Overview of Sequential Pattern-Mining Algorithms

Prof. Kirti S. Patil.

Abstract- **The concept of Sequential Pattern Mining was first introduced by R. Agrawal and R. Srikant (1995). It aimed to retrieve frequent patterns in the sequences of products bought by customers through time ordered transactions. Sequential pattern mining is significant data-mining method for determining time-related behaviour in sequence databases. In this paper, a survey of the sequential pattern mining algorithms is performed. This paper represents a review on various algorithms of discovering sequential patterns from large sequence databases which is an important problem in the field of knowledge discovery and data mining.**

Keywords: **- sequential patterns, data mining, knowledge discovery, sequential pattern mining**.

I. INTRODUCTION

The Sequential pattern mining was first proposed by Agrawal and Srikant [1] and later many algorithms were proposed by many others for performance improvements. Initially the inventors refined their algorithm by introducing Generalized Sequential Patterns (GSP) based on the based on implementing apriori dimensions [2]. Few other interesting algorithms on persistent pattern mining are SPADE [3], SPAM [4], and PrefixSpan [5]. The SPADE uses lattice theoretic approach based on vertical ID list in order to optimize original search space into optimized smaller spaces. Later SPAM was introduces for mining long patterns this algorithm espouses a vertical bitmap representation and the performance scale shows that SPADE is more efficient than SPAM but still there is a flaw in this algorithm as it consumes more space compared with the SPAM algorithm.

II. APRIORI-BASED METHODS

A. AprioriAll

AprioriAll [1] scans the database several times to find frequent itemsets of size k at each k^{th} -iteration (starting from $k = 2$). It also has the generate-and-test feature by performing the *Apriori-generate* join procedure to join $L_k{-}1$ with itself to generate C_k, the set of candidate sequences inthe k^{th}-iteration, it then prunes sequences in $C_k$which have subsequences not in $L_k{-}1$ (i.e., are not large), creates $L_k$byadding all sequences from $C_k$with support ≥ *min sup* until there are no more candidate sequences. The pruning procedure performed on itemsets in *Ck*removes sequences that have infrequent subsequences. The AprioriAll technique suffers from increased delays in mining as the number of sequences in the database gets larger.

***Manuscript received Feb, 2014*.**
Kirti S. Patil, *IT Department, KCES'SCOEIT, Jalgaon . Jalgaon, India, Mobile No-9890444422*

B. PSP:

This is another apriori-based algorithm, also built around GSP, but the difference of using a prefix-tree. In PSP [6], user access sequences are sorted in the web log according to the IP address. The user is allowed to provide a time period t by which access sequences that are temporally close to each other are grouped. A prefix-tree is then built to handle the mining procedure in a way similar to GSP. Following up on tree traversal, graph traversal mining uses a simple unweighted graph to reflect the relationship between pages of websites. The algorithm is similar to apriori, without performing the Apriori-generate join. The database still has to be scanned several times, but it is more efficient than GSP.

C. GSP:

The GSP algorithm, adopt a multiple-pass candidate generate-and-test method for finding sequential patterns. The first pass on the database determines the support for each item in finding frequent 1-sequences based on the minimum support. The first ($k = 1$) scan over the table generates the set of candidate 1- sequences C_1 ,giving the seed set of frequent 1-sequences L_1 . These sequences provide a *seed set* $L_k$which is used to generate next-level candidate sequences in the next pass k+1. The next C_k+1 candidate set is generated by performing a GSP-join of $L_k$on itself. The GSP-join, like the apriori-generate join requires that two sequences in $L_k$join together if two conditions are met. Here, a sequence $s1$ in $L_k$joins another sequence $s2$ in L_k if the subsequence obtained by dropping the first ($k - 1$) items of $s1$ is the same as the subsequence obtained by dropping the last ($k{-}1$) items of $s2$, hence the candidate sequence generated is the sequence $s1$ extended with the last item in $s2$ and is added to C_k. The prune phase deletes candidate sequences that have a contiguous ($k - 1$)-subsequence with support less than *min sup*. Each candidate sequence has one more item than a seed k-sequence, so all candidate sequences have the same number of items at each level. Support for these candidate sequences is again found during a pass over the data. The algorithm terminates when no new sequential pattern is found in a pass, or no candidate sequence can be generated. The generate-and-test feature of candidate sequences in GSP consists of two phases: Join Phase and Prune Phase. During the join phase, candidate sequences are generated by joining L_k−1 with itself using *GSP-join*. For increased efficiency, GSP employs a hash-tree to reduce the number of candidates in C that are checked for sequences. GSP is reported to be 2 to 20 times faster than AprioriAll.

D. SPAM:

SPAM is provided in Ayers et al. [4] integrates the ideas of GSP, SPADE, and FreeSpan. The entire algorithm with its data structures fits in main memory, and is claimed to be the first strategy for mining sequential patterns to traverse the lexicographical sequence tree in depth-first fashion. Each node in the tree has sequence-extended children sequences generated in the *S-Step* of the algorithm, and itemset-extended children sequences generated by the *I-Step* of the algorithm at each node. SPAM traverses the sequence tree in depth-first search manner and checks the support of each sequence-extended or itemset-extended child against *min sup* recursively. If the support of a certain child *s* is less than *min sup*, there is no need to repeat depth-first search on *s* by the apriori property. Apriori-based pruning is also applied at each S-Step and I-Step of the algorithm, minimizing the number of children nodes and making sure that all nodes corresponding to frequent sequences are visited. For efficient support-counting, SPAM uses a vertical bitmap data structure representation of the database similar to the id list in SPADE. Each bitmap has a bit corresponding to each element of the sequences in the database. Each bitmap partition of a sequence to be extended in the S-Step is first transformed using a lookup table, such that all the bits after the index of the first “1” bit (call it index *y*) are set to one and all the bits with index less than or equal to *y* are set to zero. When SPAM was compared to SPADE, it was found to outperform SPADE by a factor of 2.5, while SPADE is 5 to 20 times more space-efficient than SPAM, making the choice between the two a matter of a space-time trade-off. Here, Ayres et al. [4] propose to use a compressed bitmap representation in SPAM to save space, but do not elaborate on the idea.

III. PATTERN GROWTH BASED METHODS

A. FreeSpan:

Frequent pattern projected Sequential pattern mining [7]uses frequent items to recursively project sequence databases into a set of smaller projected databases and grows subsequence fragments in each projected database. This process partitions both the data and the set of frequent patterns to be tested, and confines each test being conducted to the corresponding smaller projected database. FreeSpan first scans the database, collects the support for each item, and finds the set of frequent items. Frequent items are listed in support descending order (in the form of item :support). According to flist, the complete set of sequential patterns in S can be divided into 6 disjoint subsets: (1) the ones containing only item ‘a’, (2) the ones containing item ‘b’, but containing no items after ‘b’ in flist, (3) the ones containing item ‘c’, but no items after ‘c’, in flist, and so on, and finally, (6) ones containing item ‘f’. The subsets of sequential patterns can be mined by constructing projected databases. Infrequent items, such as ‘g’ in this example, are removed from construction of projected databases.

B. PrefixSpan:

Prefix-projected Sequential pattern mining [5] works similar to FreeSpan except that the partitioning is done using prefixes of sequences. Its general idea is to examine only the frequent prefix subsequence’s and project only their corresponding postfix subsequence’s into projected databases because any frequent subsequence can always be found by growing a frequent prefix.No candidate sequence needs to be generated by PrefixSpan. Projected databases keep shrinking. The major cost of PrefixSpan is the construction of projected databases. To further improve mining efficiency, two kinds of database projections are explored: level-by-level projection and bi-level projection. Moreover, a main-memory-based pseudo-projection (using pointers rather than physically copying postfix sequences) technique is developed for saving the cost of projection and speeding up processing when the projected (sub)-database and its associated pseudo-projection processing structure can fit in main memory. PrefixSpan mines complete set of patterns much faster than both GSP and FreeSpan.

*C. WAP-**mine**:*

It is Pattern-Growth Miner with Tree Projection. At the same time as FreeSpan and PrefixSpan in 2000/2001, another major contribution was made as a pattern growth and tree structure-mining technique, that is, is the WAP-mine algorithm [7] with its WAP-tree structure. Here the sequence database is scanned only twice to build the WAP-tree from frequent sequences along with their support; a “*header table*” is maintained to point at the first occurrence for each item in a frequent itemset, which is later tracked in a threaded way to mine the tree for frequent sequences, building on the suffix. The first scan of the database finds frequent 1- sequences and the second scan builds the WAP-tree with only frequent subsequences.The frequent subsequence of each original database sequence is created by removing infrequent 1-sequences. The tree starts with an empty root node and builds downward by inserting each frequent subsequence as a branch from root to leaf. During the construction of the tree, a header link table is also constructed, which contains frequent 1-sequences, each one with a link connecting to its first occurrence in the tree and threading its way through the branches to each subsequent occurrence of its node type. To mine the tree, WAP-mine algorithm starts with the least frequent item in the header link table and finds frequent items building on the suffix to get frequent *k*-sequences. The WAP-mine algorithm is reported to have better scalability than GSP and to outperform it by a margin. Although it scans the database only twice and can avoid the problem of generating explosive candidates as in apriori-based and candidate generate-and-test methods, WAP-mine suffers from a memory consumption problem, as it recursively reconstructs numerous intermediate WAP-trees during mining, and in particular, as the number of mined frequent patterns increases. This problem was solved by the PLWAP algorithm [3], which builds on the prefix using position-coded nodes.

D. FS-Miner:

FS-Miner is a tree projection pattern growth algorithm that resembles WAP-mine and supports incremental and interactive mining [8]. The significance of FS-Miner is that it starts mining immediately with 2-subsequences from the second (which is also the last scan) of the database (at $k = 2$). It is able to do so due to the compressed representation in the FS-tree, which utilizes a header table of edges rather than single nodes and items, compared to WAP-tree and PLWAP-tree. It is also considered a variation of a trie, as it stores support count in nodes as well as edges of the tree that represents 2-sequences and is required for the incremental mining process. The sequence database is scanned once to find counts for links and to insert them in the header table. The FS-tree is built during the second scan of the database in a manner similar to WAP-tree and PLWAP-tree, except that a sequence that contains infrequent links is split and each part is inserted in the tree separately. Pointer links from header table are built to connect occurrences of header table entries, as they are inserted into the FS-tree similar to WAP-tree linkage.

IV. CONCLUSION

This survey has provided an overview of recent work in the area of sequential pattern mining algorithm techniques. Sequential pattern mining is trying to find the relationships between occurrences of sequential events, to find if there exist any specific orders of the occurrences. We can find the sequential patterns of specific individual items; also we can find the sequential patterns cross different items. With over a decade of extensive research, therehave been hundreds of research publications and tremendous research, development and application activities in this domain.

REFERENCES

[1]. Agrawal, R. and Srikant, R. "Mining sequential patterns". *In Proceedings of 11th International Conference on Data Engineering (ICDE). Taipei, Taiwan*, pp.3-14. 1995

[2]. Srikant R. and Agrawal R., "Mining Sequential Patterns: Generalization and Performance Improvements." *In Proceeding 5th International Conference Extending Database Technology (EDBT),* Springer-Verlag, Avignon France, pp: 3–17. 1996

[3]. Zaki and Mohammed J. "SPADE: An efficient algorithm for mining frequent sequences." *Machine learning* 42, no. 1-2, 2001, pp: 31-60,ISSN:1573-0565.

[4]. Ayres, Jay, Jason Flannick, Johannes Gehrke, and TomiYiu. "Sequential pattern mining using a bitmap representation." In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 429-435. ACM, 2002.ISBN:1-58113-567-X

[5]. Han, Jiawei, Jian Pei, BehzadMortazavi-Asl, Helen Pinto, Qiming Chen, UmeshwarDayal, and M. C. Hsu. "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth." In *Proceedings of the 17th International Conference on Data Engineering*, pp. 215-224. 2001.ISSN :1063-6382,Print ISBN:0-7695-1001-9

[6]. Masseglia, Florent, Pascal Poncelet, and RosineCicchetti. "An efficient algorithm for web usage mining." *Networking and Information Systems Journal*2, no. 5/6, 2000, pp: 571-604.

[7]. Han J., Pei J., Mortazavi-Asl B., Wang J., Chen Q., Dayal U. and Hsu M. (2000) "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining." *In Proceedings 2000 ACM SIGKDD International Conference Knowledge Discovery in Databases*, pp: 355-359.ISBN:1-58113-233-6

[8]. El-Sayed, Maged, Carolina Ruiz, and Elke A. Rundensteiner. "FS-Miner: efficient and incremental mining of frequent sequence patterns in web logs." In *Proceedings of the 6th annual ACM international workshop on Web information and data management*, pp. 128-135. ACM, 2004.ISBN:1-58113-978-0

Kirti S. Patil completed the B.E. &M.E. degree in Computer Science & Engineering from North Maharashtra University Jalgaon (M.S.). I presently work as Asst. Prof. in department of IT at K.C.E.S's College of Engineering and Information Technology, Jalgaon (M.S.). I have presented 5 National papers & 2 International papers in Conferences. I also published a paper in International Journal.