Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

Analysing Customer Data using Utility Mining

Karunambika R
PG Scholar
Department of CSE
PSG College of Technology
Coimbatore, India

Dr.R.Venkatesan
Professor&Head
Department of CSE
PSG College of Technology
Coimbatore, India

Dr.S.Lovelyn Rose
professor
Department of CSE
PSG College of Technology
Coimbatore, India

Abstract— Mining High Utility Itemsets is a popular data mining task, which discovers High Utility Itemsets (HUIs) from a transaction database. Recently, several efficient algorithms have been proposed for this task with different datasets. But, many of them do not consider the on-shelf time period of items and negative unit profit value of items, which is common in real transaction databases. For efficiently finding HUIs, considering these two things is crucial. In this paper, we address both of these challenges in analyzing customer data by proposing a novel efficient algorithm named FHM(Fast High Utility Itemsets Mining) to mine HUIs while considering on-shelf time periods of items, and items having positive or negative unit profit. FHM is an extension of EFIM algorithm. An extensive experimental study with real-life dataset shows that the proposed algorithm can analyze the customer data (transaction database may represent purchasing behavior of customer) efficiently to discover the importance (utility) of those customer data.

Keywords: Itemset mining, High-utility mining; Fast-utility mining; High-utility database merging and projection; on-shelf time periods.

1. INTRODUCTION

Frequent Mining (FM) is a popular data mining task, which discovers frequent itemsets (groups of items) appearing frequently in the transaction database. But, the main drawback of FM is that it assumes all items have the same weight (profit or importance) and appears only once in each transaction. This lead to a bias in the real-time applications. For example, consider the transactions of the database which describes the customer behavior of purchasing items and unit profit value for each item in the transactions. In that case, FM discovers only the frequent itemsets that the customer bought, it does not consider about profit value of that frequent itemsets. So there is a chance to miss high profitable itemsets which appears infrequently in that transaction (customer's purchasing behavior). To address this issue many efficient algorithms for mining high utility itemsets are proposed which are used different techniques that reduce the time complexity and different data structures which reduces space complexity to discover high profitable itemsets from transactions. EFIM (A Fast and Memory Efficient Algorithm for High Utility Itemset Mining) is one of the most efficient algorithms for finding high profitable itemsets from transactions. EFIM uses database projection and transaction merging technique to analyze the customer data(transactions) and discovering high profitable itemsets., but EFIM was not designed to handle negative unit profit value and the on-self periods of items. In this paper, we present a novel algorithm named FHM (Fast High-utility itemsets Mining) to mine HUIs while considering both positive and negative unit profits with the on-self periods. It extends the current EFIM algorithm so that it can handle negative unit profits and the on-self periods efficiently. The rest of this paper is organized as follows. Section 2, 3, 4and 5 respectively presents the problem definition and related work, the proposed FHM algorithm, the experimental evaluation and the conclusion.

2. PROBLEM STATEMENT AND RELATED WORK

In this section, we first state the problem of utility mining and then review the previous extensive studies of utility mining.

Problem Statement

Given a transaction database, the problem of finding high utility itemset is to find the complete set of itemsets whose utilities are greater than or equal to minimum utility threshold by analyzing customer data which contains different time periods for each

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

transaction, positive or negative unit profit value and minimum utility threshold.

Related Works

Extensive studies have been proposed for mining high utility itemsets. A base of utility mining is frequent itemsets mining. One of the important frequent mining algorithms is FP-Growth [1], which is the tree-based approach, so it can find frequent itemsets without generating any candidate itemset by scanning the database just twice. However, the main drawback of frequent mining is not considering unit profit value of items. To overcome this drawback many utility mining algorithms are proposed, like FP-Growth one of the popular algorithms for utility mining is UP-Growth: An Efficient Algorithm for High Utility Itemset Mining[2], which is also used tree-data structure to efficiently pruning unprofitable itemsets by scanning the database just twice and it uses different utility values like TU and TWU for pruning the search space for discovering high utility itemsets. But it generates more candidate itemsets before discovering HUIs based on user-specified minimum utility threshold. Mining High Utility Itemsets without Candidate Generation [3] is another popular utility mining algorithm, which used list structure instead of a tree. It scans the transaction database just once to calculate TWU and construct list structure for each item with item utility and remaining utility. Based on this remaining utility, items can be explored for discovering HUIs so it can give better performance than UP-Growth. EFIM is one of the most efficient algorithms [4], which uses different techniques (explained in section 3.2) to discover HUIs in less time and space requirement. The above algorithms are only considering positive unit profit value of items, does not care about negative unit profit.

3. THE FHM ALGORITHM

We next present our proposal, the FHM algorithm. It is an extended version of EFIM one-phase algorithm, EFIM uses several novel ideas to minimize the time and memory required for discovering high utility itemsets. FHM algorithm uses these novel ideas to the database which has transactions with the on-self period, negative or positive unit profit value. Subsection 3.1 reviews the depth-first search of itemsets. Subsection 3.2 gives an overview of EFIM algorithm. Subsection 3.3 presents proposed FHM algorithm. Finally, subsection 3.4 gives the pseudo code of FHM.

3.1 DEPTH-FIRST SEARCH

The FHM algorithm explores this search space using a depth-first search, starting from the root (the empty set). During this depth-first search, for any itemset, FHM recursively adds one item at a time according to the increasing order of TWU because it generally reduces the search space, to generate larger itemsets.

3.2 EFIM - ALGORITHM

EFIM is a single phase algorithm to discover High Utility Itemsets (HUIs) for the customer data with only positive unit profit value. Table 1. shows a Transaction Database which contains five transactions, each row represents the transaction.id, items in that transaction, TU, the utility of the items in that transaction and the on-self period of that transaction.

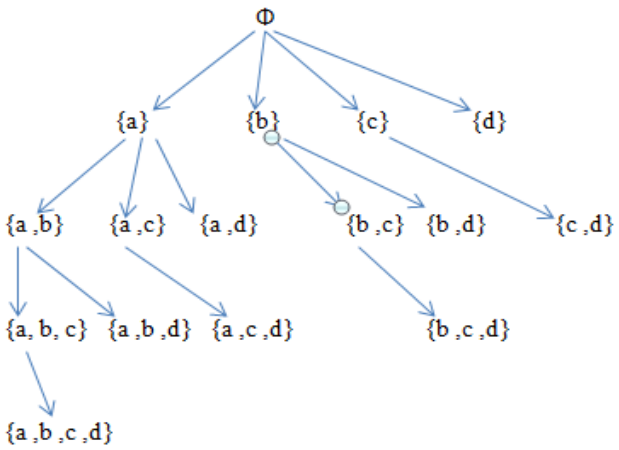

Fig.1. Depth-First Search Procedure for I - {a,b,c,d}

Table.1. Transaction Database

TransactionId	TransactionItems	TU	Utility	Period
T1	a, c, d	3	-5, 1, 2	0
T2	a, c, e, g	17	-10, 6, 6, 5	0
T3	a, b, c, d, e, f	25	-5, 4, 1, 12, 3, 5	1
T4	b, c, d, e	20	8, 3, 6, 3	1
T5	b, c, e, g	11	4, 2, 3, 2	2

Table.2. Projected Database of {c}

TransactionID	TransactionItems
T1	d
T2	e, g
T3	d, e, f
T4	d, e
T5	e, g

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

3.2.1 Transaction Utility (TU) and Transaction Weighted Utility (TWU)

Transaction Utility is a utility of transaction, which is a summation of utility values of the items in a transaction. The utility value of an item is a product of unit profit and frequency count of an item. For example, In Table 1- first transaction, item a's utility is -5, likewise (c, 1) and (d, 2). TU is 3(actually it is -3(-5+1+2) but here we take only positive utilities for TU calculation while we have negative utility, the reason for this is given in section 3.3.2). Transaction weighted Utility is a summation of the TU values of transactions in which a particular item appears. For example, item a's TWU is 45(3+17+25). TU values of transactions shown in Table 1.

3.2.3 Three Techniques in EFIM

This algorithm mainly used three techniques to efficiently reduce the search space and time required to discover HUIs. Three techniques are,

1. High Utility Database Projection,
2. High Utility Transaction Merging,
3. Using Utility-Bin Array to calculate Utility (Transaction Utility(TU), Transaction Weighted Utility(TWU).

3.2.3.1 High Utility Database Projection (HDP)

EFIM is a pattern-growth algorithm. It scans the horizontal database to calculate the utility. By using projected databases it reduces the cost of database scans. To reduce the memory space, EFIM performs pseudo-projections. Table 2 shows projection database of an item {c}.

3.2.3.2 High Utility Transaction Merging (HTM)

To further reduce the cost of database scans, EFIM also introduced an efficient transaction merging technique named High-Utility Transaction Merging. HFM is based on the observation that transaction databases often contain identical transactions. The technique consists of identifying these transactions and then replaces them with a single transaction while combining their utilities. Table 3 -shows Merged Database for an item {c}. In that Transactions T2 and T5 has identical items e and g so, those two transactions can be merged into a single transaction T(2,5) and frequency count of those items also added respectively(for example if T2 is (e,1) (g,1) and T5 is (e,2) (g,1) after merging it becomes T(2,5) which is (e,3) (g,2)).

3.2.3.3 Utility-Bin Array to calculate Utility

A utility-bin array can be used to efficiently calculate TWU of items in O(n) time (n is the number of transactions). A utility-bin array U is initialized with zero. For example, consider the database of the running example. In this example I = {a; b; c; d; e; f}. A utility-bin array U is constructed with 7 bins since there are 7 items as illustrated in Fig. 2 A. Then, the database is scanned one transaction at a time. The first transaction is T1, which has a transaction utility of 3. Because items a, c, and d appear in transaction T1, the value 3 is added to the utility-bins U[a], U[c], and U[d]. The result is shown in Fig. 2 B. Then the next transaction T2 is read, which has a transaction utility of 17. Because items a, c, e, and g appear in transaction T2, the value 17 is added to the utility-bins U[a], U[c], and U[d]. The result is shown in Fig. 2 C. Then, the same process is repeated for the remaining transactions. The content of the utility-bin array after reading transactions T3, T4, and T5 are respectively shown in Fig. 2 D, E, and F. After the last transaction has been read, it is found that the TWU of items a, b, c, d, e, f, and g are respectively 45,56, 76, 48, 73, 25, and 53, according to their respective entries in the utility-bin array.

Table.3. Merged Transactions Database of {c}

TransactionId	TransactionItems
T1	d
T(2,5)	e, g (add frequency count of 'e' and 'g' respectively)
T3	d, e, f
T4	d, e

3.3 FHM - ALGORITHM

Fast High Utility Itemsets Mining (FHM) is an extended version of EFIM algorithm with the on-self period and positive or negative unit profit value. It uses the same three techniques explained in 3.2.3, but with some modifications corresponding to negative unit profit and the on-self period. Following subsections illustrates handling on-self periods of transactions (3.3.1) and handling negative unit profit of items (3.3.2).

3.3.1 HANDLING ON-SELF PERIODS

This high utility mining algorithm does not consider the on-shelf period of items. In real-life, some items are only sold during specific time periods (For example summer). High utility mining is biased towards the items with long shelf-time because they have more opportunity to generate a higher profit. Each transaction has its period which is shown in Table 1. In FHM, we first calculate Total Period Utility (PTO) of each period. Our running example has three periods (0, 1, 2). Total Period Utility of a period is defined as the summation of TU (- and +) of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

transactions in which that period appears. For example period 0^{th} PTO is 5(-2 for T1 + 7 for T2). In FHM, we calculate TWU for all items and for all periods. Each TWU is divided by that corresponding period's PTO. This value can be used for comparing with user-defined Minimum Utility threshold value (minutil) for discovering HUI.

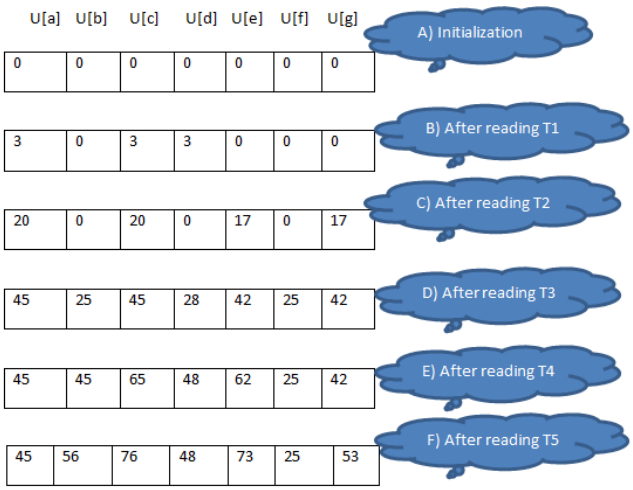

Fig.2. TWU Calculation using Utility-Bin Array

3.3.2 HANDLING NEGATIVE UNIT PROFIT

In high utility mining, items are not allowed to have negative unit profit, but in real-life transaction databases, items are often sold at a loss. FHM integrate new ideas to handle negative items more efficiently. In FHM, first, we calculate two types of TU, TU (+) is calculated with the only positive utility of items and TU (- and +) is with both positive and negative utility of items. While calculating TWU, we use TU (+) because if we use the negative utility of an item then that item can't be explored, but there may be a chance to miss itemset which can give HUI. Likewise while calculating Total Period Utility and transaction merging (for comparing items to find identical items) we used TU (- and +).

3.4 PSEUDOCODE OF FHM

In this subsection, we present the FHM algorithm, which combines all the ideas presented in the previous section. The main procedure (Algorithm 1) takes as input a transaction database and the minutil threshold. The algorithm initially considers that the current itemset α is the empty set (line 1). The algorithm then scans the database once to calculate the local utility of each item w.r.t. α and positive and negative unit profit, using a utility-bin array (line 2). Note that in the case where α = ϕ, the local utility of an item is its TWU which is divided by PTO (h) h represents corresponding period. Then, the local utility of each item is compared with minutil to obtain the secondary items w.r.t to α that is items that should be considered in extensions of α (line3). Then, these items are sorted by ascending order of TWU and that order is thereafter used as the x order (line 4). The database is then scanned once to remove all items that are not secondary items w.r.t to α since they cannot be part of any high-utility itemsets (line 5). At the same time, items in each transaction are sorted according to x, and if a transaction becomes empty, it is removed from the database. Then, the database is scanned again to sort transactions by the xT order to allow O(nw) transaction merging, thereafter (line6). Then, the algorithm scans the database again to calculate the sub-tree utility of each secondary item w.r.t. α, using a utility-bin array (line 7 and 8).Thereafter, the algorithm calls the recursive procedure Search to perform the depth-first search starting from α (line 9).

The Search procedure (Algorithm 2) takes as parameters the current itemset to be extended, the α projected database, the primary and secondary items w.r.t α and the minutil threshold. The procedure performs a loop to consider each single-item extension of α of the form β = α U {i}, where i is a primary item w.r.t α (line 1 to 9). For each such extension β, a database scan is performed to calculate the utility of β and at the same time construct the β projected database (line 3). Note that transaction merging is performed at the same time the β projected database is constructed. If the utility of β is no less than minutil, β is output as a high-utility itemset (line 4). Then, the database is scanned again to calculate the sub-tree and local utility w.r.t β of each item z that could extend β (the secondary items w.r.t to α), using two utility-bin arrays (line 5). This allows determining the primary and secondary items of β (line 6and 7). Then, the Search procedure is recursively called with β to continue the depth-first search by extending β (line 8). Based on properties and theorems presented in previous sections, it can be seen that when FHM terminates, all and only the high-utility itemsets have been output.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

Algorithm 1: The FHM algorithm

input: D: a transaction database, minutil: a user-specified threshold
output: the set of high-utility itemsets

1 α = ϕ;
2 Calculate lu (α, I, h) for all items i in I and each period h w.r.t both positive and negative unit profit by scanning D, using a utility-bin array;
3 Secondary(α) = {i | i in I ^ (lu(α, i, h)/PTO(h)) ≥ minutil };
4 Let x be the total order of TWU ascending values on Secondary (α);
5 Scan D to remove each item i not in Secondary (α) from the transactions, sort items in each transaction according to x, and delete empty transactions;
6 Sort transactions in D according to xT;
7 Calculate the sub-tree utility su (α, i, h) of each item i and in Secondary (α) w.r.t each period i by scanning D, using a utility-bin array;
8 Primary (α) = {i | i in Secondary (α) ^ (su (α, I, h)/PTO (h)) ≥ minutil};
9 Search (α, D, Primary (α), Secondary (α), minutil);

Algorithm 2: The Search procedure

input: α: an itemset, α -D: the α projected database, Primary (α): the primary items of α, Secondary (α): the secondary items of α, the minutil threshold
output: the set of high-utility itemsets that are extensions of α

1 **foreach** item i in Primary (α) do
2 β = α U {i};
3 Scan α -D to calculate utility (β) and create β -D; // uses transaction merging
4 if utility (β) ≥ minutil then output β;
5 Calculate su (β, z, h) and lu (β, z, h) for all item z in Secondary (β) by scanning β -D once, using two utility-bin arrays;
6 Primary (β) = {z in Secondary (α) | su (β, z, h) ≥ minutil};
7 Secondary (β) = {z in Secondary (α) | lu (β, z, h) ≥ minutil};
8 Search (β, β -D, Primary (β), Secondary (β), minutil);
9 end

4. EXPERIMENTAL RESULT

We performed several experiments to evaluate the performance of the proposed FHM algorithm. Experiments were carried out on a computer with 64-bit Core i7 processor running Windows 7 and 4 GB of RAM. Algorithms were implemented in Java and memory measurements were done using the standard Java API. Experiments were performed using a standard dataset used in the FOSHU literature for evaluating FHM algorithm which is downloaded from SPMF open-source data mining library details shown in Table4. Subsection 4.1 gives the influence of user-specified minimum utility threshold and 4.2 gives time and space complexity of FHM algorithm.

4.1 INFLUENCE OF MINIMUM UTILITY THRESHOLD USER SPECIFIED (MINUTIL)

Table 5 shows the results for minutil (0.65, 0.75, 0. 85), we can observe that while minutil value increases the number of HUIs generated was reduced.

4.2 TIME AND SPACE COMPLEXITY

The time complexity of FHM is O(lnw), where n and w is respectively the number of transactions and the average length of transactions, finally, l is a number of itemsets in the search space. Space complexity is O(Ih)+O(lnw), wherein O(Ih) I and h means total items and period, Ih gives space requirement of bin-array to calculate TU and TWU and O(lnw) is space requirement for projected database because the number of projected databases is determined by the number of itemsets in the search space l.

Table.4. Dataset - mushroom

No Of Transactions	No Of Items	Total Periods
8124	120	25

Table.5. Results

MinimumUtilityThreshold	No Of HUIs	Time(ms)	MemoryUsage(MB)
0.65	113	66000	45
0.75	100	50000	38
0.85	80	35000	30

5. CONCLUSION

High Utility mining is an important task in many applications. But still, it has some drawbacks like not considering negative or positive unit profit value with an on-self period. The EFIM algorithm uses some unique techniques to reduce the time and memory required to discover HUIs but not handled negative profit and the on-self period. So, for analyzing customer data (for example, transactions represents purchasing behavior of customers) we presented an algorithm called FHM with on-self period and negative unit profit of items. FHM uses the techniques in EFIM with some modifications for handling periods and negative unit profit values, like using TU(+), TU(- and +), PTO for each period and TWU which is used by dividing with respect to corresponding period's PTO.

REFERENCES

[1] J. Han, J. Pei, and Y. Yin. "Mining frequent patterns without candidate generation," in Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.
[2] V.S. Tseng, B.-E. Shie, C.-W. Wu and P.S. Yu. "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases," in IEEE Trans. Knowl.Data Eng, 25(8):1772{1786, 2013.
[3] M. Liu and J. Qu. "Mining High Utility Itemsets without Candidate Generation," in Proc. 21st ACM Intern. Conf. Inform. Known. Management, pp. 55{64, 2012.
[4] Zida S, Fournier-Viger P et al." EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining," in Proceedings of the 14th Mexican

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

international conference on artificial intelligence, 9413, Springer, Berlin, pp. 530{546, 2015.

[5] G.-C. Lan, T.-P. Hong and V.S. Tseng. “Discovery of high utility itemsets from on-shelf time periods of products,” in Expert Systems with Applications,”.38:5851{5857, 2011.

[6] G.-C. Lan, T.-P. Hong, J.-P. Huang and V.S. Tseng. “On-shelf utility mining with negative item values,” in Expert Systems with Applications,”.41:3450{3459, 2014.

[7] C.-Y. Li, J.-S. Yeh, C.-C. Chang. “Isolated items discarding strategy for discovering high utility itemsets,” in Data & Knowledge Engineering, 64(1): 198{217, 2008.