJ 9994)

[6] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal and M.-C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-projected Pattern Growth," Proceedings of 2001 International Conference on Data Engineering, pp. 215-224, 2001.

[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal and M.-C. Hsu, "FreeSpan: Frequent Pattern-projected Sequential Pattern Mining," Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 355-359, 2000.
[8] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, "Multi-Dimensional Sequential Pattern Mining," Proceedings of the 10th International Conference on Information and Knowledge Management, pp. 81-88, 2001.
[9] J. Ayres, J. E. Gehrke, T. Yiu, and J. Flannick, "Sequential PAttern Mining Using Bitmaps," Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, July 2002.
[10] S. Parthasarathy, M. J. Zaki, M. Ogihara, and S. Dwarkadas, "Incremental and Interactive Sequence Mining," Proceedings of the 8th International Conference on Information and Knowledge Management, Kansas, Missouri, USA, pp. 251-258, Nov. 1999.
[11] M. J. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences," Machine Learning Journal, Vol. 42, No. 1/2, pp. 31-60, 2001.
[12] Dr P padmaja, P Naga Jyoti, m Bhargava "Recursive Prefix Suffix Pattern Detection Approach for Mining Sequential Patterns" IJCA September 2011.