ISSN: 2277-9655
Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A SURVEY PAPER ON HIGH UTILITY ITEMSETS MINING

Mrs. Shweta A. Dubey*, Prof. Kemal. Koche

* M.tech Student: Dept. of Computer Sci. and Engg. J. D. College of Engineering and Management, Nagpur, Maharashtra, India

Dept. of Computer Sci. and Engg.J. D. College of Engineering and Management, Nagpur, Maharashtra, India

DOI: 10.5281/zenodo.52492

ABSTRACT

An important data mining task that has received considerable research attention in recent years is the discovery of association rules from the transactional databases. Recently, Utility mining plays a vital role in data mining. To discover high utility itemset from transactional database means discovering item sets with high profits. In this survey paper, we discuss about various methods and algorithms which were used for recovering high utility itemsets from a large database without losing large amount of information.We present different kind of algorithm such as CHUD(Closed High Utility Itemset Discovery) for mining closed itemset and further a method called DAHU which discovers all high utility itemsets from a result generated after applying CHUD algorithm .Itemset mining has a wide range of applications in biomedical applications, retail stores, super market etc.

KEYWORDS: data mining, high utility,itemset discovery,frequent itemset,high profits,most preffered itemset.

INTRODUCTION

Data mining is referred to as a process of extracting useful data or information from a large set of Knowledge base. In certain application, it is a kind of analysis of data that is frequently used or purchased by customers. In this paper FIM-is the model which extract the frequent itemset from a large amount of itemset. In most cases, it may be possible that FIM may discover the items that appears frequently in the transaction but generates low profits. That is, it may lose some information on valuable items which are generating high profits but not appear frequently in transaction. The traditional association rule mining methods are based on support-confidence framework, where all items are considered with the same level of importance. The methods proposed in this paper producing the same result on a given minimum support and minimum confidence. The weighted associated with each item generalizes the traditional framework by giving importance to items, where importance is given as weight (unit).

In applications such as retail stores, it is desired to note that the quantities of items in transactions are not considered. Considering items quantities in transactions and their individual importance, high utility itemset mining (HUIM) received a considerable research attention. The utility of itemset represents its importance, which can be measured in terms of weight, profit, cost, quantity or other information depending on user preference. An itemset is said to be most preferred itemset (abbreviated as MPI) if its utility is no less than a user defined constant value otherwise it is called a less utility itemset/low utility itemset.

Downward closure property: The downward closure property (also called anti-monotonicity) of support .This property allows us to avoid counting itemset that will turn out to be infrequent at the end.

FIM: FIM is an interesting branch of data mining, which deals with looking at the action of sequence of events. In frequent itemset mining, the base data takes the form of sets of instances also called transaction that each has a number of feature also called item.

Developing a compact and a complete representation of most preferred itemset has several challenges:
1. The concept of integrating from FIM into MPI mining may produce a losy representation of all HUI's or a representation that is not meaningful to the users.
2. The representation may not achieve a significant reduction in the number of extracted patterns to justify using the representations.
3. Algorithms for extracting the representations may not be efficient. They may be slower than the best algorithm for mining all HUI.
4. It may be hard to develop an efficient method for recovering all HUI's from the representations.

In this paper, our objectives are Four –Fold and correspond to resolving the previous four challenges:

1. Our objective is to recover all HUI's and their utilities efficiently by using new structured named as "Unified Unit Array."
2. Our objective is to reduce the number of itemset by several orders of magnitude, especially for dataset containing long HUI's.
3. Our objective is to enhance performance by using three strategies REG, RML and DCM. Results show that CHUD is much faster than the current best method for mining all HUI's.
4. Our objective is to recover all HUI's efficiently from the set of closed+HUI's by using a new method DAHU (Derive all Utility itemsets)

RELATED CONCEPTS AND METHODOLOGIES

Push all the closed property into HUI Mining

The first point that we should discuss is how to incorporate the closed constraint into high utility itemset mining. There are several possibilities. First, we can define the closure on the utility of itemsets. In this case, a high utility itemset is said to be closed if it has no proper superset having the same utility. However, this definition is unlikely to achieve a high reduction of the number of extracted itemsets since not many itemsets have exactly the same utility as their supersets in real datasets. For example, there are seven HUIs in Example 1 and only one itemset {E} is non-closed, since {E}{ABE} and u({E}) = u({ABE}) = 12. A second possibility is to define the closure on the supports of itemsets. In this case, there are two possible definitions depending on the join order between the closed constraint and the utility constraint:

a. Mine all the high utility itemsets first and then apply closed constraint. We formally define this set as H' = fC(fH(L)). It follows that H'H.
b. Mine all the closed itemsets first and then apply the utility constraint. We formally define this set as C' = fH(fC(L)). It follows that C'C.

The AprioriHC Algorithm

Initially, a variable k is set to 1. The algorithm performs a database scan to compute the transaction utility of each transaction (Definition 4). At the same time, the TWU of each item is computed. Each itemhaving aTWU no less than abs_min_utility is added to the set of 1-HTWUIs C_k. Then the algorithm proceeds recursively to generate itemsets having a length greater than k. During the kth iteration, the set of k-HTWUIs L_k is used to generate (k + 1)-candidates C_{k+1} by using the Apriori-gen function. Then the algorithm computes TWUs of itemsets in C_{k+1} by scanning the database D once. Each itemset having a TWU no less than abs_min_utility is added to the set of (k +1)-HTWUIs L_{k+1}. After that, the algorithm removes non-closed itemsets in L_{k+1} by the following process. For each candidate X in L_{k+1}, the algorithm checks if there exists a subset Y _ X such that Y 2 Lk and SC(X)=SC(Y). If true, Y is deleted from Lk because Y is not a closed+ high utility itemset according to Definition 14. If false, Y is kept and marked as "closed" because it may be a closed+ high utility itemset. The phase I of Apriori-HC terminates when no candidate is generated. Then, the algorithm performs Phase II. In phase II, the algorithm scans the database once and calculates the utilities of HTWUIs that are marked as "closed" to identify the set of closed high utility itemsets.

Efficient Discovery of Closed+ High Utility Itemsets Discovery (CHUD)

a. In this subsection, we present an efficient algorithm named CHUD (Closed+ High Utility itemset Discovery) for mining closed+ HUIs. CHUD is an extension of DCI-Closed, one of the current best methods for mining closed itemsets, and it also integrates the TWU model and effective strategies to prune low utility itemsets.

CHUD consists of two phases. In phase I, CHUD discovers candidates for closed+ HUIs. In phase II, the closed+ HUIs are identified from the set of candidates found in phase I and their utility unit arrays are computed by scanning the database once.

b. Similar to the DCI-Closed algorithm, CHUD adopts an IT-Tree (Itemset-Tidset pair Tree) to find closed+ HUIs. In an IT-Tree, each node N(X) consists of an itemset X, its Tidset g(X), and two ordered sets of items named PREV-SET(X) and POST-SET(X). The IT-Tree is recursively explored by the CHUD algorithm until all closed itemsets that are HTWUIs are generated. Different from the DCI-Closed algorithm, each node N(X) of the IT-Tree is attached with an estimated utility value EstU(X).

c. A data structure called TU-Table (Transaction Utility Table) is adopted for storing the transaction utilities of transactions. It is a list of pairs <R, TU(TR)> where the first value is a TID R and the second value is the transaction utility of T_R.

Efficient Recovery of High Utility Itemsets (DAHU)

In this subsection, we present a top-down method named *DAHU* (*D*erive *A*ll *High Utility* itemsets) for efficiently recovering all the HUIs. It takes as input a *min_utility* threshold, a set of closed+ HUIs *HC* and *Kmax* the maximum length of itemsets in *HC*. DAHU outputs the complete set of high utility itemsets $H = \mathrm{U}^{K}{}_{i=1}$ *H*K respecting *min_utility*, where *H*K denotes the set of HUIs of length *K*. To derive all HUIs, DAHU proceeds as follows. First, the set *H*K*max* is initialized to *HC*K*max*, where the notation *HC*K represents the set of *K* itemsets in *HC*. During step 2 to step 14 in Figure 7, each set *H*K is constructed from *K* = (*Kmax* – 1) to *K* = 1. In each iteration, *H*(K–1) is recovered by using *HC*K. For each itemset *X* = {*a*1, *a*2,…,*a*K} in *HC*K, if the utility of *X* is no less than *min_utility*, the algorithm outputs the high utility itemset *X* with its exact utility and then generates all (*K*–1)-subsets of *X*. The latter are obtained by removing each item *a*i ∈ *X* from *X* one at a time to obtain subsets of the form *Y* = *X* – {*a*i}. If *Y* is not present in *H*K or *Y* is present in *H*K with *SC*(*X*) > *SC*(*Y*), *Y* is added to *H*(K–1), its support count is set to the support count of *X* (Property 4), i.e., *SC*(*Y*) = *SC*(*X*), and the utility of *Y* is set to the utility of *X* minus the *i*-th value in *V*(*X*), i.e., *u*(*Y*) = *u*(*X*) – *V*(*X*, *a*i) (Property 6-8). In addition, the utility unit array of *V*(*Y*) is set to *V*(*X*) with the value *V*(*X*, *a*i) removed (Property 8). This process is repeated until *H* has been completely recovered.

PROPOSED SCHEME

CLOSED AND MAXIMAL FREQUENT ITEMSET

The downward-closure property of support also allows us to compact the information about frequent itemsets. First, some definitions:

- An itemset is *closed* if none of its immediate itemset has the same count as the itemset.
- An itemset is *closed frequent* if it is frequent and closed.
- An itemset is *maximal frequent* if it is frequent and none of its immediate superset is frequent.

For example, assume we have items = {apple, beer, carrot} and the following baskets:

{apple, beer}
{apple, beer}
{beer, carrot}
{apple, beer, carrot}
{apple, beer, carrot}

Assuming the support threshold :

Itemset	Count	Frequent?	Closed?	Clsd Freq?	Max Freq?
{apple}	4	yes	no	no	no
{beer}	5	yes	yes	yes	no
{carrot}	3	yes	no	no	no
{apple, beer}	4	yes	yes	yes	yes
{apple, carrot}	2	no	no	no	no
{beer, carrot}	3	yes	yes	yes	yes
{apple, beer, carrot}	2	no	yes	no	no

ISSN: 2277-9655
Impact Factor: 3.785

From the table above, we see that there are 5 frequent itemsets, of which only 3 are closed frequent, of which in turn only 2 are maximal frequent. The set of all maximal frequent itemsets is a subset of the set of all closed frequent itemsets, which in turn is a subset of the set of all frequent itemsets. Thus, maximal frequent itemsets is the most compact representation of frequent itemsets. In practice, however, closed frequent itemsets may be preferable because they carry along not just the frequent information, but also the exact count.

IMPLEMENTATION

FHM takes as input a transaction database with utility information and a minimum utility threshold *min_utility* (a positive integer). Let's consider the following database consisting of 5 transactions (t1,t2...t5) and 7 items (1, 2, 3, 4, 5, 6, 7)

	Items	**Transaction utility(sum of 4th column)**	**Item utilities for this transaction**
t1	3 5 1 2 4 6	30	1 3 5 10 6 5
t2	3 5 2 4	20	3 3 8 6
t3	3 1 4	8	1 5 2
t4	3 5 1 7	27	6 6 10 5
t5	3 5 2 7	11	2 3 4 2

Each line of the **database** is:

- a set of items (the first column of the table),
- the sum of the utilities (e.g. profit) of these items in this transaction (the second column of the table),
- the utility of each item for this transaction (e.g. profit generated by this item for this transaction)(the third column of the table).
-

Note that the value in the second column for each line is the sum of the values in the third column.

What are real-life examples of such a database? There are several applications in real life. One application is a customer transaction database. Imagine that each transaction represents the items purchased by a customer. The first customer named "**t1**" bought items 3, 5, 1, 2, 4 and 6. The amount of money spent for each item is respectively 1 $, 3 $, 5 $, 10 $, 6 $ and 5 $. The total amount of money spent in this transaction is 1 + 3 + 5 + 10 + 6 + 5 = 30 $.

The output of **FHM** is the set of **high utility itemset**s having a utility no less than a ***min_utility*** threshold (a positive integer) set by the user. To explain what is a high utility itemset, it is necessary to review some definitions. An **itemset** is an unordered set of distinct items. The **utility of an itemset in a transaction** is the sum of the utility of its items in the transaction. For example, the utility of the itemset {1 4} in transaction t1 is 5 + 6 = 11 and the utility of {1 4} in transaction t3 is 5 + 2 = 7. The **utility of an itemset in a database** is the sum of its utility in all transactions where it appears. For example, the utility of {1 4} in the database is the utility of {1 4} in t1 plus the utility of {1 4} in t3, for a total of 11 + 7 = 18. A **high utility itemset** is an itemset such that its utility is no less than *min_utility* For example, if we run **FHM** with a **minimum utility of 30**, we obtain 8 **high-utility itemsets**:

itemsets	utility	support
{2 4}	30	40 % (2 transactions)
{2 5}	31	60 % (3 transactions)
{1 3 5}	31	40 % (2 transactions)
{2 3 4}	34	40 % (2 transactions)

ISSN: 2277-9655
Impact Factor: 3.785

{2 3 5}	37	60 % (3 transactions)
{2 4 5}	36	40 % (2 transactions)
{2 3 4 5}	40	40 % (2 transactions)
{1 2 3 4 5 6}	30	20 % (1 transactions)

If the database is a transaction database from a store, we could interpret these results as all the groups of items bought together that generated a profit of 30 $ or more.

Input file format
The **input file format** of **FHM** is defined as follows. It is a text file. Each lines represents a transaction. Each line is composed of three sections, as follows.

- First, the items contained in the transaction are listed. An item is represented by a positive integer. Each item is separated from the next item by a single space. It is assumed that all items within a same transaction (line) are sorted according to a total order (e.g. ascending order) and that no item can appear twice within the same transaction.
- Second, the symbol ":" appears and is followed by the transaction utility (an integer).
- Third, the symbol ":" appears and is followed by the utility of each item in this transaction (an integer), separated by single spaces.

For example, for the previous example, the input file is defined as follows:
3 5 1 2 4 6:30:1 3 5 10 6 5
3 1 4:8:1 5 2
3 5 1 7:27:6 6 10 5
3 5 2 7:11:2 3 4 2

Consider the first line. It means that the transaction {3, 5, 1, 2, 4, 6} has a total utility of 30 and that items 3, 5, 1, 2, 4 and 6 respectively have a utility of 1, 3, 5, 10, 6 and 5 in this transaction. The following lines follow the same format.

Output file format
The **output file format** of **FHM** is defined as follows. It is a text file, where each line represents a **high utility itemset**. On each line, the items of the itemset are first listed. Each item is represented by an integer, followed by a single space. After, all the items, the keyword " #UTIL: " appears and is followed by the utility of the itemset. For example, we show below the output file for this example.

2 4 #UTIL: 30 2 5 #UTIL: 31
1 3 5 #UTIL: 31
2 3 4 #UTIL: 34
2 3 5 #UTIL: 37
2 4 5 #UTIL: 36
2 3 4 5 #UTIL: 40
1 2 3 4 5 6 #UTIL: 30

For example, the first line indicates that the itemset {2, 4} has a utility of 30. The following line follows the same format.

Performance
High utility itemset mining is a more difficult problem than frequent itemset mining. Therefore, high-utility itemset mining algorithms are generally slower than frequent itemset mining algorithms.

The FHM algorithm was shown to be up to six times faster than **HUI-Miner** (also included in SPMF), especially for sparse datasets (see the performance section of the website for a comparison). But the **EFIM** algorithm (also included in SPMF) greatly outperforms FHM.

CONCLUSION & FUTURE SCOPE

The Outcome of our research is summarized as follows:

1. In Final Outcome we will achieve **high efficiency for the mining task and provide a concise mining result to users**, by using a novel framework in this paper for mining closedþ high utility itemsets (CHUIs), which serves as a compact and lossless representation of HUIs. We propose three efficient algorithms named AprioriCH (Apriori-based algorithm for mining High utility Closed þ itemsets), AprioriHC-D (AprioriHC algorithm with Discarding unpromising and isolated items) and CHUD (Closed þ High Utility Itemset Discovery) to find this representation.
2. By using method called DAHU (Derive All High Utility Itemsets), we will recover all HUIs from the set of CHUIs without accessing the original database.

Also, we can conclude that, we addressed the problem of redundancy in high utility itemset mining by proposing a lossless and compact representation named closed+ high utility itemsets, which as not been explored so far. To mine this representation, we proposed three efficient algorithms named AprioriHC (Apriori-based approach for mining High utility Closed itemset), AprioriHC-D (AprioriHC algorithm with Discarding unpromising and isolated items) and CHUID (Closedþ High Utility itemset Discovery). AprioriHC-D is an improved version of AprioriHC, which incorporates strategies DGU [24] and IIDS [19] for pruning candidates. AprioriHC and AprioriHCD perform a breadth-first search for mining closedþ high utility itemsets from horizontal database, while CHUD performs a depth-first search for mining closedþ high utility itemsets from vertical database. The strategies incorporated in CHUD are efficient and novel. They have never been used for vertical mining of high utility itemsets and closed+ high utility itemsets. To efficiently recover all high utility itemsets from closedþ high utility itemsets, we proposed an efficient method named DAHU (Derive All High Utility itemsets). Results on both real and synthetic datasets show that the proposed representation achieves a massive reduction in the number of high utility itemsets on all real datasets (e.g. a reduction of up to 800 times for Mushroom and 32 times for Foodmart). Besides, CHUD outperforms UPGrowth, one of the currently best algorithms by several orders of magnitude (e.g. CHUD terminates in 80 seconds on BMSWebView1 for min utility ¼ 2%, while UP-Growth cannot terminate within 24 hours). The combination of CHUD and DAHU is also faster than UP-Growth when DAHU could be applied.

FUTURE WORK

In this paper we have given the detail survey of various utility mining datasets and also presented the comparative analysis of different methods and algorithm. Also there aremany other compact representation that have not yet been combined with high utility itemset mining.This is an interesting research question. Although high utility itemset miningis essential to many research topics and industrial application.We will try to explore worthwhile research in this area.

REFERENCES

[1] T. Hamrouni, "Key roles of closed sets and minimal generators in concise representations of frequent patterns" Intell. Data Anal. vol. 16, no. 4, pp. 581–631, 2012.
[2] G.-C. Lan, T.-P. Hong, and V. S. Tseng, "**An efficient projection based indexing approach for mining high utility itemsets**," Knowl. Inf. Syst, vol. 38, no. 1, pp. 85–107, 2014
[3] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, "**H-mine: Fast and space-preserving frequent pattern mining in large databases**," IIE Trans., vol. 39, no. 6, pp. 593–605, Jun. 2007.
[4] B.-E. Shie, V. S. Tseng, and P. S. Yu, "**Online mining of temporal maximal utility itemsets from data streams**," in Proc. Annu. ACM Symp. Appl. Comput., 2010, pp. 1622–1626.
[5] J. Wang, J. Han, and J. Pei, "**Closet+: Searching for the best strategies for mining frequent closed itemsets**," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2003, pp. 236–245.
[6] Vincent S. Tseng, Cheng-Wei Wu, Philippe Fournier-Viger and Philip S. Yu, "**Efficient algorithms for Mining the concise and Lossless Representation of High Utility Itemsets**," in IEEE Trans. Knowl. Data Eng., vol. 27, no. 3, March 2015