Eleyon
SOUTH ASIAN JOURNAL OF ENGINEERING AND TECHNOLOGY

Full Length Article

CONFABULATION BASE INFREQUENT WEIGHTED ITEMSET MINING USING FREQUENT PATTERN GROWTH

[a] Dr. E. Baby Anitha , [a] N. S. Nithiya, [b] M. Udhayapriya , [b] R. Vigneshwaran , [b] V. RA. Vimal , [b] P. Naveen kumar

[a]Associate Professor, Department of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchengode.
[a]Associate Professor, Department of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchengode.
[b]Final Year Student, Department of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchengode.
[b]Final Year Student, Department of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchengode.
[b]Final Year Student, Department of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchengode.
[b]Final Year Student, Department of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchengode.

***Corresponding Author**
Dr. E. Baby Anitha

ABSTRACT: High utility itemset mining (HUIM) has emerged as an important research topic in data mining, with applications to retail-market data analysis, stock market prediction, and recommender systems, etc. However there are very few empirical studies that systematically compare the performance of state-of-the-art HUIM algorithms. In this paper, we present an experimental evaluation on major HUIM algorithms, using real world and synthetic datasets to evaluate their performance. Our experiments show that EFIM and d2HUP are generally the top two performers in running time, while EFIM also consumes the least memory in most cases. In order to compare these two algorithms in depth, we use another synthetic datasets with varying parameters so as to study the influence of the related parameters, in particular the number of transactions, the number of distinct items and average transaction length, on the running time and memory consumption of EFIM and d2HUP. In this work, we demonstrate that, d2HUP is more efficient than EFIM under low minimum utility values and with large sparse datasets, in terms of running time; although EFIM is the fastest in dense real datasets, it is among the slowest algorithms in sparse datasets. Suggest that, when a dataset is very sparse or the average transaction length is large, and running time is favoured over memory consumption, d2HUP should be chosen. Finally, compare d2HUP and EFIM with two newest algorithms, HUI - Miner and ULB-Miner, and find these two algorithms have moderate performance. This work has reference value for researchers and practitioners when choosing the most appropriate HUIM algorithm for their specific applications.

1 Introduction

Mining frequent itemsets from a transaction database refers to the discovery of the itemsets which frequently appear together in the transactions. The main objective of Utility Mining is to identify the itemsets with highest utilities above a user-specified threshold, by considering profit, quantity, cost or other user preferences. If the sHUPport of an itemset exceeds a user-specified minimum sHUPport threshold, the itemset is considered as frequent. Most frequent itemset mining algorithms employ the downward closure property of itemsets. However, the unit profits and purchased quantities of items are not considered in the framework of frequent itemset mining [1][2]. The basic meaning of utility is the interestedness/ importance/profitability of items to the users. The utility of items in a transaction database consists of two aspects:

(1) the importance of items of different transaction is called external utility, and (2) the importance of the items in the transaction, which is called internal utility. The utility of an itemset is defined as the external utility multiplied by the internal utility. An itemset is called a high utility *itemset* if its utility is greater than a user specified threshold; otherwise, the itemset is called a low utility itemset. Mining high utility itemsets from databases refers to finding the itemsets with high profits and it is not an easy task since downward closure property.In other words, pruning search space for high utility itemset mining is hard because a sHUPerset of a low utility itemset may be a high utility itemset. A simple method to address this problem is to enumerate all itemsets from databases by the principle of exhaustion. Obviously, this method couldn't tolerate the problems of a search space, especially when databases contain lots of long transactions or a low minimum utility threshold is set. Recently proposed compact tree structure, viz., HUP-Tree, maintains the information of transactions and itemsets, facilitate the mining

performance and avoid scanning original database repeatedly.

II RELATED WORK

A number of traditional ARM algorithms and optimizations have been proposed. One of the well-known algorithms is HUIM algorithm, which is the pioneer for efficiently mining association rules from large databases. It's widely recognized that FP-Growth achieves a better performance than HUIM Algorithm since it finds frequent itemsets without generating any candidate itemset and it scans database just twice. There are also many studies that have developed different weighting functions for weighted pattern mining. Mengchi Liu

proposed an algorithm,[3][5] called HUI-Miner (High Utility Itemset Miner), for high utility itemset mining.HUI-Miner uses a structure, called utility-list, to store the utility information of an itemset and the heuristic information for pruning the search space of HUI-Miner. By avoiding the costly generation and utility computation of numerous candidate itemsets, HUI-Miner can efficiently mine high utility itemsets from the utility lists constructed from a mined database. Although two-phase algorithm reduces search space by using TWDC property, it still generates too many candidates to obtain HTWUIs and requires multiple database scans. To overcome this problem, Li et al. proposed an isolated items discarding strategy (IIDS) to reduce the number of candidates. By pruning isolated items during levelwise search, the number of candidate itemsets for HTWUIs in phase I can be reduced. However, this algorithm still scans database for several times and uses a candidate generation-and-test scheme to find high utility itemsets. To efficiently generate HTWUIs in phase I and avoid scanning database too many times, Ahmed et al.proposed a tree - based algorithm, named IHUP. A tree based structure called IHUP-Tree is used to maintain the information about itemsets and their utilities. Each node of an IHUP-Tree consists of an item name, a TWU value and a sHUPport count. [4][5]. IHUP algorithm has three steps: 1) construction of IHUP-Tree, 2) generation of HTWUIs, and 3) identification of high utility itemsets. In step 1, items in transactions are rearranged in a fixed order such as lexicographic order, sHUPport descending order or TWU descending order. Then the rearranged transactions are inserted into an IHUP-Tree. In step 2, HTWUIs are generated from the IHUP-Tree by applying FP-Growth. Thus, HTWUIs in phase I can be found without generating any candidate for HTWUIs. In step 3, high utility itemsets and their utilities are ident field from the set of HTWUIs by scanning the original database once. Although IHUP achieves a better performance than IIDS and Two-Phase, it still produces too many HTWUIs in phase I. Note that IHUP and Two-Phase produce the same number of HTWUIs in phase I since they both use TWU framework to overestimate itemsets utilities. However, this framework may produce too many HTWUIs in phase I since the overestimated utility calculated by TWU is too large. Moreover, the number of HTWUIs in phase I also affects the performance of phase II since the more HTWUIs the algorithm generates in phase I, the more execution time for identifying high utility itemsets it requires in phase II.

III PROBLEM STATEMENT

In the literature we have studied the different methods proposed for high utility itemset mining from large datasets. But all this methods frequently generate a huge set of PHUIs and their mining performance is degraded consequently.[2] Further in case of long transactions in dataset or low thresholds are set, then this condition may become worst. The huge number of PHUIs forms a challenging problem to the mining performance since the more PHUIs the algorithm generates, the higher processing time it consumes. Thus to overcome this challenges the efficient algorithms presented recently in. These methods in outperform the state-of-the-art algorithms almost in all cases on both real and synthetic data set.[4] However this approach in is still needs to be improved in case of less memory based systems.

IV EXISTING SYSTEM

The framework of the existing methods consists of three steps: 1) Scan the database twice to construct a global HUP Tree with the first two strategies 2) recursively generate PHUIs from global HUP -Tree and local HUP-Trees by HUP-Growth with the third and fourth strategies or by HUP-Growth+ with the last two strategies and 3) identify actual high utility item sets from the set of PHUIs. To distinguish the patterns found by our methods from HTWUIs since our methods are not based on traditional TWU model. By our effective strategies, the set of PHUIs will become much smaller than the set of HTWUIs [6]. After constructing a global HUP-Tree, a basic method for generating PHUIs is to mine HUP-Tree by FP-Growth. Thus, we propose an algorithm HUP-Growth by pushing two more strategies into the framework of FP-Growth. By the strategies, overestimated utilities of item sets can be decreased and thus the number of PHUIs can be further reduced. HUP-Growth achieves better performance than FP-Growth by using DLU and DLN to decrease overestimated utilities of item sets.

V PROPOSED SYSTEM

The goal of utility mining is to generate all the high utility itemsets whose utility values are beyond a user specified threshold in a transaction.

A.HUIM Growth

The HUIM-Growth is one of the efficient algorithms to generate high utility itemsets depending on construction of a global HUIM-Tree. In phase I, the framework of HUIM-Tree follows three steps: (i).

Construction of HUIM-Tree. (ii). Generate PHUIs from HUIM-Tree. (iii). Identify high utility itemsets using PHUI. The construction of global HUIM-Tree is follows, (i). Discarding global unpromising items (i.e., DGU strategy) is to eliminate the low utility items and their utilities from the transaction utilities. (ii). Discarding global node utilities (i.e., DGN strategy) during global HUP-Tree construction. By DGN strategy, node utilities which are nearer to HUP-Tree root node are effectively reduced. The PHUI is similar to TWU, which compute all itemsets utility with the help of estimated utility. Finally, identify high utility itemsets (not less than min_sHUP) from PHUIs values. The global HUP-Tree contains many sub paths. Each path is considered from bottom node of header table. This path is named as conditional pattern base (CPB).

Disadvantages

It requires multiple database scans.It Generate multiple candidate Itemset.Other Algorithm like HUIM treats all item with same importance or profit. It consumes more memory space and performs badly with long pattern dataset. These methods are further needs to be improved over their limitations presented below:

(1) Performance of this methods needs to be investigated in low memory based systems for mining high utility itemsets from large transactional datasets and hence needs to address further as well. (2)These proposed methods cannot overcome the screenings as well as overhead of null transactions; hence, performance degrades drastically.

B. HUP Growth+

Although DGU and DGN strategies are efficiently reduce the number of candidates in Phase 1(i.e., global HUP-Tree). But they cannot be applied during the construction of the local HUP -Tree (Phase-2). Instead use, DLU strategy (Discarding local unpromising items) to discarding utilities of low utility items from path utilities of the paths and DLN strategy (Discarding local node utilities) to discarding item utilities of descendant nodes during the local HUP-Tree construction. Even though, still the algorithm facing some performance issues in phase-2. To overcome this, maximum transaction weight utilizations (MTWU) are computed from all the items and considering multiple of min_sHUP as a user specified threshold value as shown in algorithm. By this modification, performance will increase compare with existing HUP-Tree construction also improves the performance of HUP-growth algorithm. An improved utility pattern growth is abbreviated as IHUPG.

Advantages

It scan the database just twice. It is easy to implement.It reduces unnecessary calculation when database is HUPdated, and when user specified minimum threshold is changed. It requires less memory space and less execution time.

C. HUP + Algorithm

Input: Transaction database D, user specified threshold.

Output: high utility itemsets.

Begin

1. Load dataset contains number transactions Td ∈ D

2. Determine transaction utility of Td in D and TWU of itemset (X)

3. Compute min_sHUP (MTWU * user specified threshold)

4. If (TWU(X) ≤ min_sHUP) then Remove Items from transaction database

5. Else insert into header table H and to keep the items in the descending order.

6. Repeat step 4 & 5 until end of the D.

7. Insert Td into global HUP-Tree.

8. Apply DGU and DGN strategies on global HUP- tree.

9. Re-construct the HUP-Tree

10. For each item ai in H do

11. Generate a PHUI Y= X U ai

12. Estimate utility of Y is set as ai's utility value in H

13. Put local promising items in Y-CPB into

H

14. Apply strategy DLU to reduce path utilities of the paths

15. Apply strategy DLN and insert paths into

Td

16. If Td ≠ null then call for loop

End for End

D. Application

Rare itemsets provide useful information in different decision-making domains such as business transactions, medical, security, fraudulent transactions and retail communities. For example, in a

sHUPermarket, customers purchase microwave ovens or frying pans rarely as compared to bread, washing powder, soap. But the former transactions yield more profit for the sHUPermarket. Similarly, the high-profit rare itemsets are found to be very useful in many application areas. For example, in medical application, the rare combination of symptoms can provide useful insights for doctors. A retail business may be interested in identifying its most valuable customers i.e. who contribute a major fraction of overall company profit.Even though, still the algorithm facing some performance issues in phase-2. To overcome this, maximum transaction weight utilizations Performance of this methods needs to be investigated in low memory.

VI IMPLEMENTATION

A. System Architecture and Design

This is basic system architecture to represent the basic functionality of the system. To construct the HUP-Tree to apply the two algorithms HUP-Growth and HUP-Growth+ to find the potential high utility item sets. Main intension of this system is reducing item sets over calculated utilities.

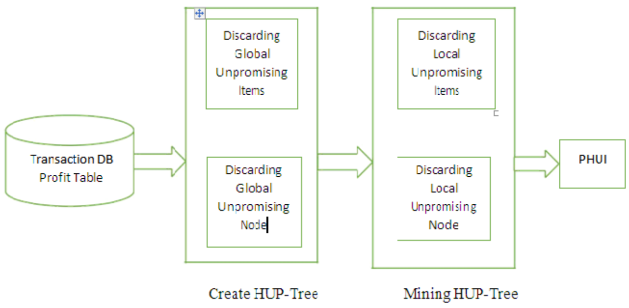

Fig .1 System Architecture

Fig. 1 contains the following blocks:

Transaction DB and Profit table are input to the system to discover potential highly utilized Item sets. Create HUP-tree: HUP-tree is created using discarding unfavourable global items and reducing global node utility. HUP-tree has fields as Node.name which contain name of the item, Node. Count, Node.nu, Node. parent, Node. hlink.

Discarding global unpromising items: After calculating transaction utility and transaction weighted utility, the item sets having less utility than predefined minimum threshold utility are disposed. Discarding global node utility: After disposing the unfavourable items the global node utilities are reduced. And nodes are inserted into HUP treeusing create HUP-tree algorithm. Mining HUP-tree: In which local unpromising Item and node utility. Discarding local unpromising items: Construct conditional pattern base of bottom item entry in header table Retrieve the entire path related to that item CPB. Conditional HUP tree created by two scans over CPB. Local unfavourable items removed using path utility of each item in CPB paths are organized in descending order. Discarding local node utility: Reorganized path is inserted into conditional utility pattern tree using reduce local node utility strategy. Potential High Utility Item sets: Identify potential high utility item sets and their utilities form HUP tree mining using Dispose of local unfavourable items and Reduce local node utility.

VII CONCLUSION AND FUTURE SCOPE

Proposed system HUP-Growth and HUP-Growth+ Mining for discovering High utility item sets from databases. Data Structure HUP-Tree for recording the information of highly utilized item sets and four effective strategies, DGU, DGN, DLU and DLN, to minimize search space and the number of candidates for utility mining. Potential high utility item sets can be generated from Utility Pattern Tree with only two scans of the database. HUP-Growth especially HUP-Growth+ Algorithm is faster than previous algorithms when database have lots of long transactions.

The current study proposed two definitions to capture the effects of the noise in the data. This pointed out possible scenarios where the mining of these patterns is central as well as the challenges in developing efficient mining algorithms. Future works include the extension of the temporal utility pattern tree to mine noisy patterns, and developing more efficient techniques to handle genomic data.

References

1. Fournier-Viger, P., 2018. FHN: efficient mining of high-utility itemsets with negative unit profits. Adv. Data Min. Appl., 16—29.
2. Fournier-Viger, P., Wu, C.W., Zida, S., Tseng, V.S., 2014. FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. Found. Intell. Syst. 8502, 83—92.
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proc. Int. Conf. Very Large Databases, pp. 487–499, (1994)
4. Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V. S.: FHM: Faster high-utility itemset mining using estimated utilityco-occurrence pruning. In: Proc. 21st Intern. Symp. on Methodologies for Intell. Syst., pp. 83–92 (2014)
5. Vincent S Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu, Fellow, “Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases”, IEEE Transactions On Knowledge And Data Engineering, volume 25, Issue No. 8, pp 1772-1786, AUGUST 2013