Research article Available online www.ijsrr.org ISSN: 2279–0543

International Journal of Scientific Research and Reviews

Knowledge Representation of Sequential Rule Mining by Using Pattern Growth Approach

Sachdev Neetu and Tapaswi Namrata*

Computer Science and Engineering, Institute of Engineering and Science, IPS Academy Indore, India. E-mail: Neetusachdev01@gamil.com

ABSTRACT

Sequential rule mining is an important data mining problem with multiple applications. Sequential rule mining has been applied in several domains such as stock market analysis, weather observation and drought management etc. It requires that the sequential rules mined must be unique. They should not have any redundancy .Because the redundancy will affect the prediction accuracy. A critical review of all the modern sequential rue mining technique is also performed. There are many techniques for the sequential rule mining. But still there is a lot of scope for the improvement in terms of efficiency and scalability. We present a modified technique for mining unique sequential rules from a data set. The experimental results have shown that this method does not produce duplicate results.

KEYWORDS: Data Mining, sequential rule mining, pattern growth approach, minimum support, confidence.

***Corresponding author**

Dr. Namrata Tapaswi

Professor & HOD, Computer Science and Engineering

Institute of Engineering and Science, IPS Academy

Indore, India

E-mail: hod.comps@ipsacademy.org

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. A sequential pattern having three items, the constitution of the pattern could be a list of: (1) three elements where each element is an item (2) two elements where the first element has one item and the second has two items (3) two elements where the first element has two items and the second has one item (4) one element that has three distinct items. Given the same number of possible items in the itemset database and the sequence database, the potential sequential patterns having three items greatly outnumber the potential itemsets having three items.

Sequential Pattern Mining is probably the most popular set of techniques for discovering temporal patterns in sequence databases. SPM finds subsequences that are common to more than *minsup* sequences. SPM is limited for making **predictions.** For example, consider the pattern {x}, {y}. It is possible that *y* appears frequently after an *x* but that there are also many cases where *x* is not followed by *y*. For **prediction**, we need a measurement of the confidence that if *x* occurs, *y* will occur afterward.

A **sequential rule** typically has the form X->Y .A sequential rule X⇒Y has **two properties**:

- **Support:** the number of sequences where X occurs before Y, divided by the number of sequences.
- **Confidence** the number of sequences where X occurs before Y, divided by the number of sequences where X occurs.

Sequential Rule Mining finds all **valid rules**, rules with a support and confidence not less than user-defined thresholds *minSup* and *minConf* .

For Example: An example of Sequential Rule Mining is as follows:

Consider *minSup*= 0.5 and *minConf*= 0.5.

Table1: A sequence database

S.No	ID	Sequences
01	Seq1	{a,b},{c},{f},{g},{e}
02	Seq2	{a,d},{c},{b},{a,b,e,f}
03	Seq3	{a},{b},{f},{e}
04	Seq4	{b} {f,g}

Table 1: Some rules found

ID	Rule	Support	Confidence
r1	{a,b,c} ⇒ {e}	0.5	1.0
r2	{a} ⇒ {c,e,f}	0.5	0.66
r3	{a,b} ⇒ {e,f}	0.5	1.0
r4	{b} ⇒ {e,f}	0.75	0.75
r5	{a} ⇒ {e,f}	0.75	1.0
r6	{c} ⇒ {f}	0.5	1.0
r7	{a} ⇒ {b}	0.5	0.66
… ..	…..	…..	…..

RELATED WORK

The sequential pattern is a sequence of itemsets that frequently occurred in a specific order, all items in the same itemset are supposed to have the same transaction time value or within a time-gap. Sequential patterns indicate the correlation between transactions while association rules represent intra-transaction relationships.

Sequential pattern mining was first introduced by [2]. It is the process of extracting certain sequential patterns whose support exceeds a predefined minimal support threshold. Since the number of sequences can be very large, and users have different interests and requirements, to get the most interesting sequential patterns usually a minimum support is predefined by the users. By using the minimum support we can prune out those sequential patterns of no interest, consequently making the mining process more efficient. Obviously a higher support of a sequential pattern is desired for more useful and interesting sequential patterns. However some sequential patterns that do not satisfy the support threshold are still interesting.

LITERATURE SURVEY

Sequential rule mining has been applied in several domains such as stock market analysis [4, 10], weather observation [7] and drought management [8, 5].

The most famous approach for sequential rule mining is that of [4] 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. 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 [7,10,5] which respectively discover rules between several events and a single event, between two events, and between several events.

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 [4, 8]. For example, [4] 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 [4] is highly inefficient as it tests all possible rules, without any strategy for pruning the search space. To our knowledge, only the algorithm of [8] 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 [4] and [8] 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 [4] and [8] 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 **S**equential **pa**ttern 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* (**S**equential **PA**ttern **D**iscovery using **E**quivalence 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.

CMRules**:** An association rule mining based algorithm for the discovery of sequential rules [15].

PROPOSED ALGORITHM

We will propose a novel algorithm for mining top ranked sequential rules. Unlike other algorithms, new algorithm will mine the unique sequential rules from a data set. He output will not contain redundant information.

Inputs:

- Sequential data set S
- K- number of rules to be generated
- Minimum confidence

Variables:

- N- used to store the top k sequential rules
- E- used to store the rules to be expanded right or left
- Minimum support

Initialization:

- set N = Null
- E = Null
- Minimum Support = 1

Pattern Growth Approach

TOPSEQRULES(S, *minsup*, *minconf*)

1. **Scan** the database S once. For each item *c* found, record the *sids* of the sequences that contains *c* in a variable *sids*(*c*).

2. For every pair of items X, Y. Where sids(X) >= minsup & sids(Y) >= minsup

{

Set sids(X => Y) = Ø. & set sids(Y =>X) = Ø.

3. For each sid s ∈ sids(X) ∩ sids (Y) {

If X occurs before Y in the sid s then sids(X => Y) = sids (X => Y) U {s}

If Y occurs before X in sid s then sids(Y => X) = sids (Y => X) U {s}

}

4. If |sids (X => Y) |/ |Total sequences| > = minsup Then {

Confidence (X => Y) = |sids(X=>Y) |/|sids(X) |

If confidence (X=>Y) >= minconf then

If |N| < K then

If there exist a rule r2 in N which is similar to the currently generated rule r1 & whose support is also similar to the support of r1 then the rule r1 is not added to N.

Otherwise this rule r1 is added to the N.

It is also added to E: E = E U r1.

If |N| >= K then

Remove a rule s from N whose support is equal to the present minimum support

Set minimum support = lowest support of rules in N.

}}

5. Repeat step 3 for pair Y & X

6. While (E is non empty)

{

Select a rule r with the highest support in E

Perform the left expansion

Perform right expansion

Remove r from E

}

CONCLUSION

In this paper, we presented a review of the algorithm for mining sequential rules. It is found that most of them are based on the generate-candidate-and-test approach. The sequential rule mining is a very popular and useful aspect of data mi ning. By using the sequential rule mining one can find the most frequent elements which occurred in a particular sequence. We have also presented a method to find sequential rules. It will generate exact top k rules from a sequential data base.

FUTURE WORK

With the mining capabilities of the proposed algorithms, there are several interesting extensions on frequent pattern mining, as listed below The discovery of sequential patterns with time constraints by memory indexing: It is worthy of study on extending the memory

indexing approach for efficient mining of generalized sequential patterns.

REFERENCES

1. Agrawal R , Imielminski T, Swami A . Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference. 1993; 207-216 .
2. Agrawal R , Srikant R . Mining Sequential Patterns. Proc. Int. Conf. on Data Engineering. 1995; 3-14.
3. Cheung D.W, Han J, Ng V, Wong Y. Maintenance of discovered association rules in large databases. An incremental updating technique. 1996; 106-114.
4. Das G , Lin K.I , Mannila H , Renganathan G , Smyth P . Rule Discovery from Time Series. Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining. 1998.
5. Deogun J S , Jiang L . Prediction Mining – An Approach to Mining Association Rules for Prediction. Proc. of RSFDGrC Conference. 2005; 98-108.
6. Faghihi U , Fournier-Viger P , Nkambou R , Poirier P . The Combination of a Causal Learning and an Emotional Learning Mechanism for Improved Cognitive Tutoring Agent. Proceedings of IEA-AIE 2010.
7. Kabanza F , Nkambou R , Belghith K . Path-planning for Autonomous Training on Robot Manipulators in Space. Proc. 19th Intern. Joint Conf. on Artificial Intelligence.2005; 35-38.
8. Hamilton H J , Karimi K . The TIMERS II Algorithm for the Discovery of Causality. Proc. 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining.2005; 744-750.
9. Harms S K , Deogun J , Tadesse T . Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences. Proc. 13th Int. Symp. on Methodologies for Intelligent Systems.2002;373-376.
10. Hegland M . The Apriori Algorithm – A Tutorial. Mathematics and Computation. Imaging Science and Information Processing.2007; 11:209-262.
11. Hsieh Y L , Yang D L , Wu J . Using Data Mining to Study Upstream and Downstream Causal Realtionship in Stock Market. Joint Conference on Information Sciences.2006.
12. Pei J , Han J . Mining Sequential Patterns by Pattern-Growth: ThePrefixSpan Approach. IEEE Trans. Knowledge and Data Eng. 2004; 1-17.
13. Laxman S , Sastry P . A survey of temporal data mining. Sadhana. 2006;3: 173-198.
14. Mannila H , Toivonen H , Verkano A.I .Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery. 1997;1(1): 259-289.

15. Fournier-Viger P , Faghihi U , Nkambou R , Mephu Nguifo E . CMRules: An Efficient Algorithm for Mining Sequential Rules Commonto Several Sequences. Knowledge Based Systems. 2012;63-76.