Constrained Top k Frequent Sequential Pattern Mining

Extension to extract Interesting Patterns

1st Muhammad Umair Ali
Fakultät für Informatik
Otto von Guericke Universität
Magdeburg, Germany
aliumair.ovgu@gmail.com

2nd Apoorva Chandrashekar
Fakultät für Informatik
Otto von Guericke Universität
Magdeburg, Germany
apoorva.byaladakere@st.ovgu.de

3rd Sesha Sai Kiran, Bhavaraju
Fakultät für Informatik
Otto von Guericke Universität
Magdeburg, Germany
sesha.bhavaraju@ovgu.de

4th Pawan Kumar
Fakultät für Informatik
Otto von Guericke Universität
Magdeburg, Germany
pawan.kumar@ovgu.de

5th Raj Rajeshwari Prasad
Fakultät für Informatik
Otto von Guericke Universität
Magdeburg, Germany
raj.prasad@st.ovgu.de

6th Ajay Kumar Chadayan
Fakultät für Informatik
Otto von Guericke Universität
Magdeburg, Germany
ajay.chadayan@st.ovgu.de

***Abstract*—Data Mining in its core sense consists of the procedures and explanations as to extract meaningful information from, typically a huge silo of data. The data in question might be sourced from various real-life scenarios like, Market basket analysis, for instance, which is collected from the purchases made by customers on a day to day basis in a general shopping scenario. The data might be then expressed in meaningful schemes like, Transactions, Customers and the relations between them. Pattern mining now, then becomes a task of extracting meaningful patterns which might be item-set (set of purchases) occurring multiple times which becomes a Frequent Pattern mining task. The evaluation framework discussed above is traditionally called support and is widely used in a Support Confidence Scenario. More precisely Sequential pattern mining consists of discovering interesting a set of Patterns in succession constituting a sequence. There are many real time examples for sequential pattern mining. In this scientific work our aim is to understand the task of Sequential Pattern mining including the key concepts and terminology also getting familiar to the main methods of finding interesting sequential patterns. Based on the existing approaches our goal is to develop and tune a method to mine sequential patterns from big data by utilizing various measures of interestingness and comparing our approach to the existing algorithms in terms of memory, time and recall. We also discuss the comparisons between these measures and their significance in our mining task.**

***Index Terms*—TKS, interesting patterns, Top k Frequent Sequential Pattern Mining, Lift, All-Confidence**

I. Introduction

Data Mining as can be formally defined as an intelligent search procedure to retrieve information. Data mining plays an important role in the modern day because we as a populous are more digitized than ever. We have devised intricate data sourcing procedures from our daily use of various bits of technology to implement a set of features onto consumer products or services like, Targeted advertisement technique to boost sales, or implementing various optimization techniques based on the kind of data being transferred on a product between services or over the internet. Data mining has enabled us to make intelligent design choices in the health care sector finding new ideas which were only possible via a thorough exploratory analysis. It has helped better plan routes in the transport sector based on the consumer and supplier information, optimizing fuel consumption or resource exhaustion in general.For the scope of this research paper we are only interested in a more typical form of data expressed in the form of Patterns and Sequences. We aim at extracting Frequent Sequential Patterns using the existing algorithms and modifying the said to include interesingness as an added constraint into question as well.

Sequential Pattern Mining is concept where in the information retrieved is in the form of Sequences, and the input being a Sequential Database. A sequence, in our case can be defined as a set of item-sets occurring in succession for a given atomic transaction. Item-sets can then be formally defined as a collection of items which might represent a product bought by a customer or a word in a sentence. The task of discovering Sequential Patterns has been achieved previously by many algorithms using a variety of evaluation metrics. Some popular examples of such algorithms are GSP , Spade , Spam , CM-Spam , CM-Spade , Fast Miner , and MAX Miner. The above mentioned algorithms all work on a sequential database, and require the user to specify a minimum Support threshold. The output is generally a set of frequent sequential patterns. The results of all the algorithms in question is reproducible, meaning the output does not vary if the input parameters are unchanged. The main differences between the said can then be boiled down to how each one handles the input, how the data is processed in its intermediate stages(Data Structures used)

and how each strategy used fairs against efficiency, usability, time required to execute, and the memory consumption.
In general, the above mentioned sequential pattern mining algorithms fall into these basic implementational approaches

- Breadth First search algorithms
- Depth First search algorithms
- Map Reduce based algorithms

II. Related Work

A. The GSP Algorithm

Generalized Sequential Pattern Algorithm also known as GSP is a Vanilla version algorithm to mine 'Sequential Pattern' from a Sequential Database. The understanding of the algorithm helped us to recognise the motivation behind successor algorithms that used GSP as the baseline for both development and comparison of the performance.

A GSP algorithm is based on the Apriori assumption i.e., If an item-set is not frequent then its extensions will be infrequent. Further, the GSP algorithm takes in a threshold value called 'Minimum Support Value' as a hyper parameter. This threshold value acts as a filter while eliminating infrequent item-sets. If an item-set has support value equivalent or greater than the threshold value, then the item-set is preserved and is used for further pattern extension, else it is eliminated. The preserved item-sets are called 'Frequent item-set'.

$$Support(a) = Occurrences of a / Total Items in database$$

TABLE I
Sample Database

SID	Sequence
1	({a}, {a,b}, {d}, {g})
2	({a}, {d}, {b})
3	({b},{g})

The "Tab. I" denotes a simple example of a sequential database with items denoted as I = (a, b, c, d, g).The GSP algorithm adopts 'Breadth-First' approach hence perform level-wise search to extent the itemsets to extract sequential patterns. It initially scans the database and the calculates the support of each item. Further, it prunes the elements which are infrequent. The remaining items are considered as valid candidates for pattern extension and are saved in the memory. In the next step, the candidates are extended to 2-sequence pattern. This method is called as 'Candidate Generation'. Post candidate generation, the new support of extended patterns are calculated considering the database. Suppose for the given example we have minimum support value = 2. Then all the items are initially considered as frequent. The 2-sequence extension after candidate generation will be (a,b),(a,c),(a,d), (a,g), (a, a), (a,b), (a,c),(a,d), (a,g).....(g,g). The extended items will be pruned and again the candidate list will be updated. The step of extending the pattern is done in a loop until no further extension of itemsets is feasible. The final list of extended items is considered as most frequent sequential items.

Though, the algorithm was considered to be a state of art, there are several demerits associated with the approach. Assignment of the Support threshold value is a cumbersome task and needs heuristics from user. The problem associated with this method of hit and trial is that with threshold value set too high, important patterns may be missed and with threshold value too less, irrelevant patterns may also be incorporated in the result. Preserving candidates in memory is an inefficient method. Also, the algorithm is computationally very inefficient as it needs to read the database again and again.

The very idea of dealing with the Minimum support value as the threshold motivated us to look for an alternate methodology to deal with the heuristics of assigning threshold.

B. SPADE (Sequential Pattern Discovery using Equivalence classes)

Below we have described the working of the SPADE algorithm, as the vertical database representation of our algorithm is similar to SPADE algorithm.

Spade uses a vertical database representation, which helps it to create the 1-frequent itemset and 2-frequent itemset in one scan of the database as opposed to multiple scans in GSP. The vertical database representation for "Fig. 2" can be seen in "Fig. 1".

SIDs	Sequences
1	({a, b},{c},{f,g},{g},{e})
2	({a, d},{c},{b},{a,b,e,f})
3	({a},{b},{f,g},{e})
4	({b},{f,g})

Fig. 1. Horizontal Database for SPADE

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	3
4	2

Fig. 2. Vertical Database for SPADE

The vertical dataset represents the SID and the places where it occurs in an itemset, like in SID-2 the item 'a' appears in itemset 1 and 4.

The vertical representation is also helpful as it helps us to calculate the support of a particular itemset quickly, as opposed to GSP. The support of a particular item is nothing but the number of distinct SIDs, in which the item appears. Like for the item 'a' the number of distinct SIDs, in which it appears is 3, hence we can say its support is 3.

This representation of database in used in several popular algorithms, like SPAM, TKS, CM-SPAM. This particular dataset

TABLE II
DATABASE FOR SPAM

Sequence Number	Sequence
1	ABBC
2	BCA
3	ABBCA

representation is further optimised by SPAM, by representing the data as bit vectors.

C. The SPAM Algorithm

The vertical database representation of the dataset has certain advantages over the horizontal database format which includes support calculation without performing costly database scans and providing better results for datasets with long and dense sequences in the terms of memory and speed.

The Spam algorithm is used for searching frequent sequential patterns from large databases, the algorithm uses a bitmap representation of the database for efficient support counting and uses depth first search strategy for pruning the infrequent candidates .The algorithm differs from existing vertical bitmap algorithm such as SPADE on the basis of candidate generation following the itemset extended sequences(I-Step) and sequence extended sequences(S-step).

The Spam algorithm begins with empty tree and uses the apriori assumption to consider frequent and infrequent sequences, the leaf nodes of the root are created using the candidate generation algorithm following sequence extended sequences and itemset extended sequences. Considering the lexicographic analysis "Fig. 3" for the dataset "Tab. II" provided , we could observe that the algorithm begins with empty string with root node as .Assuming A , B , C items separately at level 1 of the tree , for item A using the candidate generation process we can generate sequence extended sequences as A, A , A, B , A, C and itemset extended sequences as (A, B) and (A, C) at level 2. Similarly at each tree level we would generate sequences with length equal to the level of tree.

After the entire tree is generated the algorithm traverses using depth first search to test the support of sequences at each node level , nodes with support value equal or greater than the minimum support threshold are stored and the remaining sequences are pruned , the algorithm repeats the depth first strategy recursively on the nodes. Candidate Generation is done via an S extension which extends the patterns as a Sequence extension and the I Step which pertains to an item-set extension. The exact extension procedure is further explained in the implementation of our custom algorithm, which incorporates a modified and operation. Pruning of the infrequent nodes are performed assuming the apriori principle through the S-step and I-step of the node, for e.g.: considering for S-step given sequences (A, A), (A,B) and (B, C) and (A, B) is infrequent then the following sequence would be also considered infrequent: (A A B) , (A C B) , (A B,C) Similarly, for I-step it would apply the similar apriori principle for pruning the candidate children generated from I-step, for e.g.: considering item set sequences (A,B) and (A,C), if (A,C) is infrequent then (A,C,B) will be also infrequent. However, vertical algorithms face a crucial limitation of generating candidates and then testing which results in generating a large number of infrequent candidates

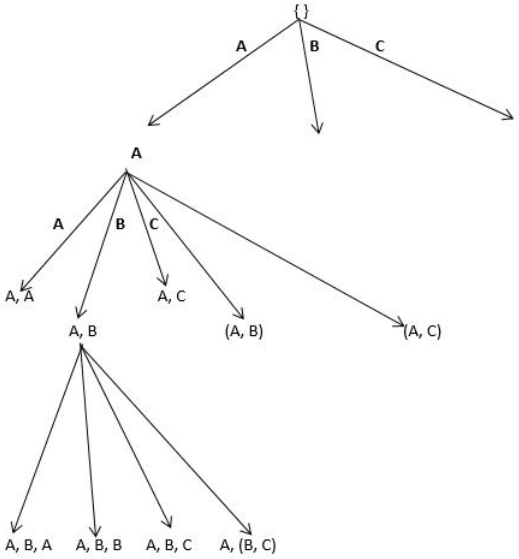


Fig. 3. Lexicographic Sequence Tree

TABLE III
A Sequential Database

SIDs	Sequences
1	({a},{abc},{ac},{d},{cf})
2	({ad},{c},{bc},{a,e})
3	({ef},{ab},{df},{c},{b})
4	({e},{g},{af},{c},{b},{c})

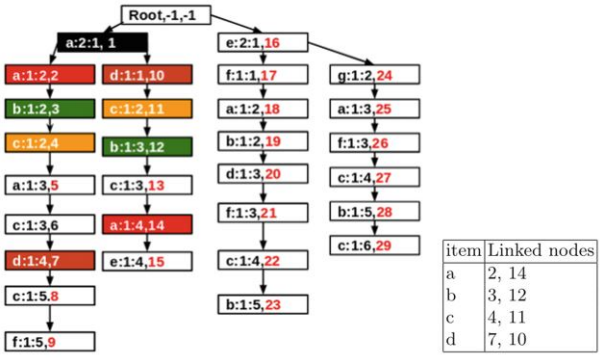
D. Tree Miner

A very common apriori like method, such as GSP which mostly works on candidate generation and test purposes. There are very high computational costs involved because of huge numbers of candidates, also large amounts of database scans. This causes the database to perform slow in most of the cases. To overcome such issues pattern-growth based algorithms show some effective results by following divide and conquer approach which in apriori based algorithms is difficult to achieve. Thus, major improvements are required for such algorithms. In paper [8] the authors have come up with a tree based approach to store the sequential patterns in the database in an efficient manner. The proposed method consists of three stages. At first they proposed a tree based data structure named SP-Tree to store the patterns in the database in an effective manner with a specific property of build once, mine many. Afterwards they tried to mine those patterns using the proposed algorithm Tree-Miner. They then performed several pruning techniques to the approach to reduce the run time, also keeping the scope of scalability and extensibility.

Considering Table II. they have defined their node structure for the tree. Where each sequence have a label for the item

and named it as node's label. An Event Number to represent those labels and finally the count, which shows the number of times the node has been traversed during the construction of that tree from the given sequence. A very unique approach (the recursive next link) has been introduced in this paper as well. Which works in a way that for every node there is a next link move as shown in fig(3) which demonstrates the occurrence of the first node in the sub-tree. By introducing this new method they intended to move faster through the nodes so the computational costs can be reduced. It is important for each node to store the information about its parent node labels in the same event so the path to the root node is known. This helps in reducing the search space of the algorithm. In this approach, they used a bit set to store such information. For Example, if the items are numbered as 0, 1, 2, it is stored for the items 'a' as 0, 'b' as 1, 'c' as 2 and respectively for further nodes. Such a bit based approach will help the algorithm to improve the performance in terms of run-time. The authors then used the approach of co-existing tables to understand which item co-exist in the database either in the same or different event.

Further the authors talked about some pattern expanding approaches which means it starts from an empty set of sequence and by traversing gradually using the next link approach it adds new items at the very end of the events as Item Extension or Sequence Extension. For the proposed algorithm Tree-Miner the authors then defined specific pattern formations rules such as Node Concatenation by Item-set Extension which includes the concept of direct and indirect node in order create a node-List so that we can have an idea where the pattern ends.



item	Linked nodes
a	2, 14
b	3, 12
c	4, 11
d	7, 10

Fig. 4. Next Link for Node 1

For the pruning purpose of the approach they then introduced a list of pruning techniques such as Coexisting Item Table Based Pruning which defined the satisfied condition of performing actual candidacy measure of support counting. sList and iList Pruning and Heuristic iList Pruning to compute the attribute value of the parent node. After performing their experiment on different datasets they concluded their approach to be very efficient with a comparative analysis to Prefix and SPAM.

E. The TKS Algorithm

We have discussed a set of algorithms above which had their specific merits and demerits. All of them had one thing in common which was that they required a minimum support parameter to be set at the beginning of the algorithmic loop. This is a huge problem when we have a huge database in consideration with millions of entries. It is already hard in this scenario to evaluate on the entire database for a specific support threshold which is initially set by the statistician. Upon the non use-fullness of the results the minimum support value would then have to be lowered to extract more number of sequences hoping for a better result this time. It is apparent how the situation needs multiple Database Scans and would thereby require the resources for the same. This situation might be mitigated with the help of a strategy proposed in the Top K Frequent Sequential pattern strategy described by this algorithm. This algorithm takes a sequential Database as an input and performs 1 database scan to construct a vertical bitmap structure as described in the SPAM algorithmic approach. A vertical bitmap representation is relatively less expensive in terms of computation and memory to maintain and will support logical combinatorial operations. For instance combining two item-sets is effectively a logical and operation where the resultant item-set would have the support equal to the cardinality of its bit vector or bit set. This is because the support information is already captured whilst constructing the bit set in the first place. Hence this transitive property holds, if $Support(\{a\}) = Cardinality(BitVector\{a\})$ and $Support(\{b\}) = Cardinality(\{b\})$ then $Support(\{a, b\}) = Support(\{a\} + \{b\}) = Cardinality(Bitvector(\{a\}\{b\}))$. However this relation only holds true in case of Item extended item-sets. For sequence extensions the authors have followed the modified and operation from SPAM as well. This algorithm proposes a method for raising the internal minimum support value which is initially set to 0, after the candidate pruning. Candidate pruning is done on an Apriori principle that if a sequence does not qualify the internal minimum support threshold, all its extensions are deemed infrequent as well. This disregard to infrequent items in the algorithm also improves the execution time as the search space shrinks as time progresses. The algorithm maintains different lists for storing Candidate patterns and Top K patterns. The internal minimum support is always set to the minimum support after populating the top K list. It is also important to note that this algorithm generates at-least K frequent sequential patterns. If the support of two Extended sequences are the same both of them are kept.

We have chosen to work on this kind of approach in our scientific experiments mainly attributing to these reasons,

1) It mitigated the burden of setting the minimum support initially.
2) It made use of a clearly distinguishable *SEARCH* procedure "Fig. 16" and *SAVE* procedure "Fig. 17" which brought in a better separation of concerns and easier to modify individual parts.

3) It was also the most efficient as far as scanning the database is concerned (1 Scan).
4) The Execution time and The memory consumption was the most competent especially considering the input size.
5) There were already optimization procedures already described which could be implemented to further increase its efficacy.

III. Proposed Approach

A. Datasets

The SPMF website provides us with many datasets for free public or educational use. The databases are either synthetically generated or books from different languages are translated to sequence databases. This is done by considering each word as an item. Generally, all the datasets follow a general structure where in each itemset is separated by a -1 and the sequence is separated by a -2. For instance, "Fig. 5" can be interpreted as "Fig. 6"

```
1 -1 1 2 3 -1 1 3 -1 4 -1 3 6 -1 -2
1 4 -1 3 -1 2 3 -1 1 5 -1 -2
5 6 -1 1 2 -1 4 6 -1 3 -1 2 -1 -2
5 -1 7 -1 1 6 -1 3 -1 2 -1 3 -1 -2
```

Fig. 5. SPMF Dataset Format

ID	Sequences
S1	(1), (1 2 3), (1 3), (4), (3 6)
S2	(1 4), (3), (2 3), (1 5)
S3	(5 6), (1 2), (4 6), (3), (2)
S4	(5), (7), (1 6), (3), (2), (3)

Fig. 6. SPMF Dataset Represented as Sequences

Some properties of the dataset can be found in the "Fig. 7".

Dataset	Sequences	Distinct Items	Avg. Seq Length(items)
Leviathan	5834	9025	33.81 (std = 18.6)
Bible	36369	13905	21.64 (std = 12.2)
Sign	730	267	51.99 (std =12.3)

Fig. 7. Data-set Properties

B. Database Representation

We used the baseline TKS algorithm to find top k frequent sequential patterns. TKS uses the vertical database representation and the candidate generation procedure from SPAM are employed. We explain the candidate generation procedure of SPAM by which sequences are extended in two parts, I extension which extends a sequence item wise, i.e. intra-itemset and S- extension which extends an already existing pattern sequence wise, i.e. Inter itemset below. But first we would like to discuss a few things about the vertical database representation and the way support is calculated. Let the figure below "Fig. 8" be the sequence database in consideration.

SID	Sequences
1	$\langle\{a, b\},\{c\},\{f, g\},\{g\},\{e\}\rangle$
2	$\langle\{a, d\},\{c\},\{b\},\{a, b, e, f\}\rangle$
3	$\langle\{a\},\{b\},\{f\},\{e\}\rangle$
4	$\langle\{b\},\{f, g\}\rangle$

Fig. 8. Sequential Database

TABLE IV
Vertical Database

SID	1					2				3				4	
Item	{a,b}	{c}	{f,g}	{g}	{e}	{a,d}	{c}	{b}	{a,b,e,f}	{a}	{b}	{f}	{e}	{b}	{f,g}
a	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0
b	1	0	0	0	0	0	0	1	1	0	1	0	0	1	0
c	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
d	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
e	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
f	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1
g	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1

We construct the vertical database as follows. Vertical Database of the sequential Database is defined as a set of q bit vectors of size n(where n represents the number of unique SID and ItemsetID) such that each item x has a corresponding bit vector BitVecx. Each bit of a bit vector BitVecx is set to 1 if and only if x appears in the itemset represented by this bit. If x does not appear in the itemset represented by this bit it is set to 0. For Example the table below "Tab. IV" shows the conversion of the Database above to a vertical database

Calculation of Support of an item: Lets say we need to calculate the support of an item a for instance. We take the bv(a) as a bitset and get the cardinality of the same. In our case the bv(a) contains 4 bits set to true. So the Supp(a) would be 4.

IV. Candidate Generation With SPAM

The S-step is the process for generating Sequence extended children sequences with the help of depth first search strategy for mining sequential pattern. For instance, if we have a sequence represented as this,

```
S = {a, b, c, d}
```

the possible sequence-extended sequences are

```
({a},{a}),({a},{b}),({a},{c}),({a},{d})
```

On the Other I-Step is the process for generating Itemset extended children sequences with the help of depth first strategy. For example, for the above defined sequence S(a) we would have possible itemset extended sequence as

```
({a,b}),({a,c}),({a,d})
```

The Initial step for the algorithm iterates with transforming our sequences into Vertical bitmap representation and illustrative example for a sample sequence as follows

The Bitmap representation for the above sequence after transformation looks as follows

A possible visualization for the above transformed bitmap representation for S-extension and I-extension could be as follows, For the bitmap representation of the S-step "Fig. 11", this procedure was described by the authors, to generate $(\{a\})_s$ from $(\{a\})$. Every bit in $(\{a\})_s$, after the first index

CID	Sequence
1	$(\{a,b,d\},\{b,c,d\},\{b,c,d\})$
2	$(\{b\},\{a,b,c\})$
3	$(\{a,b\},\{b,c,d\})$

Fig. 9. Initial Sequential Database

CID	TID	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$
1	1	1	1	0	1
1	3	0	1	1	1
1	6	0	1	1	1
-	-	0	0	0	0
2	2	0	1	0	0
2	4	1	1	1	0
-	-	0	0	0	0
-	-	0	0	0	0
3	5	1	1	0	0
3	7	0	1	1	1
-	-	0	0	0	0
-	-	0	0	0	0

Fig. 10. Bitmap Representation

$(\{a\})$		$(\{a\})_s$		$\{b\}$		$(\{a\},\{b\})$
1		0		1		0
0		1		1		1
0		1		1		1
0		1		0		0
0	S-step	0		1	result	0
1	$\longrightarrow$	0	&	1	$\longrightarrow$	0
0	process	1		0		0
0		1		0		0
1		0		1		0
0		1		1		1
0		1		0		0
0		1		0		0

Fig. 11. S Step Extension

$(\{a\},\{b\})$		$\{d\}$		$(\{a\},\{b,d\})$
0		1		0
1		1		1
1		1		1
0		0		0
0		0	result	0
0	&	0	$\longrightarrow$	0
0		0		0
0		0		0
0		0		0
1		1		1
0		0		0
0		0		0

Fig. 12. I Step Extension

of 1 in $(\{a\})$, is set to 1, noting the start of the modified *AND* procedure. $(\{a\})_s$ and $(\{b\})$ logically combined with an AND operation to construct $(\{a\},\{b\})$. For the bitmap representation of the I-Step "Fig. 12", new candidates are generated by performing a logical AND operation between the Bit vectors of $(\{a\},\{b\})$ and $(\{d\})$ to construct $(\{a\},\{b,d\})$

V. Interestingness measures

Interestingness of an itemset is an important measure to define if we want to measure the quality of an item set. Many of the Pattern mining algorithms depend heavily on the support confidence framework where in each item is either pushed further for extension into candidate patterns depending on just the support value. This is also true in case of the traditional TKS approach which considers patterns for extension based on support. The algorithm doesn't incorporate any constraints with respect to the mutual independence of an item within the itemset or the confidence of an itemset. We have seen a potential feature option to add interestingness measure based pruning technique at this point in the algorithm. We think interestingness is an important measure to consider as a constraint because support of an itemset alone cannot capture the importance entirely. We explain the previous statement with the help of this example. Lets say a typical market basket analysis has grouped coffee, milk as a potential itemset for the generation of association of rules. Lets assume the itemset has a non-trivial support value of 30 percent. This typical scenario when considered for an S extension or an I extension of the candidate generation process might not capture the importance of the itemset entirely. Milk might already be a very popular item with better support of (70 percent) and coffee might be discounting the chances of the entire itemset. It might help in this scenario to calculate the Lift value (Interest) of the itemset might be helpful because it typically measures the departure of an itemset from independence. We consider two measures for pruning the candidate further after the S and I extensions from SPAM, which we discuss below, All Confidence and Lift (Interest).

A. All Confidence

All Confidence is our first interestingness measure. All-confidence is defined on item sets (not rules) as

$$all\text{-}confidence(X) = \frac{supp(X)}{max_{x \in X}(supp(x))} = \frac{P(X)}{max_{x \in X}(P(x))} = min\{P(X|Y), P(Y|X)\}$$

Fig. 13. All Confidence

Where maxxX(supp(xX)) is the support of the item with the highest support in X. All-confidence means that all rules which can be generated from itemset X have at least a confidence of all-confidence(X).This indicates that there is a dependency between the items in the itemset and no item is discounting the other. The degree of this dependence can be calculated by specifying a threshold value. It can be inferred that if an itemset qualifies a threshold value any usage of this itemset in further extensions will mean that each item in the further extensions will have a greater chance of having the mutual correspondence. The All-Confidence value of an itemset could also be computed efficiently if we have the support values of the individual components which we do in our case.

B. Lift

Lift is our second interestingness measure. Lift or interest on an itemset (A,B) is defined as

Lift typically measures the departure from independence in an Itemset. A typical lift value of 1 would mean that we have

$$\frac{support(\{A,B\})}{support(\{A\}) \cdot support(\{B\})}.$$

Fig. 14. Lift

independence in the itemset and can also be incorporated to measure the interestingness of an itemset. It can be implied that if an itemset has a lift value of 1 and a high support, it might mean its further extensions are more meaningful than if the items inside the itemset discounted each other.

VI. Experiment with TKS

The existing TKS mining loop of TKS is described in the pseudocode below. The search and save procedures are used as described in the pseudocode below. We use these steps in addition to a pruning step while populating the candidate list for further extension. TKS natively generates candidates for extension based on the support value to calculate the frequency of a generated pattern using the Vertical Database as described above. We continue mining until we have K patterns in the final list. TKS is guaranteed to generate K patterns at least, and might generate more if there are multiple sequences with the same support value.

```
TKS(SDB, k)
1.  R := ∅. L := ∅. minsup := 0.
2.  Scan SDB to create V(SDB).
3.  Let S_init be the list of items in V(SDB).
4.  FOR each item s ∈ S_init, IF s is frequent according to bv(s) THEN
5.      SAVE(s, L, k, minsup).
6.      R := R ∪ {<s, S_init, items from S_init that are lexically larger than s>}.
7.  WHILE ∃ <r, S1, S2> ∈ R AND sup(r) ≥ minsup DO
8.      Select the tuple <r, S1, S2> having the pattern r with the highest support in R.
9.      SEARCH(r, S1, S2, L, R, k, minsup).
10.     REMOVE <r, S1, S2> from R.
11.     REMOVE from R all tuples <r, S1, S2> ∈ R | sup(r) < minsup.
12. END WHILE
13. RETURN L.
```

Fig. 15. TKS Mining Loop

```
SEARCH(pat, S_n, I_n, L, R, k, minsup)
1.  S_temp := I_temp := ∅
2.  FOR each item j ∈ S_n,
3.      IF the s-extension of pat is frequent THEN S_temp := S_temp ∪ {i}.
4.  FOR each item j ∈ S_temp,
5.      SAVE(s-extension of pat with j, L, k, minsup).
6.      R := R ∪ {< s-extension of pat with j, S_temp, all elements in S_temp greater than j>}.
7.  FOR each item j ∈ I_n,
8.      IF the i-extension of pat is frequent THEN I_temp := I_temp ∪ {i}.
9.  FOR each item j ∈ I_temp,
10.     SAVE(i-extension of pat with j, L, k, minsup).
11.     R := R ∪ {< the s-extension of pat with j, S_temp , all elements in I_temp greater than j >}.
```

Fig. 16. TKS Search Loop

A. Candidate Pruning based on Interestingness Parameters

We have modified the existing TKS mining loop to incorporate these interestingness measures we have discussed above. We have altered the way sequences are considered for further extension. We experimented with various threshold values for all-confidence and Lift and tried to generate the same amount of sequences as the traditional TKS algorithm

```
SAVE(r, L, k, minsup)
1.  L := L∪{r}.
2.  IF |L| >k THEN
3.      IF sup(r) > minsup THEN
4.          WHILE |L| > k AND ∃s ∈ L| sup(s) = minsup, REMOVE s from L.
5.      END IF
6.      Set minsup to the lowest support of patterns in L.
7.  END IF
```

Fig. 17. TKS Save Loop

did. The values have been heuristically set to mimic generation of similar amount of explored sequences from the original TKS algorithm. We do not further extend sequences which do not qualify a specific threshold and stop the further mining of such patterns. We have programmed all the changes in Java and used the original implementation of the algorithm to make our experiment comparable.

VII. Outputs and Evaluations

We infer that our modified approach should perform similar to the original TKS algorithm because we have added a constraint and havent taken away anyting from the original algorithm. We also understand that our algorithm might utilize more memory as we need to hold more items in memory and more properties for each itemset and a sequence. We also estimate that our algorithm might explore more Candidates as the set of candidates that we discard is a superset of the original TKS discard pile. We can aim to generate more interesting patterns than just generating frequent sequential patterns. We have run our modified approach against the original settings of TKS and performed similar memory usage and execution time comparisions. We experimented with 3 different k parameters 1000, 2000, 3000 to assess its influence on our modified approach. We also considered the Leviathan Book dataset and performed the comparisions considering 20, 40, 60, 80 percent of the original dataset.

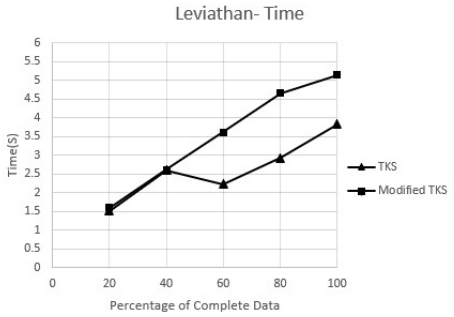


Fig. 18.

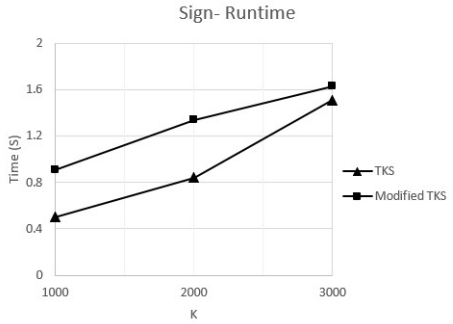


Fig. 19.

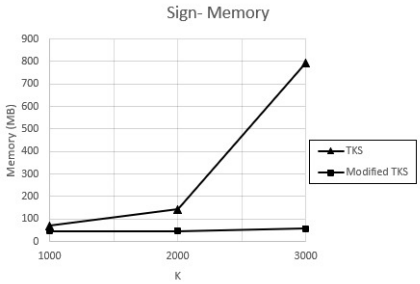


Fig. 20.

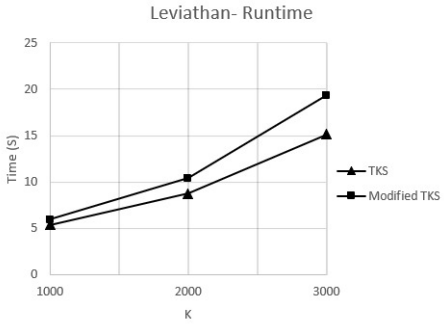


Fig. 21.

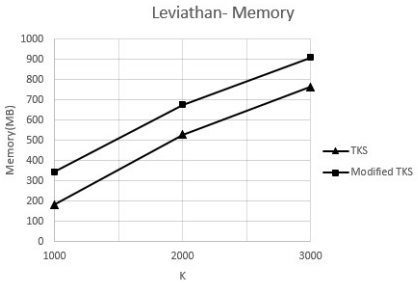


Fig. 22.

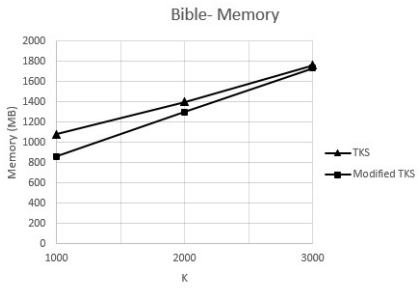


Fig. 23.

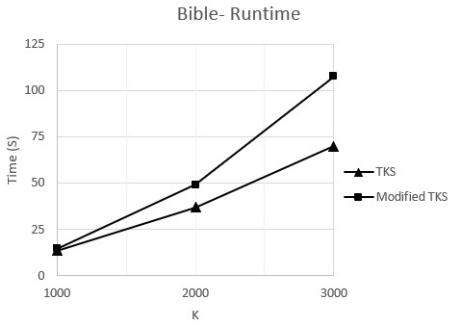


Fig. 24.

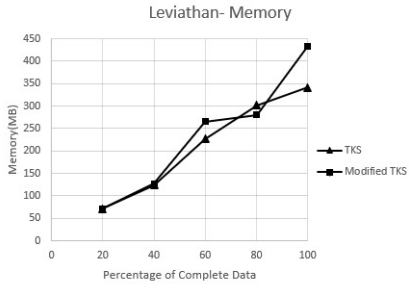


Fig. 25.

VIII. Conclusion

We have proposed a method to incorporate interesting ness to the exiting TKS algorithm. We however were unsuccessful in implementing the algorithm on spark which was our initial aim at this project. We assume that the proposed method generates interesting patterns within our evaluation metrics and constraints. We however havent discussed on how interesting-ness of a pattern could be intuitively visualized. The proposed approach aims at finding top k frequent sequential patterns with interestingness as a added constraint for candidate pruning. We thank our supervisors for the constant help and support in our hardships. They helped us learn a lot about sequential pattern mining.

References

[1] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. 2002. Sequential PAttern mining using a bitmap representation. In Proceedings of the eighth ACM SIGKDD international conference

on Knowledge discovery and data mining (KDD '02). Association for Computing Machinery, New York, NY, USA, 429–435. DOI:https://doi.org/10.1145/775047.775109

[2] Fournier-Viger P., Gomariz A., Gueniche T., Mwamikazi E., Thomas R. (2013) TKS: Efficient Mining of Top-K Sequential Patterns. In: Motoda H., Wu Z., Cao L., Zaiane O., Yao M., Wang W. (eds) Advanced Data Mining and Applications. ADMA 2013. Lecture Notes in Computer Science, vol 8346. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-53914-5_10

[3] Edward R. Omiecinski. 2003. Alternative Interest Measures for Mining Associations in Databases. IEEE Trans. on Knowl. and Data Eng. 15, 1 (January 2003), 57–69. DOI:https://doi.org/10.1109/TKDE.2003.1161582

[4] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. 1997. Dynamic itemset counting and implication rules for market basket data. In Proceedings of the 1997 ACM SIGMOD international conference on Management of data (SIGMOD '97). Association for Computing Machinery, New York, NY, USA, 255–264. DOI:https://doi.org/10.1145/253260.253325

[5] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. 1997. Dynamic itemset counting and implication rules for market basket data. SIGMOD Rec. 26, 2 (June 1997), 255–264. DOI:https://doi.org/10.1145/253262.253325

[6] R. Srikant, and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," The International Conference on Extending Database Technology, pp. 1-17, 1996.

[7] P. Fournier-Viger, Jerry C. W. Lin, R. U. Kiran, Y. S. Koh and R. Thomas, "A Survey of Sequential Pattern Mining", Data Science and Pattern Recognition, vol.1,February 2017

[8] M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," Machine learning, vol.42(1-2), pp. 31-60, 2001.

[9] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.429-435, 2002.

[10] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information," The Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 40- 52, 2014.

[11] Jay Ayres, Johannes Gehrke, Tomi Yiu, and Jason Flannick, "Sequential Pattern Mining using A Bitmap Representation", Dept. of Computer Science, Cornell University, 2002.

[12] Tree-Miner: Mining Sequential Patterns from SP-Tree Redwan Ahmed Rizwee, Mohammad Fahim Arefin, and Chowdhury Farhan Ahmed ' Department of Computer Science and Engineering Universirty of Dhaka, Bangladesh.

[13] P. Fournier-Viger, and V. S. Tseng, "Mining top-k sequential rules, The International Conference on Advanced Data Mining and Applications, pp. 180-194, 2011.

[14] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng, "Mining top-k association rules, The Canadian Conference on Artificial Intelligence, pp.61-73, 2012.

[15] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information," The Pacic-Asia Conference on Knowledge Discovery and Data Mining, pp. 40-52, 2014.

[16] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," The International Conference on Very Large Databases, pp. 487-499, 1994.

[17] S. Aseervatham, A. Osmani, and E. Viennet, "bitSPADE: A latticebased sequential pattern mining algorithm using bitmap representation," The International Conference on Data Mining, pp.792-797,2006

[18] B. Vo, T. P. Hong, and B. Le, "DBV-Miner: A Dynamic Bit-Vector approach for fast mining frequent closed itemsets," Expert Systems with Applications, vol. 39(8), pp. 7196-206, 2012.

[19] JinDai, LiangGuo, PengZhang,"Mining Optimized Frequent Itemsets Based on Max-miner: A Distributed Approach", Cloud Computing and Big Data Analytics, 2020