International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 12, ISSN No. 2455-2143, Pages 58-64
Published Online October-November 2016 in IJEAST (http://www.ijeast.com)

SURVEY ON SEQUENCE MINING ALGORITHMS

Dr. Shalini Bhaskar Bajaj
Department of Computer Science and Engineering
Amity University
Gurugram, Haryana

Deepika Garg
Research Scholar, Department of Computer Science and Engineering
Amity University
Gurugram, Haryana

Abstract— **Sequential pattern mining is used to find relevant patterns from data, where the values are delivered in a sequence. Sequential mining can be used in different areas like marketing strategy, medical treatment, stock marketing, crime detection, DNA sequencing and so on. This paper presents the analysis of common existing sequential pattern mining algorithms, which has not been reviewed earlier. The paper presents a study of sequential pattern-mining algorithms based on four major classes: apriori based algorithm, breadth first search based algorithms, depth first search based algorithms and pattern growth based algorithms. A comparative analysis has been done on the basis of important key features supported by various algorithms. This study gives an enhancement in the understanding of the approaches of sequential pattern mining.**

Keywords— **Sequence pattern mining, pattern growth, candidate pruning**

I. INTRODUCTION

Sequential pattern mining is an important research area in the field of data mining. It is an extension of the association rule mining [12]. Sequence mining has a wide range of real-life applications. Sequential mining algorithms solve the problem of discovering the presence of frequent sequences in sequential database [1]. The database consists of a set of sequences called as data sequences. Each data sequence is a list of customer transaction, and each transaction is a set of items. The transaction time is associated with each transaction in the sequence database. The sequential pattern mining is almost similar to the association rule mining, but the difference is that, in sequence mining the events are linked with time. The sequential pattern mining discovers the correlation between the different transactions, but in case of association rule mining it discovers the relationship of items in the same transaction. Association rule mining is used to discover different frequent items brought together by various customers whereas sequential pattern mining is used to discover the items that are brought in a particular order by a single customer. The sequential pattern mining is very useful in various domains such as marketing, stock marketing, crime detection, DNA sequencing.

Several algorithms for sequential pattern mining have been proposed and most algorithms are based on apriori property proposed by Agrawal and Srikant in 1994 [3]. The apriori property states that a frequent pattern contains sub-patterns that are in turn frequent. Based on this assumption, a succession of algorithms has been proposed. In 1995, the algorithms AprioriAll, AprioriSome, DynamicSome have been proposed by Agrawal and Srikant [1]. The apriori-based horizontal formatting method (GSP) has been presented in 1996 by Agrawal and Srikant [2]. In the year 2010, Gauda K., et al. proposed PRISM [11], Fournier P., et al. in 2011 proposed RuleGrowth [7], Steeps algorithm was proposed by Liu J. in 2012 [14]. In 2013, Tseng V., et al. [19] proposed an algorithm TNS. Fournier P., et al. proposed an algorithm ERMiner in 2014[6]. In 2014 Fournier P., et al. upgraded the algorithms SPADE, SPAM and named them as CM-SPADE, CM-SPAM respectively. Details on the above mentioned algorithms is given in the next section.

II. LITERATURE REVIEW

Various researchers have proposed algorithms for sequential pattern mining. Some of the proposed algorithms rebated on apriori and some are pattern growth based sequential pattern mining algorithms. The study and review of some of the algorithms proposed in the field of sequential pattern mining is presented in the subsequent paragraphs.

Agrawal R. et al. [1] presented three algorithms to solve the problem for generating sequential patterns and empirically evaluate their performance using synthetic data. Two of the proposed algorithms, AprioriSome and AprioriAll have comparable performance although AprioriSome performs a little better when the minimum number of customers that must support a sequential pattern is low. AprioriSome and AprioriAll have excellent scale-up properties with respect to the number of transactions per customer and the number of items in a transaction. Another algorithm, DynamicSome skips counting candidate sequences of certain lengths in the forward phase.

Zaki J. [21] proposed an algorithm named SPADE (full form). This technique divides the problems into sub-problems and each sub-problem can be solved independently in main memory using efficient lattice search technique. This reduces the I/O cost because it requires only three database scan to generate all patterns. It has linear scalability with respect to the number of input sequences and a number of other database parameters.

Ayres J. et al. [5] proposed SPAM (full form) algorithm which integrates a depth-first traversal of the search space with effective pruning mechanisms. The search strategy combines a vertical bitmap representation of the database with efficient support counting. A salient feature of this algorithm is that it incrementally outputs new frequent itemsets in an online fashion. This algorithm is especially efficient when the sequential patterns in the database are very long.

Wang J. et al. [20] proposed an algorithm named as BIDE (BI-Directional Extension), an efficient algorithm for mining frequent closed sequences without candidate maintenance. It prunes the search space using the BackScan pruning method and the Scan-Skip optimization technique. A thorough performance study with both sparse and dense real-life data sets has demonstrated that BIDE significantly outperforms the previous algorithms. It consumes order(s) of magnitude less memory and can be more than an order of magnitude faster. It is also linearly scalable in terms of database size.

Gouda K. et al. [11] proposed PRISM algorithm (PRIme encoding based Sequence Mining) for mining frequent sequences. It utilizes the vertical approach for enumeration and support counting. It uses the concept of prime block encoding which in turn is based on prime factorization theory. The algorithm uses the concept of GCD and LCM for intersection and union of data sets. PRISM algorithm was applied on both real and synthetic datasets and compared with other algorithm like SPADE, SPAM and Prefixspan. The comparison results shows the superiority of PRISM over other algorithm.

Fournier P., et al [9] proposed CMRULES, an algorithm for mining sequential rules common to many sequences in sequence databases. This algorithm uses the existing apriori algorithm to find association rules. Then it eliminates association rules that do not meet minimum confidence and support thresholds according to the time ordering. The performance of CMRULES is evaluated on a public dataset in terms of execution time. Results show that CMRULES is more efficient for low support thresholds, and has a better scalability.

Liu J. [14] proposed a new data storage structure called frequent sequence tree and discussed the construction algorithm for frequent sequence tree under the name Con_FST. The root node of the frequent sequence tree stores the frequent sequence tree support threshold and the path from the root node to any leaf node represents a sequential pattern in the database. Frequent sequence tree stores all the sequential patterns with its support that meet the frequent sequence tree support threshold. Therefore, when the support is changed the algorithm which uses frequent sequence tree as storage structure could find all the sequential patterns without mining the database.

Fournier P., et al. [7] proposed a novel algorithm for mining sequential rules common to several sequences named as RuleGrowth. Unlike other algorithms, RuleGrowth uses a pattern-growth approach for discovering sequential rules such that it can be much more efficient and scalable. RuleGrowth performs well under low support and confidence threshold and has much better scalability. A drawback of this approach is that it decreases the performance of database as it repeatedly performs a costly database projection operation.

Liu J. [15] gave the idea of structure of sequence tree based on projected database, called sequence tree, and proposed Steeps algorithm which is used to construct the sequence tree. Sequence tree is a data storage structure and is similar to the prefix tree. But, the sequence tree stores all the sequences in the original database. The path from the root node to any leaf node represents a sequence in the database. The structural characteristic of sequence tree makes it suitable for incremental sequential pattern mining. Steeps algorithm works well at low support thresholds.

Tseng V., et al. [19] considers together the problem of time-consuming to select the parameters to generate a desired amount of rules and redundancy in results by proposing an efficient algorithm named TNS for mining the top-k non-redundant sequential rules. TNS is based on depth first search and rely on an approximate approach that can guarantee exact result under certain conditions. Comparison of the performance of TNS with TopSeqRules on three real datasets shows that TNS has excellent performance and scalability. TNS provides the benefit of eliminating redundancy. This is the main improvement of TNS over TopSeqRules algorithm.

Fournier P., et al. [6] overcome the drawback of RuleGrowth by proposing an algorithm named ERMiner (Equivalence class-based sequential Rule Miner) for mining sequential rules. It uses the concept of searching using equivalence classes of rules having the same antecedent or consequent. Furthermore, it includes a data structure named SCM (Sparse Count Matrix) to prune the search space. ERMiner is up to five times faster than RuleGrowth and consumes less memory.

Fournier P., et al. [10] gives a new structure named as CMAP (Co-occurrence MAP) for storing co-occurrence information. They upgrade the previous algorithms SPAM, SPADE, GSP by applying on CMAP data structure and the new algorithms are renamed as CMSPADE, CMSPAM, and CM-ClaSP. The algorithms are applied on six real-life data set and results shows that CM-SPADE and CM-ClaSP have better performance for mining sequential patterns and closed sequential patterns.

III.COMPARISON OF DIFFERENT EXISTING SEQUENTIAL PATTERN MINING ALGORITHMS

The sequence mining algorithm has been classified into the following classes: Apriori-like algorithms, BFS (Breadth First Search)-based algorithms, DFS (Depth First Search)-based algorithms, and pattern growth based algorithms.

A. Apriori-like algorithms

CMRULE: Algorithm [9] is specifically designed for mining sequential rules common to many sequences. It does not restrict the number of events in each rule. The algorithm does not rely on a sliding-window approach. Instead, it finds associations rules between items to prune the search space to items that occur jointly in many sequences. Finally, the association rules that do not meet minimum confidence and support thresholds according to the time ordering are eliminated.

Algorithm

- Consider the sequence database as transactional database.
- Find all association rules from the transaction database by applying an association rule mining algorithm such as Apriori
- Select minsup = minSeqSup and minconf = minSeqConf.
- Scan the original sequence database to calculate the sequential support and sequential confidence
- Eliminate each rule r such that whose support< minsupport
- Return the set of rules

Advantages

- It is easy to implement.
- more efficient for low support thresholds, and has a better scalability

Disadvantages

- Its performance decreases as the number of rules increases.
- It becomes inefficient when the dataset is large.

B. BFS-based algorithms

A number of algorithms were developed in the past using the principles of BFS algorithm. Some of them are listed below:

GSP: Generalized Sequential Patterns (GSP) [13] uses the concept of BFS algorithm. This algorithm uses the downward-closure property of sequential patterns and follows a multiple pass candidate generate-and-test approach. The GSP algorithm [13] doesn't require finding all the frequent itemsets. This algorithm allows placing bounds on the time separation between adjacent elements in a pattern. GSP is designed for discovering generalized sequential patterns. The GSP algorithm makes multiple passes over sequence database. In the first pass, it finds the frequent sequences that have the minimum support. At each pass, every data sequence is examined in order to update the occurrence number of the candidates contained in this sequence.

Algorithm

- Generate frequent 1-sequence after database scan.
- Do while F(k)!= Null;
- Generate candidate sets Ck+1
- If Ck+1 is not empty, find the set of length-(k+1) sequential patterns
- Return the set of rule.

Advantages

- GSP is much faster than the AprioriAll algorithm.
- GSP scales linearly with the number of data-sequences, and has very good scale-up properties with respect to the average data sequence size.

Disadvantages

- It requires multiple scans of the database.
- Algorithm is inefficient for mining long sequential patterns

MFS: It is a modified version of GSP [21], with the aim to reduce the I/O cost needed by GSP. In first pass, it computes the rough estimate of all the frequent sequences set as a suggested frequent sequence set and to maintain the set of maximal frequent sequences known previously it uses the candidate generation function of GSP. The results show that MFS saves I/O cost significantly in comparison with GSP.

C. DFS-based algorithms

A number of algorithms were developed in the past using the principles of DFS algorithm. Some of them are listed below:

SPADE: Sequential PAttern Discovery using Equivalence classes (SPADE) [19][21] uses a vertical id-list database format, in which each sequence list is associated with objects in which it occurs, along with the time-stamps. The algorithm shows that all frequent sequences can be enumerated via simple temporal joins (or intersections) on id-lists. It uses a lattice-theoretic approach to decompose the original search space (lattice) into smaller pieces (sub-lattices) which can be processed independently in main-memory. It requires three database scans, or only a single scan with some pre-processed information, thus minimising the I/O costs. It uses two different search strategies for enumerating the frequent sequences within each sub-lattice: breadth-first and depth-first search.

Algorithm

- Given the vertical id-list database, compute all frequent 1-sequences.
- Compute the frequent 2-sequences.
- Frequent sequences are generated by joining the id-lists of all pairs of atoms and checking the cardinality of the resulting id-list against min sup.
- Process is repeated until all frequent sequences have been enumerated.
- Once all the frequent sequences for the next level have been generated, delete the current level sequences.
- Return the set of rule.

Advantages

- SPADE is about twice as fast as GSP.
- SPADE uses a more efficient support counting method based on the id-list structure.

SPAM: Sequential PAttern Mining (SPAM) [5] uses the concept of Depth First Search. SPAM assumes that the entire database completely fit into main memory. SPAM uses a vertical bitmap data layout which allows simple and efficient counting. The algorithm considers lexicographical ordering of the items in the database. Each sequence in the sequence tree can be considered as either a sequence-extended sequence or an item-set extended sequence. A sequence-extended sequence is a sequence generated by adding a new transaction consisting of a single item to the end of its parent's sequence in the tree. An item-set extended sequence is a sequence generated by adding an item to the last item-set in the parent's sequence.

Algorithm

- Initialize the variables for sequence and item-set extension.
- Generate the sequence-extension.
- Check for frequent sequence-extended.
- Call DFS pruning
- Generate item-set extension
- Check for frequent item-set extension.
- Call DFS pruning.
- Return the set of rule.

Advantages

- SPAM performs so well for large datasets.
- SPAM handles counting process efficiently.

Disadvantages

- SPAM is quite space-inefficient.

CMSPADE and **CMSPAM**: CMSPADE and CMSPAM algorithms [10] are the integrated version of SPADE and SPAM. The difference between SPADE and CMSPADE, SPAM and CMSPAM, is of data structure and pruning mechanism. SPADE and SPAM algorithms are applied on data structure CMAP and renamed as CMSPADE and CMSPAM respectively. Co-occurrence MAP (CMAP) is a structure mapping each item k ϵ I to a set of item succeeding it. It uses two CMAP named CMAPi and CMAPs to store itemset extension and sequence extension respectively. Pruning is done using two properties.

Property 1 (pruning an i-extension): Let A be a frequent sequential pattern and k be an item. If there exists an item j in the last itemset of A such that k belongs to cmi(j), then the i-extension of A with k is infrequent.

Property 2 (pruning an s-extension): Let A be a frequent sequential pattern and k be an item. If there exists an item j ∈ A such that the item k belongs to cms(j), then the s-extension of A with k is infrequent.

Advantages of CMSPADE and CMSPAM:

- Eight times faster than original SPADE, SPAM.
- Memory consumption is very low.
- Prune large amount of candidates.

D. Pattern growth based algorithms

RuleGrowth: RuleGrowth algorithm relies on a pattern-growth approach [7]. RuleGrowth first find rules between two items and then recursively grow them by scanning the database for single items that could expand their left or right parts (these processes are called left and right expansions). Like PrefixSpan, RuleGrowth also includes some ideas to prevent scanning the whole database every time. It takes as parameters a sequence database and the minsup and minconf thresholds. This procedure first generates all rules r of size 1*1 such that sup(r) ≥ minsup and then call two recursive procedures for growing each rule.

Algorithm

- Scan the database. Read the each sequence.
- rules ←Φ ;
- check first and last occurrence of each item
- check the support
- Expand left and right part of rule if support>=minsup.
- Check the confidence.
- Confidence>=minconf, the generated rule is valid

The algorithm is applied on three real databases having different characteristics and representing three real-life situations and result shows that the algorithm is much more efficient and scalable.

Advantages

- RuleGrowth does not utilises more memory space as it relies on the pattern-growth approach instead of a generate-candidate-and-test approach.
- More efficient and scalable than CMRule and CMDeo

Disadvantages

- It repeatedly performs a costly database projection operation, which decrease performance of datasets.

ERMiner: ERMiner (Equivalence class based sequential Rule Miner) [6] relies on vertical representation of the database to avoid performing database projection and the novel idea of exploring the search space of rules using equivalence classes of rules having the same antecedent or consequent. It includes a data structure named SCM (Sparse Count Matrix) to prune the search space. ERMiner first scans the database to build all equivalence classes for frequent rules of size 1 * 1. Then, it recursively performs left/right merges on the identified equivalence classes to generate the other equivalence classes.

The problem of duplicity of rule is solved by not allowing a left merge after a right merge, rather by allowing a right merge after a left merge.

Algorithm

- leftStore ← Φ;
- rules ←Φ;
- Scan SDB once to calculate EQ, the set of all equivalence classes of rules of size 1*1;
- Call left-search for find left equivalence classes
- Call right-search for find right equivalence classes
- Store equivalence class in left store.
- Return set of rules.

Advantages

- Five times faster than RuleGrowth algorithm.
- Does not generate duplicate rule.
- Does not require rescanning of database.

Disadvantages

- Memory consumption is high

TNS: TNS (Top-K Non-Redundant Sequential Rules) [19] is based on depth first search. This algorithm is an improvement over TopSeqRule algorithm. This algorithm is designed to removes the redundant rule and to reduce the time for selecting parameters. It is based on a recently proposed approach for generating sequential rules that is named "rule expansions" and adds strategies to avoid generating redundant rules.

Algorithm

- Consider a sequence database, an integer *k* and the *minconf* threshold. Set minsup=0
- First scans the database to identify single items that appear in at least *minsup* sequences.
- Recursively grow rule by adding items to its antecedent or consequent
- Expand the left and right part of the rule.
- Check minsup of rule
- Repeat the procedure.
- Return the set of rules.

Advantages

- Work better for low threshold value.
- TNS always guarantee that the k rules returned are non-redundant

BIDE: BIDE (BI-Directional Extension) [20], an efficient algorithm for mining frequent closed sequences without candidate maintenance. It mines efficiently the complete set of frequent closed sequences. It does not store historical frequent closed sequence (or candidate) for a new pattern's closure checking.

Algorithm

- First scans the database once to find the frequent 1-sequences.
- Builds pseudo projected database for each frequent 1-sequence
- Check frequent 1-sequence using *BackScan* pruning method
- Computes the number of *backward-extension-items*
- Compute the number of *forward-extension-items*
- If there is no *backward-extension-item* nor *forward-extension-item*, output S as a frequent closed sequence.
- Grow S with each locally frequent item in lexicographical ordering to get a new prefix and build the pseudo projected database for the new prefix
- Check if it can be pruned, if not, compute the number of *backward-extension-items* and call itself.

Advantages

- It outperforms when the support threshold is low
- Consumes much less memory and can be an order of magnitude faster than CloSpan
- It has linear scalability in terms of base size;
- Pruning method is very effective.

IV. CHARACTERISTICS OF SEQUENTIAL MINING ALGORITHM

This section discusses the different parameters on which the sequential mining algorithms are characterised.

Statical Database: A database consisting of 'information-based relationships', one that is rigorously structured to facilitate retrieval and update in terms of inherent relationships.

Database MultiScan: Scanning of database multiple times.

Generate & Test: The process of forming n-length candidate and frequent pattern is repeated until there is no more frequent pattern combination found.

Pattern Growth: It uses the concept of divide and conquer approach. It recursively partition the dataset based on frequent patterns generated and then it mines them for frequent patterns in each of the partition.

DFS Based Approach: is an algorithm for traversing or searching tree or graph data structures. It starts at the root and explores as far as possible along each branch before backtracking.

BFS Based Approach: is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root and explores the neighbor nodes first, before moving to the next level neighbors.

Top-Down Search: In top-down approach the subsets of sequential patterns can be mined by constructing the corresponding set of projected data bases and mining each repetition from top to bottom.

Bottom-up Search: is the piecing together of systems, thus making the original systems into sub-systems of the emergent system.

Database vertical Projection: use vertical data format.

V. CONCLUSION

For retrieving useful information from the large amount of data, sequence mining plays an important role. A number of sequence mining algorithms have been designed like GSP, SPADE, CMSPAM, etc. These algorithms have been classified into four major classes. The paper presents a comparative analysis of important sequence mining algorithms selected from the previously described algorithms based on some features. These algorithm can be applied in various applications like market analysis, mining education data, health record, web logs, stock marketing. For example, AprioriAll has been used to analyse the probability and intensity of the forest fire, PrefixSpan to detect the malware in expert systems. Table 1 summarizes the feature of different sequential mining algorithms.

Table - 1 Comparative study of important sequential pattern mining algorithms

	Apriori-Like		**BFS Based**	**DFS Based**					**Patterrn Growth**			
Algorithm characteristics	**Apriori All**	**CM-RULE**	**GSP**	**SPADE**	**SPAM**	**CM-SPADE**	**CP-SPAM**	**Prefix Span**	**BIDE**	**ER Miner**	**Rule Growth**	**TNS**
Statical database	**yes**	**yes**	**yes**	**yes**	**yes**	**Yes**	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**
DataBase MultiScan	**yes**	**yes**	**yes**								**yes**	
Genrate & Test	**yes**	**yes**	**yes**	**yes**								
Pattern Growth									**yes**		**yes**	**yes**
Candidate Sequence Pruning			**yes**	**yes**		**yes**	**yes**	**yes**	**yes**			
DFS based approach				**yes**	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**
BFS based approach	**yes**	**yes**	**yes**									
Top-down search						**yes**	**yes**	**yes**		**yes**	**yes**	**yes**
Bottom-up search		**yes**	**yes**	**yes**	**yes**							
Database vertical projection		**yes**		**yes**	**yes**	**yes**	**yes**		**yes**	**yes**	**yes**	**yes**

VI. REFERENCES

1. Agrawal, R. and Srikant, R.., "Mining Sequential Patterns", In Proceedings of the 11th International Conference on Data Engineering, Taipei, Taiwan, pp. 3-14, 1995.
2. Agrawal, R. and Srikant, R.., "Mining Quantitative Association Rules in Large Relational Table", In Proc. of the ACM-SIGMOD, Conference on Management of Data, Montreal, Canada, 1996.

3. Agrawal, R. and Srikant, R.., “Fast algorithms for mining association rules in large databases” In Proceedings of 20th International conference on Very Large Databases, 1994.
4. Agrawal, R. and Srikant, R.., “Mining Sequential Patterns: Generalizations and Performance Improvements”, In Proc. of the 5th International Conference on Extending Database Technology (EDBT), Avignon, France, 1996.
5. Ayres, J., Flannick, J., Gehrke , J. and Yiu T., “Sequential Pattern Mining Using a Bitmap Representation”, In Proceedings of Conference on Knowledge Discovery and Data Mining, pp. 429–435, 2002.
6. Founier, P., Zida, S., Guenieche, T. and Tseng V., “ERMiner: Sequential Rule Mining using Equivalence Classes”, Advanced in intelligent data Analysis, 13th International Symposium, pp . 108-119, 2014.
7. Founier, P., Nkambou, R. and Tseng V.,” RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth”, Symposium on Applied Computing, pp . 951-960, 2011.
8. Fournier, P. and Tseng, V. S.” Mining Top-K Sequential Rules”, In Proc. of the 7th International Confrence on Advanced Data Mining and Applications (ADMA 2011), Beijing, China, 2011.
9. Fournier, P., Faghihi, U., Nkambou, R. and Nguifo, E., “ CMRULES: An Efficient Algorithm for Mining Sequential Rules Common to Several Sequences”, Association for the Advancement of Artificial Intelligence, 2010.
10. Fournier, P., Gomariz, A., Campos, M. and Thomas, R., “Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information” PAKDD 2014, Part I, LNAI 8443, pp. 40–52, 2014.
11. Gouda, K., Hassaan, M. & Mohammed Zaki, J., “PRISM: An effectinve approach for frequent sequnce mining via prime block encoding” Journal of Computer and System science, pp. 88-102, 2010.
12. Han, J. and Kamber, M., “Data Mining: Concepts and Techniques”, Morgan Kaufman publishers, 2001.
13. Kao, B., Zhang, M. and Cheung, D., “A GSP-based efficient algorithm for mining frequent sequences”, In Proc. of IC-AI’2001, Las Vegas, Nevada, USA, 2001.
14. Liu, J. “The design of storage structure for sequence in incremental sequential patterns mining,” Networked Computing and Advanced Information Management (NCM), pp. 330 - 334, 2010.
15. Liu, J. “The design of frequent sequence tree in incremental mining of sequential patterns,” Software Engineering and Service Science (ICSESS), pp. 679- 682, 2012.
16. Liu,J., Pan, Y., Wang, K. and Han, J. “Mining frequent item sets by opportunistic projection” In ACM SIGKDD international conference on knowledge discovery in databases, 2002.
17. Masseglia, F., Poncelet, P. and Teisseire, M., "Incremental mining of sequential patterns in large databases", Data & Knowledge Engineering, Vol. 46, No.1, pp. 97-121, 2003.
18. Pei, J., Han, J. and Wang, W., "Constraint-based sequential pattern mining: the pattern-growth methods", Journal of Intelligent Information Systems, Vol:28, No: 2 ,pp:133-160, 2007.
19. Tseng, V. and Fournier, P., “TNS: Mining Top-K Non-Redundant Sequential Rules” Proc. 28th Symposium on Applied Computing (ACM SAC 2013). ACM Press, 2013.
20. Wang, J., Han, J., “BIDE: Efficient Mining of Frequent Closed Sequences” , In Proc. of 2004 Int. Conf. on Data Eng., Boston, pp 79–90, 2004.
21. Zaki, J., “SPADE: An Efficient Algorithm for Mining Frequent Sequences”, Kluwer Academic Publishers. Manufactured in The Netherlands, Machine Learning, pp. 42, 31–60, 2001.
22. Zaki, J., “Scalable algorithms for Association Mining”, IEEE Transactions on Knowledge and Data Engineering., pp. 372-390, 2000.