Imperial Journal of Interdisciplinary Research (IJIR)
Vol-3, Issue-6, 2017
ISSN: 2454-1362, http://www.onlinejournal.in

Mining High Utility Itemsets Using Efficient Algorithm

Akash Jagtap, Ajay Kangare, Nilesh Patil, Pratik Thorat.
Department Of Computer Engineering, Zeal College OF Engineering And Research, Pune, Maharashtra, India

Abstract : *High utility itemsets (HUIs) mining is becoming an emerging topic in data mining, which refers in discovering all itemsets having a utility meeting a user-specified minimum utility threshold min_util. However, setting min_util appropriately is a difficult problem for users. Generally speaking, finding an appropriate minimum utility threshold becomes tedious process for users by trial and error method. If min_util is set too low, too many HUIs will be generated, which may in turn cause the mining process to be very inefficient. On the other hand, if min_util is set too high, it is likely that HUIs will not be found. In this paper, we address the above issues by proposing a new framework for top-k high utility itemset mining, where k is the desired number of HUIs to be mined. An efficient algorithm named TKU (miningTop-K Utility itemsets) is proposed for mining such itemsets without the need to set min_util. We provide a structural comparison of the algorithms with discussions on their advantages and limitations. Empirical evaluations on both real and synthetic datasets show that the performance of the proposed algorithms is close to that of the optimal case of state of the art utility mining algorithms.*

1. Introduction

It Frequent itemset mining (FIM) is a fundamental research topic in data mining. Altough, the traditional FIM may discover a large amount of frequent but low-value itemsets and may lose the information on valuable itemsets having low selling frequencies. Hence, it fails to satisfy the requirement of users who wish to discover itemsets with high utilities such as high profits. To address these issues, utility mining emerges as an important topic in data mining and has received extensive attention in recent years. In utility mining, each item is associated with a utility (e.g. unit profit) and an occurrence count in each transaction (e.g. quantity). The utility of an itemset depicts its importance, which can be measured in terms of weight, value, quantity or other information depending on the user specification. An itemset is called high utility itemset (HUI) if its utility is not less than a user-specified minimum utility threshold min_util. HUI mining is essential to many applications such as streaming analysis market analysis, mobile computing and biomedicine.

However, efficiently mining HUIs in databases is not an simple task because the downward closure property, which is used in FIM does not hold for the utility of itemsets. In other words, pruning search space for HUI mining is difficult because a superset of a low utility itemset can be high utility. To tackle this problem, the concept of transaction weighted utilization (TWU) model is introduced which facilitates the performance of the mining task. In this model an itemset is called high transaction-weighted utilization itemset (HTWUI) if its TWU is no less than min_util, where the TWU of an itemset represents an upper bound on its utility.

Because of this a HUI must be a HTWUI and all the HUIs must be included in the complete set of HTWUIs. A classical TWU model-based algorithm mainly consists of two phases. In the first phase, which is called phase I, the complete set of HTWUIs are found. In the second phase, called which is called phase II, all HUIs are obtained by calculating the exact utilities of HTWUIs with one database scan.

Although there lot many studies which have been devoted to HUI mining but it becomes difficult for users to choose an appropriate minimum utility threshold in practice. Depending on the threshold, the output size can be very small or very large. Besides, the threshold choice greatly influences the performance of the algorithms. If the threshold is set too low, too many HUIs will be presented to the users and then it becomes difficult for the users to comprehend the results. A verylarge number of HUIs also causes the mining algorithms to become inefficient or it may even run out of memory, because the more HUIs the algorithms generate, the more resources they consume. On the contrary, if the threshold is set too high, no HUI will be found. To find an appropriate value for the min_util threshold, users need to try different thresholds by

using try and error and re-executing the algorithms over and over until being satisfied with the results. This process is inconvenient as well as time-consuming. To very precisely control the output size and discover the new itemsets with the highest utilities without setting the thresholds, a promising solution is to redefine the task of mining HUIs as mining top-k high utility itemsets (top-k HUIs). The idea behind this is to let the users specify k, i.e., the number of desired itemsets, instead of specifying the minimum utility threshold. Setting k is more intuitive than setting the threshold because k represents the number of itemsets that the users want to find whereas choosing the threshold depends primarily on database characteristics, which are often unknown to users. Using a parameter k instead of the min_util threshold is very compatible for many applications. For example, to analyze customer purchase behavior, top-k HUI mining serves as a promising solution for users who desire to know "What are the top-k sets of products that contribute the highest profits to the company?" and "How to efficiently find these itemsets without setting the min_util threshold?".

Although top-k HUI mining is essential to many applications, developing efficient algorithms for mining such patterns is not an easy task. It poses four major challenges as discussed below.

Firstly, the utilities of the itemsets are neither monotone nor ant monotone. In other words, one can say that the utility of an itemset may be equal to, higher or lower than that of its supersets and subsets. Hence, many techniques developed in top-k frequent pattern mining which rely on anti-monotonicity toprune the search space cannot be directly applied to top-k high utility itemset mining.

Secondly the challenge is how to incorporate the concept of top-k pattern mining with the TWU model. Even tough theTWU model is widely used in utility mining; it is difficult to adapt this model to top-k HUI mining because the exact utilitiesof itemsets are unknown in phase I. When a HTWUI is generated in phase I, we cannot guarantee that its utility is higher than other HTWUIs and that it is a top-k HUI before performing phase II. To guarantee that all the top-k HUIs can be captured in the set of HTWUIs, a naive approach is to run the algorithm with min_util =0. However, this approach may face the problem of a very large search space.

The third challenge is that the min_util threshold is not given in advance in top-k HUI mining. In traditional HUI mining, the search space can be efficiently pruned by the algorithms by using a given min_util threshold. However, in the scenario of top-k HUI mining, no min_util threshold is provided in advance. Therefore, the minimum utility threshold is initially set to 0 and the designed algorithm has to gradually raise the threshold to prune the search space.

Such a threshold comes under internal parameter of the designed algorithm and is called the border minimum utility threshold min_utilBorder included in this paper. It is different from the external parameter min_util that is given by users in advance. If an algorithm cannot raise the min_utilBorder threshold effectively and efficiently, it will produce too many intermediate low utility itemsets during the process of mining, which may degrade its performance in terms of memory usage and execution time. Yet, the challenge is to design effective strategies that can raise the min_util threshold as high as possible and as early as possible, and further reduce as many as possible the number of candidates and intermediate low utility itemsets produced in the mining process.

Lastly the challenge is how to effectively raise the min_utilBorder threshold without missing any top-k HUIs. A good algorithm is one which can effectively raise the threshold during the mining process. However, if an incorrect method for raising the threshold is implemented, it may result in some top-k HUIs which are being pruned. Thus, now the question is how to raise the threshold efficiently and effectively without missing any top-k HUI which is acrucial challenge for this work.

2. Related Works

In this section we introduce related works about top-k high utility itemset mining, which includes high utility itemset mining, top-k frequent pattern mining and top-k high utility itemset mining. In recent years, high utility itemset mining has received lots of attention and many efficient algorithms have been proposed, such as Two-Phase, IHUP, IIDS, UPGrowth, d2HUP and HUI-Miner. These are the algorithms which can be generally categorized into two types: two-phase and one-phase algorithms. The main characteristic of two-phase algorithms is that they consist of two phases. In first phase, they generate a set of candidates which are potential high utility itemsets. In second phase, the exact utility of each candidate is calculated found in the first phase to identify high utility itemsets. Two-Phase, IHUP, IIDS and UP-Growth are the two-phase based algorithms. UPGrowth is one of the state of the art two-phase algorithms, which incorporates four effective strategies DGU, DGN, DLU and DLN for pruning candidates in the first phase. The main

characteristic of one-phase algorithms is they discover high utility itemsets using only one phase and produce no candidates. d2HUP and HUI-Miner are one-phase algorithms. d2HUP transforms a horizontal database into a tree like structure called CAUL and adopts a pattern-growth strategy to directly discover high utility itemsets in databases. HUI-Miner considers a database of vertical format and transforms it into utility-lists. The utility-list structure used in HUI-Miner allows directly computing the utility of generated itemsets in main memory without scanning of the original database. Although the above studies may perform well in some applications, they are not developed for top-k high utility itemset mining and still suffer from the problem of setting appropriate thresholds.

Top-K Pattern Mining

There are many studies that have been proposed to mine different kinds of top-k patterns, such as top-k frequent itemsets top-k frequent closed itemsets, top-k closed sequential patterns, top-k association rules, top-k sequential rules, top-k correlation patterns and top-k cosine similarity interesting pairs. The thing that distinguishes each top-k pattern mining algorithm is the type of patterns discovered, as well as the data structures and search strategies that are employed. For example, some algorithms use a rule expansion strategy for finding different patterns, while others rely on a pattern-growth search using structures such as FP-Tre. The choice of data structures and search strategy may affect the efficiency of a top-k pattern mining algorithm in terms of both memory and execution time. However, the above algorithms discover top-k patterns according to traditional measures instead of the utility measure. As they may miss patterns yielding high utility.

Top-K High Utility Pattern Mining

The task of top-k high utility pattern mining was firstly introduced by Chan et al. But the definition of high utility itemset used in their study is different from the one which is used in this work. Chan et al.'s study has mainly considered utilities of various items, but quantitative values of items in transactions were not taken into consideration. We have defined the task of top-k high utility itemset mining by considering both quantities and profits of items.

The TKU Algorithm

In this section, we propose an efficient algorithm named TKU (mining Top-k Utility itemsets) for discovering top-k HUIs without specifying min_util.

The Baseline Approach Tkubase

The baseline approach TKUBase is mainly an extension of UPGrowth a tree-based algorithm for mining HUIs. TKUBase adopts the UP-Tree structure of UP-Growth to maintain the information of transactions and top-k HUIs. TKUBase is executed in three steps

1. Construction of the UP-Tree.
2. Generation of potential top-k high utility itemsets (PKHUIs) from the UP-Tree.
3. Identification of top-k HUIs from theset of PKHUIs.

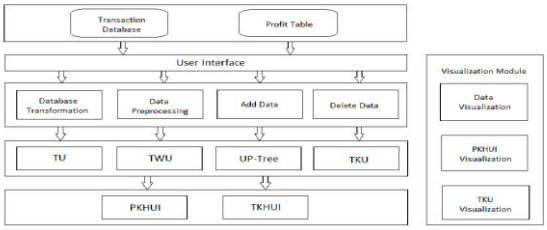

Figure: Architecture

Up-Tree Structure

Next, we introduce the UP-Tree structure. 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 made to facilitate the traversal of the UP-Tree.

A header table entry will contain the 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 done before. The nodes whose item names are the same will 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 two times. In the first scan, the transaction utility of each transaction and TWU of each item are computed. Thus, items and their TWUs are obtained. Simultaneously, items are inserted into the header table in descending order of their TWUs. In the second database scan, transactions are reorganized and then inserted into the UP-Tree. At the starting the tree is created with a root R. 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 (RTU).

Generating Pkhuis From The Up-Tree

The TKUBase algorithm uses an internal variable named border minimum utility threshold (denoted as min_utilBorder) which is initially holds the value 0 and raised dynamically after a sufficient number of itemsets with higher utilities has been captured during the generation of PKHUIs.

Identifying Top-K Huis From Pkhuis

After identifying PKHUIs, TKUBase calculates the utility of PKHUIs by scanning the original database, to identifies the top-k HUIs. This process is very similar to that of Phase II. However, in the previous work, all the candidates were considered. In this work, we only consider a candidate itemset X if its estimated utility value reached after phase I is no less than min_utilBorder, i.e., min{ESTU(X), MAU(X)} min_utilBorder.

3. Conclusion

As we have studied the problem of top-k high utility itemsets mining, where k is the desired number of high utility itemsets to be mined. TKU is the first two-phase algorithm for mining top-k high utility itemsets, which incorporates five strategies PE, NU, MD, MC and SE to effectively raise the border minimum utility thresholds and further prune the search space. On the other hand, we have proposed a new framework for top-k HUI mining, it has not yet been incorporated with other utility mining tasks to discover different types of top-k high utility patterns such as top-k high utility episodes, top-k closed high utility itemsets, top-k high utility web access patterns and top-k mobile high utility sequential patterns. These leave wide rooms for exploration as future work.

4. References

[1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” in Proc. Int. Conf. Very Large Data Bases, 1994, pp. 487–499.

[2] C. Ahmed, S. Tanbeer, B. Jeong, and Y. Lee, “Efficient tree structures for high-utility pattern mining in incremental databases,” IEEE Trans. Knowl. Data Eng., vol. 21, no. 12, pp. 1708–1721, Dec. 2009.

[3] K. Chuang, J. Huang, and M. Chen, “Mining top-k frequent patterns in the presence of the memory constraint,” VLDB J., vol. 17, pp. 1321–1344, 2008.

[4] R. Chan, Q. Yang, and Y. Shen, “Mining high-utility itemsets,” in Proc. IEEE Int. Conf. Data Mining, 2003, pp. 19–26.