TKAR: Efficient Mining of Top-k Association Rules on Real—Life Datasets

O. Gireesha and O. Obulesu

Abstract Data mining is an important facet for discovering association rules among the biggest scope of itemsets. Association rule mining (ARM) is one of the techniques in data processing with the two sub processes. One is identifying frequent itemsets and the other is association rule mining. Frequent itemset mining has developed as a major issue in data mining and assumes an essential part in various data mining tasks, for example, association analysis, classification, etc. In the structure of frequent itemset mining, the outcomes are itemsets which are frequent in the entire database. Association rule mining is a basic data mining task. Researchers developed many algorithms for finding frequent itemsets and association rules. However, relying upon the choice of the thresholds, present algorithms become very slow and produce a greatly large amount of outcomes or generates few outcomes, omitting usable information. Furthermore, it is well-known that an expansive extent of association rules produced is redundant. This is truly a significant issue because in practice users don't have much asset for analyzing the outcomes and need to find a certain amount of outcomes within a limited time. To address this issue, we propose a one of a kind algorithm called top-k association rules (TKAR) to mine top positioned data from a data set. The proposed algorithm uses a novel technique for generating association rules. This algorithm is unique and best execution and characteristic of scalability, which will be a beneficial alternative to traditional association rule mining algorithms and where k is the number of rules user want to mine.

Keywords Association rules • Data mining (DM) • Frequent itemset mining (FIM) • k • Min_conf • Transactional database

O. Gireesha (✉) · O. Obulesu
Department of Information Technology, Sree Vidyanikethan Engineering College, Tirupati 517502, Andhra Pradesh, India
e-mail: gireesha93@gmail.com

O. Obulesu
e-mail: oobulesu681@gmail.com

1 Introduction

Data Mining is vital for data analysis in numerous fields as in the references [1–3]. It has pulled in a lot of consideration in the information industry and in the public arena as a whole in recent years because of the wide accessibility of immense measures of data and the impending requirement for transforming such data into valuable information and knowledge. The tasks of data mining are of two types: Descriptive and predictive mining. Descriptive mining characterize the significant features in database. The different techniques of descriptive mining are clustering, association rule mining, summarization and sequential pattern mining. Predictive mining forecast the unknown or hidden values of variables utilizing their existing values. Predictive mining techniques are classification, deviation detection, regression, and prediction (Fig. 1).

Association Rule Mining (ARM) is one of the significant issues in mining prospect which was presented in [4] and extended in [5] to produce association rules. ARM deal with invention of valuable co-occurrence and correlations among itemsets present in vast number of transactions. Rules found fulfill user defined minimum support and confidence thresholds. The generation of association rules can be produced in two parts. Firstly, we can discover frequent itemsets from the datasets, and the secondary deals with the generation of association rules from those frequent patterns [6] (Fig. 2).

Frequent pattern mining is a critical zone of Data mining research. The occurrence of items repeatedly within transactions in a transactional database is called frequent itemsets. Frequent itemset mining will be used to find useful patterns over customer's transaction databases. This kind of finding item set helps organizations to make significant decisions, for example, inventory drawing, cross marketing and customer shopping performance scrutiny. A consumer's transaction database

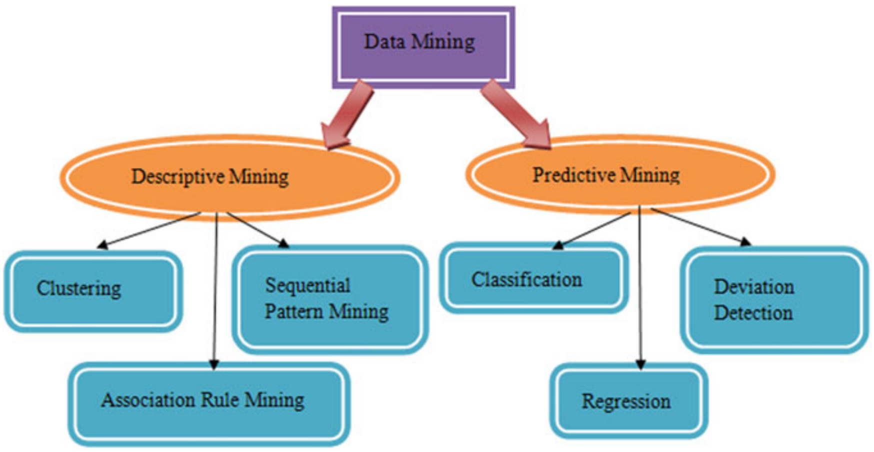

Fig. 1 Categories of data mining tasks [7]

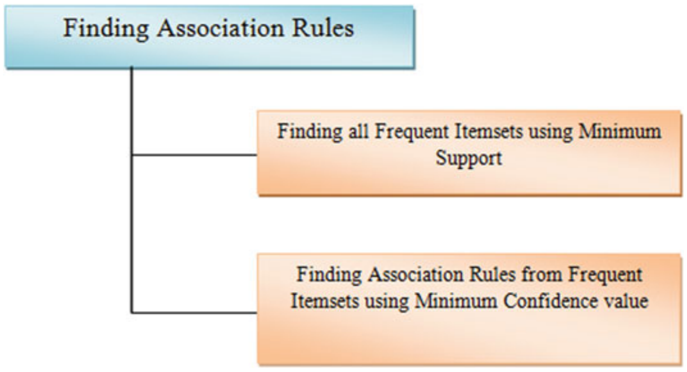
Finding Association Rules
Finding all Frequent Itemsets using Minimum Support
Finding Association Rules from Frequent Itemsets using Minimum Confidence value

Fig. 2 Generation of Association Rules [8]

comprises of set of transactions, where each and every transaction is an itemset. The purpose of frequent itemset mining is to mine all frequent itemsets in a transactional database. Frequent itemset mining can be ordered into different classifications based upon the type of itemset to be mined. These are discussed in detail in the below [7] (Fig. 3):

(a) **Approximate Frequent Itemset**: When an itemset derive approximate support count from its supersets.
(b) **Closed Frequent Itemset**: An itemset is closed if none of its prompt supersets has the same support as an itemset.
(c) **Constrained Frequent Itemset**: An itemset which satisfy user-specifies set of rules through mining.

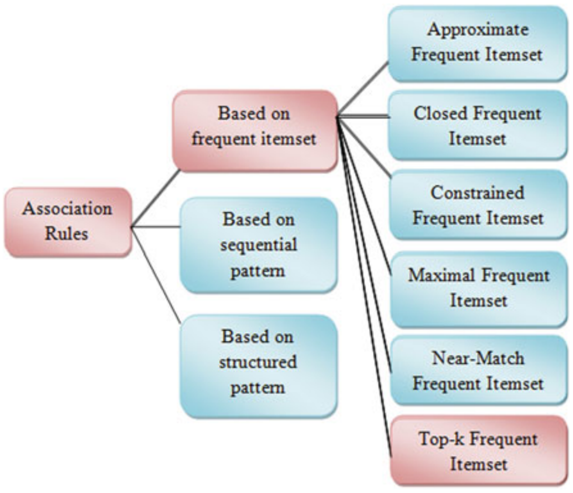
Association Rules
Based on frequent itemset
Based on sequential pattern
Based on structured pattern
Approximate Frequent Itemset
Closed Frequent Itemset
Constrained Frequent Itemset
Maximal Frequent Itemset
Near-Match Frequent Itemset
Top-k Frequent Itemset

Fig. 3 Kind of pattern to be mined with Association Rules [7]

(d) **Maximal Frequent Itemset**: An itemset is maximal frequent if none of its prompt supersets is frequent.
(e) **Near-Match Frequent Itemset**: An itemset is said to be near-match when its support count is same that of its near-by or similar items.
(f) **Top-k Frequent Itemset**: Out of complete set of frequent itemset, top itemsets that pass user determined value "k".

2 Literature Survey

In the preceding discussion we have studied the fundamental ideas of data mining, Association Rule Mining and frequent itemset mining. This segment will proceed with the literature review of some popular algorithms related to these mining concepts which have been presented from many years in different researches. In this section, we examine closely related past studies and contrast them with our methodology.

The AIS (Agrawal, Imielinski, Swami [4]) algorithm was introduced to mine the association rules [4]. It has some expertise in the standard of databases alongside essential to method call support queries. Through this algorithm one frequent itemsets are produce, which suggests that the following of these guidelines exclusively contain one thing, as a case we tend to exclusively produce rules like $A \cap B -> C$ however not those tenets as $A -> B \cap C$. The database is scanned number of times to generate the frequent itemsets in AIS. The bottleneck of this algorithm is several candidate itemsets are produced that requires more space and scans.

The AprioriTID algorithm [5] generates the candidate itemsets. The difference between AIS and AprioriTID is it requires only one database scan and a group of candidate itemsets employed for $k > 1$. Apriori and AprioriTID use consistent candidate generation methodology and calculation sturdy itemsets. Apriori Hybrid algorithm uses Apriori inside the underlying passes and changes to AprioriTid once it expects that the candidate itemsets at the tip of the pass will be in memory. In the SETM algorithmic guideline, candidate itemsets square measure produced on-the-fly in light because the data is scanned, however counts at the highest point of the pass. The disadvantage of SETM algorithm is similar to the AIS algorithm. Another weakness is that for each candidate itemset, there square measure as a few entries as its support value.

The FP-Growth Algorithm [8], proposed by J. Han requires constructing FP-tree. It is a productive and scalable technique for mining the complete set of frequent patterns through pattern fragment growth, utilizing an amplified prefix-tree structure for storing compacted and pivotal data about frequent patterns (FP) called as Frequent Pattern Tree (FP-tree). This algorithm needs 2 passes irrespective of candidate generation and utilizes divide and conquer approach. The database scanned two times. In the first scan, it maintains a list of frequent items sorted by recurrence

Table 1 Input parameters description

S. no.	Input parameters	Description
1.	TD/Dataset	A transactional database
2.	k	Number of rules user wants to find
3.	Minconf	Minimum confidence threshold value

in dropping order and in the second scan; the FP-tree is generated through the database compression. One issue, which hampers the well known utilization of frequent itemset mining, is that it is troublesome for users to choose a legitimate minimum support threshold. To precisely control the number of the most regular itemsets, is to mine top-*k* frequent itemsets [9–17] rather than to mine frequent itemsets. The setting of k—value is more instinctive than setting the minimal support threshold because *k* represents the number of itemsets the user needs to discover while choosing the threshold depends exclusively on database's attributes, which are frequently obscure to users [18]. Despite the fact that these algorithms are different in that they make use of different data structures and pursuit strategies to find top-*k* frequent itemsets, they follow a same general procedure for finding top-*k* itemsets (Table 1).

3 Top-k Association Rules Mining (TKAR) Algorithm

The Top-k Association Rules mining algorithm takes as input a transactional database, minimum confidence threshold and k the number of rules user wants to discover. The TKAR algorithm is an advantage to conventional association rule mining algorithms meant for users to control the number of association rules. Table 2 presents an overview of the input parameters for the proposed algorithm.

The proposed TKAR algorithm will work as follows:

Algorithm (dataset, k, minconf)

1. minsup = 1; C = NULL; D = NULL
2. dataset is scanned one time and the sids (sequential ids of each item I is stored in a variable called Tids(i)
3. For every pair of items P & Q Where Tids(P) > = min_sup & Tids (Q) > = min_sup, Set Tids(P => Q) = Null & set Tids(P =>Q) = Null

Table 2 Dataset Characteristics

Datasets	No. of transactions	No. of distinct items	Average transaction size
Chess	3196	75	37
Connect	67557	129	43
Mushrooms	8124	128	23
Pumsb	49046	7116	74

```
4. For each Tid which belongs to Tids(P) ∩ Tids (Q)
5. If P occurs before Q in the Tid s then Tids(P => Q) = Tids (P => Q) U {s}
   OR If Q occurs before P in Tid s then Tids(Q => P) = Tids (Q => P) U {s}
6. If │Tids (P => Q) │/│Total sequences│ > = minimum support Then
7. Confidence (P => Q) = │Tids (P =>Q) │/│Tids(P) │
8. If confidence (P =>Q) > = minimum confidence then
9. If │C│ < K then
10. If there exist a rule r2 in C which is similar to the currently generated rule r1 &
    whose support is also similar to the support of r1 then the rule r1 is not added to
    C Otherwise this rule r1 is added to the C. D: D = D U r1.
11. If │C│ > = K then
12. Remove a rule r from C whose support is equal to the present minimum
    support, Set minimum support = lowest support of rules in C
13. while D is non-empty
14. Do
15. Select a rule r with the highest support in D
16. Expand-Left (rule, N, E, min_sup, min_conf)
17. Expand-Right (rule, N, E, min_sup, min_conf)
18. Remove r from D
19. END
```

Mining the Top-k association rules has two difficulties:

1. Top-k rules can't depend on minimum support to prune the search space, however they may be adjusted to mine frequent itemsets with minimum support = 1 to guarantee that all top-k rules can be generated in the generating rules step.
2. Top-k association rules mining algorithm can't utilize the two stages procedure to mine association rules however they would need to be altered to discover top-k rules by generating rules and keeping top-k rules.

4 Experimental Evaluation

For an experimental evaluation of the proposed algorithm (TKAR), we performed several experiments on real-life datasets. We implemented the proposed algorithm on a computer equipped with a Core i5 processor running Windows 7 and 4 GB of main memory running under Java. All datasets (mentioned in the Table 2) were obtained from the Machine Learning Repository[1] (UCI). The Chess and Connect datasets are derivative from their respective game steps, the mushroom dataset contains characteristics of a variety of species of mushrooms, and the Pumsb consists of the population and housing (census) data. Table 3 shows some

[1]http://fimi.ua.ac.be/data/

Table 3 Evaluation for Mushroom Dataset

Min_Conf	Execution time (Milliseconds)				Memory usage (Mega bytes)			
	k = 75	k = 150	k = 600	k = 1500	k = 75	k = 150	k = 600	k = 1500
0.1	6	7	22	35	8.39248	10.28102	12.42887	16.86091
0.2	4	9	22	33	8.61269	9.13397	10.09296	17.32567
0.3	7	10	9	37	9.02098	10.44871	12.13493	17.55789
0.4	7	11	10	38	10.95368	11.74442	13.33350	18.31702
0.5	7	8	6	25	10.10949	8.89583	8.92863	14.42796
0.6	7	10	8	28	11.18904	10.24600	11.28932	24.30187
0.7	8	10	23	36	11.07919	10.73683	13.86193	25.24107
0.8	7	11	14	38	11.14436	11.39779	13.90453	28.53613
0.9	9	6	17	40	11.44678	12.39027	14.01256	34.57773

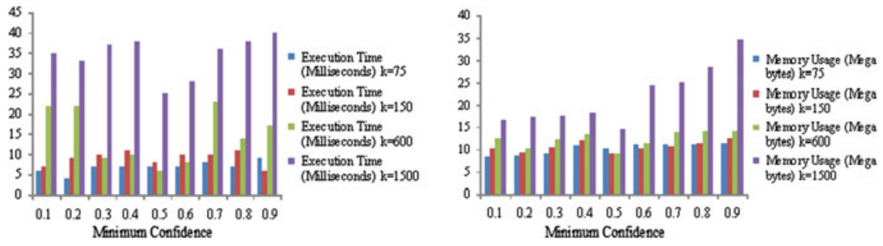

Fig. 4 Detailed results for varying k for the mushroom dataset

characteristics of the real datasets. Typically, these datasets are dense, that is, they generate many frequent itemsets even for very high values of support.

Jeff Schlimmer utilized a database and applied mining techniques with the intention to obtain classifiers which can be used on similar databases. He compiled a mushroom dataset illustrating the records from The Audubon Society Field to North American Mushrooms [6] (1981). The Audubon Society's vision is to preserve and reinstate natural ecosystems. The dataset comprises 8124 instances among them 4208 are classified as edible while 3916 are poisonous and 23 species of gilled mushrooms in the Agaricus and Lepiota families. Each species may recognize as belong to one of the three classes, namely definitely edible, definitely poisonous, or unknown edibility and thus, not recommended. The purpose of choosing the mushroom dataset is to detach edible mushrooms from poisonous ones. It is a classification problem with a target variable of whether mushrooms can be eaten or not. The results generated through TKAR algorithm on mushroom dataset indicates to increase the survival rate when we are in the wild life. If one is capable of distinguishing mushrooms between edible and poisonous by their appearances then we may able to survive in deep mountains by eating mushrooms.

Impact of the k parameter

We have run algorithm by fluctuating estimation of k and considers its values as 75, 150, 600 and 2000 separately. Furthermore we have taken minconf value as 0.1–0.9. As estimation of k increases, both the execution time and memory also increases. This assessment has been appeared underneath in the table and its respective graph. Initially as appeared underneath we consider results of mushroom dataset with various estimation of k. This Table 3 and Fig. 4 indicates memory necessity is directly expanding with estimation of k.

5 Conclusion

The users need to choose the parameter estimations of minconf and minsup in classical algorithm. So amazingly vast number of rules is produced. Due to this the algorithm experience the long execution time and more memory required. We proposed the methodology which produce governs precisely how much user needs.

This proposed algorithm gives preferred execution and performance over others and it is worthwhile to classical algorithms when the user needs to control the number of association rules produced. The top-k association rules mined through the proposed algorithm have the redundancy rules. There is a scope to develop efficient algorithms for mining non-redundant association rules.

References

1. L. Bin, Q. Huayong, S. Yizhen, Realization and application of customer attrition early warning model in security company. TELKOMNIKA Indonesian J. Electr. Eng. **10**(5), 10(5), 1106–1110 (2012)
2. L. Zhou, H. Wang, W. Wang, Parallel implementation of classification algorithms based on cloud computing environment. TELKOMNIKA Indonesian J. Electr. Eng. **10**(5), 10(5), 1087–1092 (2012)
3. A. Ali, M. Elfaki, D. Norhayati, Using Naïve Bayes and bayesian network for prediction of potential problematic cases in tuberculosis. Int. J. Inf. Commun. Technol. (IJ-ICT), **1**(2),1(2), 63–71 (2012)
4. R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in Proceedings of the ACM SIGMOD International Conference on Management of data, vol. 22, no. 2, (ACM, New York, 1993), pp. 207–216
5. R. Agrawal, R. Srikant, Fast algorithms for mining Association Rules in large databases, in Proceedings of the 20th International Conference on Very Large Data Bases (Morgan Kaufmann Publishers Inc., San Francisco, 1994), pp. 487–499
6. D. Jain, S. Gautam, Implementation of apriori algorithm in health care sector: a survey. Int. J. Comput. Sci. Commun. Eng. Nov 2013
7. S. Panwar, Analytical study of frequent Itemset and Utility Mining Approaches. Int. J. Eng. Appl. Technol. 180–187, July 2015. ISSN: 2321-8134
8. Y.L. Cheung, A.W. Fu, Mining frequent itemsets without support threshold: with and without item constraints. IEEE TKDE J. **16**(6), 1052–1069 (2004)
9. T. Le, B. Vo, Bay: An N-list-based algorithm for mining frequent closed patterns. Expert Syst. Appl. **42**(19), 6648–6657 (2015)
10. Q. Huynh-Thi-Le, T. Le, B. Vo, H. Le, H. Bac, An efficient and effective algorithm for mining top-rank-k frequent patterns. Expert Syst. Appl. **42**(1), 156–164 (2015)
11. Z.H. Deng, Fast mining top-rank-k frequent patterns by using node-lists. Expert Syst. Appl. **41**, 1763–1768 (2014)
12. M. Zihayat, A. An, Mining top-k high utility patterns over data streams. Inf. Sci. (2014)
13. K. Chuang, J. Huang, M. Chen, Mining Top-K frequent patterns in the presence of the memory constraint. VLDB J. **17**, 1321–1344 (2008)
14. P. Tzvetkov, X. Yan, J. Han, TSP: Mining top-k closed sequential itemsets. KAIS J. **7**(4), 438–457 (2005)
15. J. Wang, J. Han, Y. Lu, P. Tzvetkov, TFP: an efficient algorithm for mining Top-k frequent closed Itemsets. IEEE TKDE J. **17**(5), 652–664 (2005)
16. J. Han, J. Pei, Mining frequent patterns by pattern-growth: methodology and implications. ACM SIGKDD Explor. Newsl. **2**(2), 14–20 (2000)

17. J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, D. Yang, H-mine: Hyper-Structure mining of frequent Itemsets in large databases, in ICDM, pp. 441–448 (2001)
18. C.W. Wu, B.E. Shie, P. Yu, V.S. Tseng, Mining top-K high utility itemsets, in KDD, pp. 78–86 (2012)