ISSN:0975-9646

An Efficient Mining of Sequential Rules Using Vertical Data Format

Surbhi Jigneshkumar Sheth, Shailendra K Mishra

Computer and Science Department,
Parul Institute of Technology , Gujarat - India

***Abstract*— Data mining is the method of analyzing data from different perspectives and summarizing it into useful information. Among the various data mining tasks sequential pattern mining is one of the most important tasks. Sequential pattern mining consists of mining subsequences that appear frequently in a set of sequences. The vertical representation allows generating patterns using depth-first search to discover patterns and calculating their supports without performing costly database scans. CMAP can be used to prune candidates in vertical Algorithms like SPAM. Although there have been studies on the sequential patterns and CM-SPAM algorithm is not generating rules**

***Keywords*- sequential pattern mining; sequential rule mining; vertical database format; candidate pruning; Apriori.**

I. INTRODUCTION

A. Introduction of Sequential Pattern Mining

There are several major data mining techniques that have been developed and are used in the data mining projects which include association, classification, clustering, sequential patterns, and prediction and decision tree. From all different tasks in data mining, sequential pattern mining is one of the most important tasks. Sequential pattern mining involves the mining of the subsequences that appear frequently in a set of sequences. Sequential pattern mining algorithms using a vertical representation are the most efficient for mining sequential patterns in dense and have excellent performance. Finding statistically relevant patterns between data examples where the values are delivered in a sequence. Given a set of sequences, find the complete set of frequent subsequences. A sequence database consists of ordered elements or events in sequential pattern mining.

B. Apriori Based Algorithm

The Apriori family of algorithms has typically been used to discover intra-transaction associations and then to generate rules about the discovered associations [6]. Apriori based algorithms has mainly two format using candidate generation. Horizontal Database Format Algorithm.

1. Vertical Database Format Algorithm.
2.

C. Horizontal Database Format Algorithm

1) GSP—Generalized Sequential Pattern Mining

Initially, every item in DB is a candidate of length-1 for each level (i.e., sequences of length-k) do scan database to collect support count for each candidate sequence generate candidate length-(k+1) sequences from length-k frequent sequences using Apriori repeat until no frequent sequence or no candidate can be found Major strength: Candidate pruning by Apriori[8] Benefits from the Apriori pruning is that in apriori algorithm Reduces search space and Bottlenecks Scans the database multiple times Generates a huge set of candidate sequences

D. Vertical Database Format Algorithm.

1) Vertical Algorithms

Vertical Algorithms basically used in SPAM, SPADE, bitSPADE, ClaSP, VMSP. The database to convert it to a vertical representation (**sids lists**).

Table4 vertical format [3]

a	
SID	Itemsets
1	1
2	1,4
3	1
4	

b	
SID	Itemsets
1	1
2	3,4
3	2
4	1

c	
SID	Itemsets
1	2
2	2
3	
4	

d	
SID	Itemsets
1	
2	1
3	
4	

e	
SID	Itemsets
1	5
2	4
3	4
4	

f	
SID	Itemsets
1	3
2	4
3	3
4	2

g	
SID	Itemsets
1	3,4
2	
3	
4	2

Perform a depth-first search by joining items to each pattern by i-extension and s-extension. Calculate the support of a pattern by join operation of sid lists. Does not require to scan the database more than once.

Drawback: generate a huge amount of candidates.

E. SPADE

SPADE (Sequential PAttern Discovery using Equivalent Class) .A vertical format sequential pattern mining method. A sequence database is mapped to a large set of Item: <SID, EID> Sequential pattern mining is performed by growing the subsequences (patterns) one item at a time by Apriori candidate generation.

F. SPAM

Sequential pattern mining algorithm uses depth first search traversal of search space vertical representation of the database, which enables efficiency of support count. The candidates are stored in a lexicography lattice or tree in this the candidates to be extended in one of two ways :

1) Sequence Extension (S- step)
2) Itemset Extension(I-step)

Which guarantees all nodes are visited. However, In this if support for a sequence s < min_supp at a particular node than no more depth first is required to downward closure.

G. Clospan

A closed sequential pattern *s*: there exists no superpattern *s'* such that *s'* ⊃ *s*, and *s'* and *s* have the same support Motivation: reduces the number of (redundant) patterns but attains the same expressive power Using Backward Subpattern and Backward Superpattern pruning to prune redundant search space.

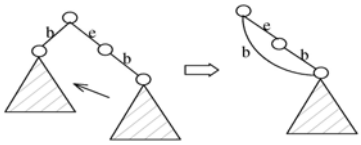

Fig1. Clospan[1]

II. RELATED WORK

The literature presents with a huge number of approaches for constraint-based sequential pattern mining and algorithms. In recent times, developing approaches for sequential mining of constraint-based sequential patterns has gained immense importance in real life applications. A concise review of some recent researches related to the sequential rule mining is presented here.

Carl H. Mooney[6] Sequential pattern mining has a technology to discover such subsequences. This form of discovery requires a different type of algorithm from those algorithms that are based on the more general transaction oriented datasets. vertical database format, where the rows of the database consist of object-timestamp pairs associated with an event. This makes it easy to generate id-lists for each event that consist of object-timestamp rows of events thus enabling all frequent sequences to be enumerated via simple temporal joins of the id-lists. The **SPADE** (Sequential Pattern Discovery using Equivalence classes) algorithm

(1) Breadth-first search: the lattice of equivalence classes is explored in a bottomup manner and all child classes at each level are processed before moving to the next.

(2) Depth-first search: all equivalence classes for each path are processed before moving to the next path.

Philippe Fournier-Viger1[3] Sequential pattern mining algorithms using a vertical representation are the most efficient for mining sequential patterns in dense and give excellent performance. Sequential pattern mining algorithms using the vertical format are very efficient because they can calculate the support of candidate patterns by avoiding costly database scans means no need to scan database again and again.

Integrate in three state-of-the-art vertical algorithms. Prune a large amount of candidates, in this it is faster than the corresponding original algorithms and SPAM gives good performance for mining sequential patterns in some databases. To perform the pruning in SPAM, we do not have to check for extensions of *pat* with *x* for all the items since such items, except for the last one, have already been checked for extension in previous steps, To perform the pruning in SPADE, we do not have to check if items of the prefix *P* are succeeded by the item $y \in Aj$. This is because the items of *P* are also in *Aj* . Therefore, checking the extension of *P* by *y* was already done, and it is not necessary to do it again. Note that to perform the pruning in SPADE, we do not have to check if items of the prefix *P* are succeeded by the item $y \in Aj$. This is because the items of *P* are also in *Aj*. Therefore, checking the extension of *P* by *y* was already done, so it is not necessary to do it again. In CM-SPAM that modified algorithm is not generating rules.

Philippe Fournier-Viger[11] In this paper, author present Rule Growth, a novel algorithm for mining sequential rules common to several sequences. Unlike other algorithms, Rule Growth uses a pattern-growth approach for discovering sequential rules such that it can be much more efficient at the same time also scalable. Here author present comparison of Rule Growth's performance with current algorithms for some of datasets. The experimental results show that RuleGrowth clearly outperforms current algorithms for all datasets under low support and confidence threshold and has a much better scalability. In this paper we also studies when we generates the rules at that time we need to generate rule's left and right both side and by this we generate redundant rules using Rule Growth algorithms. Here, we also count the minimum support and in first step we have to discard that items using minimum support, need to rule generated rules on right side and left side also. Another possible extension that would be easy to implement is to allow specifying constraint on items that should or should not appear in the left or the right part of a rule.

Algorithm of sequential patterns mining and sequential rules mining

A. The Rulegrowth Algorithm

Pattern-Growth approach is similar to the one which is used in the PrefixSpan algorithm for sequential pattern mining. 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 are also called left and right expansions. Like PrefixSpan, RuleGrowth need to scan whole database every time[11].

1) It takes as parameters a sequence database and the minsup and minconf thresholds.
2) first generates all rules r of size 1*1 such that sup(r) ≥ minsup and then call two recursive procedures for growing each rule.
3) The first step of the RULEGROWTH procedure is to scan the database one time to count the support of each item, then algorithm reads each sequence from the beginning to the end, and no for each item *c*, the sid (sequence id) of each sequence that contains *c* in a variable *sids_c*, and the first and last occurrence of the item for each of those *sids* in variables respectively named firstOccurences_c and lastOccurences_c.
4) By applying, all rules of size 1*1 are generated very efficiently without scanning the database again.

5) This algorithms takes each pair of items *i, j,* where *i* and *j* each have at least *minsup*|S|* sids (if this condition is not met, it means that no rule having at least the minimal support could be created with *i* and *j*).
6) The algorithm then initialize two variables *sidsi#j*={} and *sidsj#i*={} for counting the support of the rules {i}⇒{j} and {j}⇒{i}. Then, the algorithm loops over each sid containing *i* and *j* to check for each sid if the first occurrence of *i* occurs before the last occurrence of .
7) If it is the possible then, the current sid is added to *sidsi* into *j*.
8) Similarly, if first occurrence of *j* is before the last occurrence of *i*, the current sid is added to *sidsj#i*.
9) After this loop completed that time the support of the rule {i}⇒{j} is simply obtained by dividing *sidsi#j* by |S|.
10) If the support is higher or equal to *minsup*, the procedure EXPANDLEFT and EXPANDRIGHT are called to try to expand the rule's left and right parts and the confidence of the rule is calculated by dividing *sidsi*(Followed by or #)*j* by |*sids_i*|.
11) If the confidence is higher or equal to *minconf*, the algorithm output the rule. After this, the same process is repeated for the rule {j}⇒{i}.
12) The procedure EXPANDLEFT tries to expand the left side of a rule.
13) Its takes as parameters a rule I⇒J to be expanded (*ruleIJ*), the list of sids containing I (*sidsI)*, the list of sids containing (*sidsI#J)* and a structure (an hashmap in our implementation) indicating the last occurrence of J for each sid (*lastOccurences_J)*.
14) The procedure first tries to find items that could expand the rule. It does this by looping over each sequence from *sidsI#J*.
15) For each item *c* appearing before the last occurrence of J in a sequence, the sid of the sequence is added to a variable *sidsIc#J.*
16) The EXPANDLEFT procedure is called with the list of sids containing I∪{c} (*sidsIc*) as second parameter instead of *sidsI*. *sidsIc* is obtained by doing a simple loop over *sidsI* to check if each sid is an element of *sids_c*. After calling EXPANDLEFT, the procedure checks the confidence of I∪{c}⇒J. It is calculated by dividing |*sidsIc*■*J*| by |*sidsI* |. If the confidence is higher or equal to *minconf*, the procedure output the rule (the rule is valid)

B. SPAM algorithm

Sequential pattern mining algorithm uses depth first search traversal of search space vertical representation of the database, which enables efficiency of support count. The candidates are stored in a lexicography lattice or tree in this the candidates to be extended in one of two ways[1] :

1) Sequence Extension (S- step)
2) Itemset Extension(I-step)
3) Which guarantees all nodes are visited. However, In this if support for a sequence s < min_supp at a particular node than no more depth first is required to downward closure.
4) Steps of algorithm
5) Using candidate generation: Vertical data format
6) Spam first scan the input database SDB to construct vertical database V(SDB) and then set the frequent item F1.
7) For each item S belongs to F1.
8) Spam calls the search procedure with<s>,F1,{e belongs to F| e>lex s}, minsup.
9) The search procedure output the pattern <{s}> and recursively explore candidate patterns starting with prefix <{s}>.
10) Take pat parameter to generate candidate.
11) First set Sn is appended to pat by s-extension and second Si is appended by i-extension by join operation and then counting number of sequence where pattern appears.
12) Spam prune the search space by extending of pat are considered for extending patterns.

C. SPADE algorithm

SPADE (Sequential PAttern Discovery using Equivalent Class) .A vertical format sequential pattern mining method. A sequence database is mapped to a large set of Item: <SID, EID> Sequential pattern mining is performed by growing the subsequences (patterns) one item at a time by Apriori candidate generation.

1) Steps of algorithm[1]
2) Using candidate generation: Vertical data format
3) Spam first scan the input database SDB to construct vertical database V(SDB) and then set the frequent item F1.
4) For each item S belongs to F1.
5) Spade calls the ENUMERATE procedure with the equivalence class of size 0.
6) An equivalence class of size *k* is defined as the set of all frequent patterns containing *k* items sharing the same prefix of $k-1$ items.
7) The ENUMERATE procedure receives an equivalence class *F* as parameter.
8) Output of equivalence class is Ai initialized with empty set.
9) For each pattern Aj from large pattern and Ai is merged with Aj for large pattern
10) For each such pattern *r*, the support of *r* is calculated by performing a join operation between IdLists of *Ai* and *Aj* .
11) If the cardinality of the resulting IdList is no less than *minsup*, it means that *r* is a frequent sequential pattern.
12) It is thus added to *Ti*. Finally, after all pattern *Aj* have been compared with *Ai*, the set *Ti* contains the whole equivalence class of patterns starting with the prefix *Ai*. When all loops terminate, all frequent sequential patterns have been output

D. Integration in SPAM (CM-SPAM) [1]

1) Spam uses SEARCH procedure as follow
2) Sequential pattern both Sn and Si both in item x

3) If last item a in pat does not have an item then pattern resulting x with pat to count the support resulting pattern does not need to performed means there no need to join operation.
4) We do not have to check for extension of pat with x for all the items except for last one, have already been checked for extension in previous steps.
5) This is diagram of how we implement our algorithm and how we will improves our performance by generating rules and how to minimize time.

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

Proposed Algorithm

Input : Sequential Database, min_sup, min_conf.

Step1 : Generate Sequence Frequent patterns using CM-SPAM algorithm.

Step2 : For each Sequence frequent Pattern make rule from it.
Suppose $\alpha \Rightarrow \beta$.
Find conf = $\sup(\beta)/\sup(\alpha)$.

Step3: If (Conf >= min_conf && $\alpha \Rightarrow \beta$ must be in Sequence for each item in rules.)
[It is checked by s-extension which is generated by CM-SPAM]

Output : Sequential rules
Else
Ignore and go to Step 2 until all rules find

III. CONCLUSIONS

In rule growth algorithm rules are generated between two items and then recursively grows them by scanning the database for single items that could expand their left or right parts by this we generate redundant rules. Sequential pattern mining algorithms using the vertical format are very efficient because they can calculate the support of candidate patterns by avoiding costly database scans. In CM-SPAM algorithm can be used for pruning candidate generated by vertical algorithm. As studied previous years research papers found problem in existing algorithm CM-SPAM is not generating rules. In existing algorithm CM-SPAM we generate rules using sequential rule mining. We generate rule in existing algorithm, Compare this algorithm with RuleGrowth algorithm. Our proposed algorithm takes lesser time than existing RuleGrowth algorithm

REFERENCES

[1] Philippe Fournier-Viger, Antonio Gomariz, Manuel Campos, and Rincy Thomas:: "Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information": Springer International Publishing Switzerland 2014
[2] Chetna Chand, Amit Thakkar, Amit Ganatra "Sequential Pattern Mining: Survey And Current Research Challenges"::International Journal Of Soft Computing And Engineering ,March 2012
[3] Jian Pei, Jiawei Han, Behzad Mortazavi-aslJianyong Wang, Helen Pinto, Qiming Chen,Umeshwar Dayal, And Mei-chun Hsu "Mining Sequential Patterns By Pattern-growth The Prefixspan Approach" IEEE Transactions On Knowledge And Data Engineering,2004
[4] Carl H. MOONEY And JOHN F. Roddick: "Sequential Pattern Mining – Approaches And Algorithms" ACM Journal 2013
[5] V.Chandra Shekhar Rao And P.Sammulal: "Survey On Sequential Pattern Mining Algorithm" International Journal Of Computer Applications 2013
[6] Fournier-Viger, P., Nkambou, R., Tseng, V.S."RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth". In: Proc. ACM 26th Symposium on Applied Computing, (2011)
[7] Nidhi Grover Comparative Study of Various "Sequential Pattern Mining Algorithms" international journal of computer applications volume 90 – no 17, march 2014.
[8] Shigeaki Sakurai, Youichi Kitahata and Ryohei Orihara, "Discovery of Sequential Patterns based on Constraint Patterns", international journal of computational intelligence, 2008.
[9] 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
[10] X.Yan, and R.Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. Proceedings of the 2003 SIAM International Conference on Data Mining (SDM'03),. 2003
[11] Philippe Fournier-Viger, Usef Faghihi , Roger Nkambou1 , and Engelbert Mephu Nguifor "CMRules: Mining Sequential Rules Common to Several Sequences" Springer International Publishing Switzerland 2012
[12] Philippe Fournier-Viger, Antonio Gomariz, Manuel Campos, and Rincy Thomas:: "VMSP: E_cient Vertical Mining of MaximalSequential Patterns": Springer International Publishing Switzerland 2014.
[13] Thabet Slimani, And Amor Lazzez "Sequential Mining: Patterns And Algorithms Analysis" International Journal Of Computer And Electronics Research Cybernetics 2013
[14] Alpa Reshamwala, Neha Mishra "Analysis Of Sequential Pattern Mining Algorithms" International Journal Of Scientific & Engineering Research, Volume 5, Issue 2, February-2014

Websites

[1] [Http://Www.Thearling.Com/Text/Dmwhite/Dmwhite.Htm,2002]
[2] [Http://Himalaya-tools.Sourceforge.Net/Spam/2002].
lty/jason.frand/teacher/technologies