International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

Mining Top-K High Utility Item Sets without Minimum Utility Threshold

Al Zeba & Mrs. N.Satyavathi
[1]PG Scholar, Department of CSE, Vaagdevi College of Engineering, Bollikunta, Warangal, Telangana, Mail ID: al.zebs@gmail.com
[2]Assistant Professor, Department of CSE, Vaagdevi College of Engineering, Bollikunta, Warangal, Telangana, Mail ID:satya15_n@yahoo.co.in

ABSTRACT

High utility item sets mining is an emerging topic in data mining. Here refers discovering all item sets having a utility meeting a user specified minimum utility threshold. According to users it is a difficult to set min_util. If min_util is set too low, too many HUIs will be generated, it may cause the mining process very inefficient. If min_util is set too high, may be there is no HUIs. Here the above issues are addressed by proposing a new method for mining top-k high utility itemset. Here k is mentioned as the number of HUIs to be mined. Two algorithms named TKU (mining Top-K utility itemsets) and TKO(mining Top-K Utility itemsets) are proposed for mining. These algorithms mine the itemsets without setting min_util threshold.

I. INTRODUCTION

Data mining (occasionally called knowledge discovery) is the process of analyzing data from different angles and summarizing it into useful data. Data mining is a tool for analyzing data. It allows users to analyze data from different levels or angles, arrange it, and the relationships among the data are found. Data mining is the process of finding patterns among sufficient of fields in large relational databases.. Several studies have been done to HUI mining [1,2,4,7], it is very difficult for users to choose a minimum utility threshold. According to the value of threshold, the output size can be very small or very large. The choice of the threshold greatly influence the performance of the algorithms. If the threshold is too low, too many HUIs will be generated and it is very difficult for the users to understand the results. A large number of HUIs also causes the mining algorithms to become inefficient. If the algorithms generate more HUIs , It uses more resources also. If the threshold is set too high, no HUI will be generated. To find value for the min_util threshold, users need to try different values by guessing and re-executing the algorithms. This process is very time consuming. To limit the output size and to control the itemsets with the

highest utilities without setting the thresholds, a better solution is to change the task of mining HUIs as mining top-k high utility itemsets. Here the users specify k. Here k is the number of desired itemsets, instead of specifying the minimum utility threshold. Setting k is more easier than setting the threshold because k is the number of itemsets that the users want to find whereas selecting the threshold depends on database characteristics, which are unknown to users. Parameter k is used instead of the min_util threshold,it is very useful for many applications. This concept is used to analyze customer purchase behavior. Top k HUI mining is used to find ,what are the top-k sets of products that contribute the highest profit to the company and how to efficiently found these itemsets without setting the min-util threshold. Top-k HUI mining is essential to many applications,It is not an easy task for developing efficient algorithms for mining such patterns .Two algorithms named TKU and TKO are proposed for mining the complete set of top k HUIs in databases without the need to specify the min_util threshold. The TKU algorithm uses a tree-based structure named UP-Tree, it is used to maintain the information of transactions and utilities of itemsets. TKU inherits useful properties from the TWU model and it consists of two phases. In phase I, potential top-k high utility itemsets are generated. In phase II, top-k HUIs are identified from the set of PKHUIS generated in phase I. The next algorithm is TKO, it uses a list-based structure named utility-list to store the utility information of itemsets in the database. It uses vertical data representation techniques to find top-k HUIs in one phase.

II. RELATED WORK

1. High Utility Itemset Mining High Utility Itemset Mining is a popular concept and many algorithms have been proposed for HUI mining such as two-phase[13], IHUP[2], IIDS[15], UP-Growth[12], and HUI-Miner[14]. These algorithms can be generally classified in two types: Two-phase and one-phase algorithms. The characteristics of two-phase algorithm is that it consist of two phases. In the first phase, they create a set of candidates that are potential high utility itemsets. In the second phase, they calculate the precise utility of each candidate found in the first phase to identify high utility itemsets. Two-phase, IHUP, IIDS, and UP-Growth are two-phase based algorithms. 2. Top-k Pattern Mining Many studies have been proposed to mine diverse kinds of top-k patterns, such as top-k frequent itemsets[3,15], top-k frequent closed itemsets[3], top-k association rules[6], top-k sequential rules[5]. The choice of data structures and look for strategy affect the effectiveness of a top-k mining algorithm in terms of both memory

and execution time. Apriori is a famous algorithm used in data mining The Apriori algorithm is based on the concept that if a subset H appears N times, any other subset H' that contains H will appear N times or less. So, if H doesn't pass the minimum support threshold, neither does H'. There is no need to calculate H', it is discarded apriori. Now here going to show an example of this algorithm. Let's suppose here a client john with transactions [[pen, pencil, book], [pen, book, bag], [pen, bag], [pen, pencil, book]], and a minimum support threshold m of 50 percentage (2 transactions). First step: Count the singletons apply threshold The singletons for john are: pen: 4, pencil: 2, book: 3, bag: 2 All of the single items appear L or more times, so none of them are discarded. Second step: Generate pairs, count them and apply threshold. The pairs created were: pen, pencil, pen, book, pen, bag, pencil, book, pencil, bag, book, bag. Now we proceed to count them and applying the threshold. pen, pencil: 2 pen, book: 3 pen, bag: 2 pencil, book: 2 pencil, bag: 0 book, bag: 1 pencil, bag and book, bag have not passed the threshold, so they are discarded and any other subcombination both of them can generate. The left over pairs are put in a temporary set. Ass = pen, pencil, pen, book, pen, bag, pencil, book Step M: will generate triplets, quadruplets, etc., add up them, apply threshold and remove containing itemsets. We generate triplets from our pairs. Triplets = pen, pencil, book, pen, pencil, bag, pen, book, bag, pencil, book, bag. Now we count them: pen, pencil, book: 2 pen, pencil, bag: 0 pen, book, bag: 1 pencil, book, bag: 0 Only pen, pencil, book has passed the threshold, so now we proceed to add it to Ass, but first, we have to remove the subsets that pen, pencil, book contains. Before adding our left over triplet Ass is looked like this: pen pencil, pen, book, pen, bag, pencil, book. When we add the triplet and remove the subsets that are inside it pen, pencil, pen, book and pencil, book are the ones that should go. Ass now look like pen, pencil, book, pen, bag, and this is the final result. If we had more than 1 triplet after apply the threshold, we proceed to generating the quadruplets, counting them, applying the threshold, adding up them to Ass and removing the subsets that each quadruplet contains.

III. PROPOSED ALGORITHM

1. The TKU Algorithm. Here, proposed an algorithm named TKU for finding top-k HUIs without specify min_util. The Baseline approach of TKU is an seize of UP-Growth[15], a tree-based algorithm for mining HUIs. TKU uses the UP-Tree structure of UP-Growth to manage the information of transactions and top-k HUIs. TKU is worked in three steps. (1) Constructing the UP-Tree, (2) generating potential top-k high utility

itemsets from the UP-Tree, (3) identifying top-k HUIs from the set of PKHUIs. UP-Tree structure.: UP-Tree structure is described in [15]. Each node N of a UP-Tree has five entries: N.name is the item name of N; N.count is the support count of N; N.nu is the node utility of N; N.parent indicates the parent node of N; N.hlink is a node link which may point to a node having the same item name as N.name. The Header table is a structure employed to facilitate the traversal of the UP-Tree. A header table entry contains an item name, an estimated utility value, and a link. The link points to the first node in the UP-Tree having the same item name as the entry. The nodes whose item names are the same can be traversed efficiently by following the links in header table and the node links in the UP-Tree. Construction of UP-Tree: A UP-Tree can be constructed by scanning the original database twice. In the first scan, the transaction utility of each transaction and TWU of the each item are completed. Subsequently, items are inserted into the header table in descending order of their TW use. During the second database scan, transactions are reorganized and then inserted into the UP-Tree. Initially, the tree is created with a Root. When a transaction is retrieved, items in the transaction are sorted in descending order of TWU. A transaction after the above reorganization is called reorganized transaction and its transaction utility is called reorganized transaction utility. After inserting all the reorganized transactions, the construction of the UP-Tree is completed Generating PKHUIs from the UP-Tree : The TKU algorithm uses an internal variable named border minimum utility threshold, which is initially set to 0 and raised dynamically after a sufficient number of itemsets with higher utilities has been captured during the generation of PKHUIs. TKU applies the UP-Growth search procedure to generate PKHUIs. Example 1 Consider that k = 4 and abs_min_util = 0. Let P be the set of all 1-itemsets {{A}:20, {D}:20, {B}:16, {E}:15, {C}:13, G:7, F:5} in D, where the number beside each item-set is its absolute utility. By Lemma 1, {C}, {G}, {F} are un-promising to be the top-4 HUIs. Therefore abs_min_util can be raised to the 4-th highest utility value in P (i.e., 15) and no top-k HUIs will be missed. Identifying top-k HUIs from PKHUIs: After identifying PKHUIs, TKU calculates the utility of PKHUIs by scanning the original database once, to identify the top-k HUIs. 2. The TKO Algorithm. It can find top-k HUIs in only one phase. It uses the basic search procedure of HUI-miner and its utility-list[14] structure. Whenever an itemset is generated by TKO, utility of the generated itemset is calculated by its utility-list without scanning the original database. Construction of Utility-list structure: Utility list is described in [10] ,in the case of TKO algorithms,

each item related with a utility-list. The utility-list of items are generally called initial utility lists. These are constructed by scanning the database twice. In the first scan, the TWU and utility values of items are calculated. During the second scan, items in each transaction are sorted in order TWU values and the utility list of each item is constructed.

IV. PSEUDO CODE

Algorithms and memory usage of the algorithms on Retail and Chain store. TKO generally uses less memory than TKU , this is because TKU is a two-phase algorithm. When they could not effectively raise the minimum utility thresholds, they may consider too many candidates and local UP-Trees during the mining process, which causes the algorithm to consume much more memory than TKO. Here compares with another algorithm REPT also. The memory consumption of REPT(N=5,000) is higher than that of TKU. This is because REPT maintains not only a global UP-Tree in mem-ory but also a RSD matrix. When there are many promising items and N is set too large for REPT, the RSD matrix could be very large and make REPT uses more memory.

V. CONCLUSION AND FUTURE WORK

Here studied the problem of top-k high utility itemsets mining, where k is the number of high utility itemsets to be mined. Two efficient algorithms TKU (mining Top-K Utility itemsets) and TKO (min-ing Top-K utility itemsets in One phase) are proposed here for mining such itemsets without using minimum utility thresholds concept. TKU is the first two-phase algorithm for mining top-k high utility itemsets. On the other hand, TKO is the first one-phase algorithm developed for top-k HUI mining. Empirical evaluations on different types of real and synthetic datasets which show the proposed algorithms have good scalability on large datasets and the performance of the proposed algorithms is close to the optimal case of two-phase and one-phase utility mining algorithms [14]. Although here proposed a new framework for top-k HUI mining.

REFERENCES

1 R. Aggrawal, " Efficient Algorithms for Mining Association Rules", ‖ in Proc. of Int'l Conf. on Very Large Data Bases, pp. 487-499, 1995.

2 C. Ahmed, S. Tanbeer, B. Jeong , "Efficient Tree Structures for High-utility Pattern Mining in Databases", ‖ IEEE Transactions on Knowledge and Data Engineering , pp. 1708-1721, 2009.

3 K. Chuang, J. Huang and M. Chan, "Mining Top-K Frequent Patterns primarily in the Presence of the Memory Constraint", ‖ The Journal, Vol. 17, pp. 1321-1344, 2009.

4 R. Chan, Q. Yang and Y. Shan, "Mining High-utility Itemsets", ‖ in Proc. of IEEE Int'l Conf. on Data Mining, pp. 19-26, 2003.

5 P. Fournier-Viger, V. S Tseng, "Mining Top-K Sequential Rules", ‖ in Proc. of Int'l Conf. on Advanced Data Mining and Applications, pp. 180-194, 2011.

6 P. Fournier-Viger, C. Wu, V. S. Tsenge, "Mining Top-K Association Rules", ‖ in Proc. of Int'l Conf. on Canadian conference on Advances in Artificial Intelligence, pp. 61–73, 2013.

7 P. Fournier-Viger, C. Wu, V. S. Tseng, "Novel Concise Representations of High Utility Itemsets Using Generator Patterns", Int'l. Conf. on Advanced Data Mining and Applications and Lecture Notes in Computer Science, Vol. 8933, pp. 30-43, 2014.

8 J. Han, J. Pei and Y. Yin, "Mining Frequent Patterns without Candidate Generation", ‖ in Proc. of ACM SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.

9 J. Han, J. Wang, Y. Lu and P. Tzvetkov," Mining Top-K Frequent Closed Patterns without Minimum Support", ‖ in Proc. of IEEE Int'l Conf. on Data Mining, pp. 211-218, 2002.

10 S. Krishnamoorthy, "Pruning Strategies for Mining High Utility Itemsets", ‖ Expert Systems with Applications, Vol. 42(5), pp. 2371-2381, 2015.

11 C. Lin, T. Hong, G. Lan, J. Wong and W. Lin, "Efficient Updating of Discovered High-utility Itemsets for Transaction Deletion in Dynamic Databases", ‖ Advanced Engineering Informatics, Vol. 29(1), pp. 16-27, 2015.

12 G. Lan, T. Hong, V. S. Tseng and S. Wang, "Applying the Maximum Utility Meassure in High Utility Sequential Pattern Mining", ‖ Expert Systems with Applications, Vol. 41(11), pp. 5071-5081, 2014.

13 Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining Algorithm", ‖ in Proc. of the Utility-Based Data Mining Workshop, pp. 90-99, 2005.

14 M. Liu and J. Qu, "Mining High Utility Itemsets without Candidate Generation", ‖ in Proc. of ACM Int'l Conf. on Information and Knowledge Management, pp. 55-64, 2012.

15 J. Liu, K. Wang and B. Fung, "Direct Discovery of High Utility Itemsets without Candidate Generation", ‖ in Proc. of IEEE Int'l Conf. on Data Mining, pp. 984-989 , 2012.