INTERNATIONAL JOURNAL OF MERGING TECHNOLOGY AND ADVANCED RESEARCH IN COMPUTING

IJMTARC – VOLUME – V – ISSUE - 17 – MAR 2017

ISSN: 2320-1363

Mining Top-K High Utility Item sets

E.RAJESH *, VANAM GOPINATH **

*PG SCHOLAR,DEPEARTMENT OF CSE, SLC'S INSTITUTE OF ENGINEERING AND TECHNOLOGY,HYDERABAD

** ASSOSCIATE PROFESSOR, DEPEARTMENT OF CSE, SLC'S INSTITUTE OF ENGINEERING AND TECHNOLOGY,HYDERABAD

ABSTRACT

Extraction of high utility object sets is an emerging topic in data mining. Here refers to the discovery of all itemsets having a utility that meets a minimum utility threshold specified by the user. According to the users, it is difficult to define min_util. If min_util is set too low, too much HUI will be generated, this can make the data mining process very inefficient. If min_util is too high, there may be no HUI. In this case, the above problems are solved by providing a new method to extract the item set from the top-k high utility. Here, k is mentioned as the number of HUIs to extract. Two algorithms named TKU (extraction of Top-K utility itemsets) and TKO (extraction of Top-K Utility itemsets) are offered for extraction. These algorithms exploit sets of elements without defining a min_util threshold.

I. INTRODUCTION

Data mining (sometimes called knowledge discovery) involves analyzing data from different angles and summarizing it into useful data. Data mining is a data analysis tool. It allows users to analyze data from different levels or angles, organize them, and relationships between data are found. Data mining is the process of finding patterns among enough fields in large relational databases. Several studies have been done on HUI extraction [1,2,4,7], making it very difficult for users to choose a minimum utility threshold. Depending on the threshold value, the output size can be very small or very large. The choice of the threshold greatly influences the performance of the algorithms. If the threshold is too low, too much HUI will be generated and it is very difficult for users to understand the results. A large number of HUIs also result in inefficient data mining algorithms. If the algorithms generate more HUI, it also uses more resources. If the threshold is too high, no HUI will be generated. To find the value of the min_util threshold, users must try different values by guessing and re-executing the algorithms. This process is time consuming. To limit the output size and control sets of items with the highest utilities without setting thresholds, a better solution is to modify the HUI extraction task by extracting sets of high level utility items. superior. Here, users specify k. Here, k is the number of itemsets desired, instead of specifying the minimum utility threshold. The definition of k is simpler than defining the threshold, because k is the number of element sets that users want to find, while the threshold selection depends on the characteristics of the database, unknown to users. The k parameter is used instead of the min_util threshold, it is very useful for many applications. This concept is used to analyze the buying behavior of customers. Back to top k HUI Mineral Exploration identifies the most successful product sets that contribute the most to the bottom line and how to efficiently find these sets of items without defining the min-util threshold. Top-k HUI mining is essential to many applications, It is not an easy task to develop efficient algorithms for extracting such models. Two algorithms named TKU and TKO are proposed for the extraction of all HUI k in the databases without having to specify the min_util threshold. The TKU algorithm uses a tree structure named UP-Tree, which is used to maintain information about transactions and utilities in item sets. TKU inherits the useful properties of the TWU model and consists of two phases. During phase I, sets of high utility items of type k top-k are generated. In phase II, the HUI top-k are identified from the set of PKHUIS generated in phase I. The following 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 the best performing HUIs in a 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,

IJMTARC – VOLUME – V – ISSUE - 17 – MAR 2017 ISSN: 2320-1363

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 to find HUI top-k without specifying min_util. TKU's basic approach is a seizure of UP-Growth [15], a tree-based algorithm for mine HUIs. TKU uses the UP-Tree structure of UP-Growth to manage transaction information and the best performing HUIs. TKU is worked in three stages. (1) Build the UP-Tree, (2) generate itemsets of the high-k potential utility of the UP tree, (3) identify the HUIs of the k-top-k of the set of PKHUI . UP-Tree Structure: The UP-Tree structure is described in [15]. Each node N of a UP tree has five entries: N.name is the name of the N element; N.count is the support number of N; N.nu is the N knot utility; N.parent indicates the parent node of N; N.hlink is a node link that can point to a node with the same element name as N.name. The Header table is a structure used to facilitate the crossing of the UP-Tree. A header table entry contains an element name, an estimated utility value, and a link. The link points to the first node of the UP tree with the same element name as the entry. Nodes with identical item names can be browsed efficiently by following the links in the header table and the node links in the UP tree. UP-Tree Construction: An UP-Tree can be built by scanning the original database twice. During the first scan, the transaction utility for each transaction and the TWU for each item are completed. Subsequently, the elements are inserted in the header table in descending order of their use TW. In the second

database scan, transactions are rearranged and inserted into the UP tree. Initially, the tree is created with a root. When a transaction is retrieved, the elements of the transaction are sorted in descending order of TWU. A transaction after the reorganization above is called a reorganized transaction and its transaction utility is called a reorganized transaction utility. After inserting all reorganized transactions, construction of the UP-Tree is complete. Generate PKHUI from the UP Tree: The TKU algorithm uses an internal variable named threshold minimum utility limit, initially set to 0 and raised dynamically after a sufficient number of itemsets with higher utilities was captured during generation. PKHUI. TKU applies the UP-Growth search procedure to generate PKHUI. Example 1 Consider that k = 4 and abs_min_util = 0. Let P be the set of all one-item items {{A}: 20, {D}: 20, {B}: 16, {E}: 15, { C}: 13, G: 7, F: 5} in D, where the number next to each set of items is its absolute utility. By lemma 1, {C}, {G}, {F} do not promise to be the best 4 HUI. Therefore, abs_min_util can be raised to the 4th highest utility value in P (i.e., 15) and no higher level HUI will be missed. Identification of PKHUI High-k HUIs: After identifying PKHUI, TKU calculates the utility of PKHUI by once-analyzing the original database to identify higher-level HUIs. 2. The TKO algorithm. He can find the high-k HUIs in one phase. It uses HUI-miner's basic search procedure and its utility list structure [14]. Whenever an itemet is generated by TKO, the utility of the generated itemset is calculated by its list of utilities without scanning the original database. Construction of the list of utilities: The list of utilities is described in [10], in the case of TKO algorithms, each element being associated with a list of utilities. The list of utilities is usually called initial utility lists. These are built by scanning the database twice. During the first scan, the TWU values and utility values of the elements are calculated. During the second analysis, the items of each transaction are sorted according to TWU values and the utility list of each item is constructed

.IV. PSEUDO CODE

algorithms and memory usage of the algorithms on Retail and Chainstore. 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, we have studied the problem of extracting sets of high-useful objects k-top, where k is the number of high utility object sets to extract. Two efficient algorithms TKU (extraction of Top-K Utility itemsets) and TKO (top-K utility twinks in one phase) are proposed here to extract such itemsets without using the concept of minimum utility thresholds. TKU is the first two-phase algorithm for extracting sets of utility objects from top-k. On the other hand, TKO is the first one-phase algorithm developed for high-k HUI extraction. Empirical evaluations of different types of real and synthetic data sets that show that the proposed algorithms have good extensibility on large data sets and the performance of the proposed algorithms are close to the optimal case of utility extraction algorithms. two phases and one phase [14]. Although proposed here a new framework for HUI top-k extraction.

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

IJMTARC – VOLUME – V – ISSUE - 17 – MAR 2017 ISSN: 2320-1363

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 Genera-tion", ‖ 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

.