Mining High Utility Patterns in Single Phase

Vishnu Priya R[1], Minu Augustine[2], Chitra S Nair[3]

[1]PG Scholar, NSS College of Engineering, Palakkad, Kerala, India

[2]PG Scholar, NSS College of Engineering, Palakkad, Kerala, India

[3]Assistant Professor, NSS College of Engineering, Palakkad, Kerala, India

Abstract: ***Utility mining is a growing trend in data mining area. Most of the works on utility mining adapts a two phase candidate generation approach that is however inefficient. High utility item set refer to the set of items with high utility for example, profit in a database. To identify high utility item sets, most of the algorithms first generate candidate item sets by estimating their utilities, and then compute their exact utilities. This two phase approach has scalability issue due to the large number of candidates that it generates. This approach raises the problem that a large number of candidates are generated, but most of the candidates are found not to be high utility after their exact utilities were computed. This paper proposes a method that finds high utility patterns in single phase. All the item sets having a utility meeting a user specified minimum utility threshold are discovered. However, setting the threshold value is a difficult problem for users. If the threshold is set too small, a huge number of item sets will be generated. On the other hand, if it is set too high, no item sets will be presented for the users. This paper also address this issue by proposing a framework for top k high utility item set mining, where k is the desired number of high utility item sets to be mined.***

Keywords: Utility mining, High utility patterns, frequent item set mining, and top-k pattern mining

1. Introduction

The rapid development of database technique facilitates the storage and usage of massive data from various organizations. How to obtain valuable informations from various databases has received considerable attention that result in the rise of related research areas. Among these, the item set mining problem is most important and that drives from the frequent item set mining problem.

Finding interesting patterns has been an important data mining task, and it has got a variety of applications. For example, genome analysis, cross marketing, condition marketing etc., where interestingness measures play an important role. Mining frequent item sets is to identify the set of items that appear frequently in transactions. The frequency of an item set is measured with the support of the item set ie., the number of transactions containing the item set. If the support of an item set exceeds a user specified minimum support threshold, the item set is considered as frequent. Most of the frequent item set mining algorithms employ the down word closure property of item sets. That is, all super set of an infrequent item set are infrequent and, all sub set of a frequent item set are frequent. This property provides the algorithms with a powerful pruning strategy. In frequent pattern mining, once an infrequent item set is identified, the algorithm need not to check the super set of that item set. Mining of frequent item sets only takes in to account, the presence and absence of items. Other information about the item is not considered, such as the independent utility of an item and the context utility of an item in a transaction. In a super market database, each item has a definite price/profit and each item in a transaction is associated with a distinct count which means the quantity of the item one bought.

Frequent item set mining is a fundamental research area in data mining. However, the traditional frequent item set mining techniques may discover a large amount of frequent but low-value item sets. And they loss the information on valuable item sets having low selling frequencies. Hence, it cannot satisfy the requirement of users who want to discover item sets with high utilities such as high profits. With frequent pattern mining an item is considered to be frequent if its occurrence frequency exceeds a user specified threshold. For example, mining frequent patterns from a shopping transaction database refers to the discovery of products that are frequently purchased by a customer. However, a user’s interest may relate to many other factors that are not related to occurrence frequency. For example, a super market manager will be interested in finding products with high profits that are related to unit profit and purchased quantity, that are not considered in frequent pattern mining.

Utility mining was emerged recently to address the limitations of frequent pattern mining by considering user’s expectation as well as the raw data. Like frequent item sets, item sets with utilities not less than a user specified minimum utility threshold is generally valuable and interesting, and they are called high utility item sets. The downward closure property of item sets no longer exist for high utility item sets. When items are added to the item set one by one, the support of item set monotonously decreases or remains unchanged, but the utility of item set varies irregularly.

Most of the utility mining algorithms adopt a two phase, candidate generation approach, that is, first find candidates of high utility patterns in the first phase, and then scan the raw data one more time to find high utility patterns from thecandidates in the second phase. In the second phase, the algorithms compute the exact utilities of candidates by scanning the database. The challenge is that number of candidates can be huge, which is the scalability and efficiency bottleneck. It also creates problem when the raw data contains many more transactions or the minimum utility threshold is small. The large number of candidates issue not only in the first phase but also in the second phase. This paper proposes a method that finds high utility patterns in single phase.

Even though many methods have been proposed for high utility mining, it is difficult for users to choose an appropriate minimum utility threshold. Depending on the threshold, the output size can be either small or large. If the threshold is set too small, large number of high utility item sets will be displayed to the user. Such a large number of high utility item sets may cause the mining algorithm inefficient. It may also leads to run out of memory. Because, the higher utility item sets the algorithm produce, the more resources they consume. On the other hand, if the threshold is set too high, no high utility item sets will be generated. To find an appropriate value for the minimum utility threshold, the user needs to guess and repeatedly run the algorithm. This process is both inconvenient and time consuming. To discover the high utility item sets without defining the utility threshold is to redefine the task of high utility item sets mining as mining top k high utility item set, where k is the number of desired number of item sets that the users want to find. The idea is to let the user specify k, instead of specifying the minimum utility threshold. And also, choosing the threshold primarily depends on the database characteristics that are often unknown to users. This paper also proposes a framework for mining top k high utility item sets, where k is the desired number of item sets to be mined.

2. Related Works

High utility pattern mining is related to frequent pattern mining and constraint based mining. This section briefly describes prior works on both of these areas and how they connect to the proposed method.

2.1 Frequent pattern mining

Frequent pattern mining was first introduced by Agarwal et al, [3]. It discovers all patterns whose support are not less than a user specified support threshold. Frequent pattern mining employs the anti-monotonicity property: the support of the super set of a pattern is no more than the support of the pattern. Frequent pattern mining algorithm as well as high utility pattern mining algorithms fall into three categories, breadth-first search, depth-first search and hybrid search. Apriori by Agrawal and Srikant [4] is a breadth-first algorithm for mining frequent patterns. FP-growth by Han et al, [5] is a depth-first algorithm. It compresses the database by FP-trees in main memory. Eclat by Zaki [6] is a hybrid algorithm that keeps a database in memory by a vertical tid-list layout [8], and can work either in depth-first or breadth-first manner.

As the breadth-first strategy is more memory intensive, it is used in this paper.

2.2 Constraint based mining

The constraint based is a milestone from frequent pattern mining to utility mining. Many of the works on this area mainly focus on how to push constraints into frequent pattern mining algorithm. Bucila et al. [9] considered mining patterns that satisfy both anti-monotone and monotone constraints, and presented an algorithm, DualMiner, that efficiently reduces its search space using both anti-monotone and monotone constraints.

De Raedt et al. [10] investigated how constraints based method can be applied to constraint based mining problems with constraints that are monotone, anti-monotone and convertible. Bayarado and Agrawal [11] and Morishita and Sese [12] proposed techniques of pruning based on upper bounds when the constraint is neither monotone, anti-monotone nor convertible. This paper employs such a technique. The contribution of this paper is to tight upper bounds on the utility.

2.3 Top-k pattern mining

Many papers have been developed to mine different types of top-k patterns such as top-k frequent item sets [13, 14, 15], top-k frequent closed item sets [13, 16], top-k closed sequential item sets [17], top-k association rules [18]. The difference in each mining algorithm lies in the type of pattern discovered, as well as the data structures and search strategies that are used. For example, some algorithms [19, 18] are using rule expansion strategy for finding patterns, while others use a pattern growth search using structures such as FP-Tree[19, 20, 28]. The possibility of data structures and search method affect the efficiency of a top-k pattern mining algorithm in terms of both memory and execution time. However, these algorithms discover top-k patterns according to traditional measures instead of the utility measure. As a consequence, they may miss patterns producing high utility.

2.4 Top-k high utility pattern mining

The task of top-k high utility pattern mining was introduced by Chan et al [20]. But, the definition of high utility item set used in their study is different from the one used in other works. Chan et al's study has examined utilities of various items, but relative values of items in transactions were not taken into consideration. Zihayat and An [21] have proposed a systematic algorithm T-HUDS for mining top k HUIs over data streams. Yin et al. [22] has proposed a new framework for mining top-k high utility sequential patterns. Ryung and Yun proposed the REPT algorithm [23] with four strategies PUD, RIU, RSD and SEP for top-k HUI mining. In REPT, besides the parameter k, users need to set another parameter N to control the effectiveness of RSD [23]. However, it is not easy for users to select an appropriate N value and the choice of N greatly affects the performance of REPT.

3. Methodology

This section briefly describes the utility mining problem that we are going to deal with. Let I be the universe of items. Let D be a database of transactions $t_1 \ldots t_n$, where each $t_i \subseteq I$. Each item in a transaction is given a nonzero share. Each distinct item has a weight independent of any transaction, given by an eXternal Utility Table (XUT). The research problem of finding all high utility patterns is formally defined as follows.

The inner utility of an item i in a transaction t, represented by iu (i, t), is the share of i in t. The external utility of an item i, denoted by eu (i), is the weight of i independent of any transaction. The utility of an item i in a transaction t, represented by u(i,t), is the function f of iu(i,t) and eu(i), that is, u(i,t)=f(iu(i,t),eu(i)).

A transaction t contains a pattern X if X is a subset of t, which means that every item i in X has a non-zero share in t, that is, iu (i, t) $\neq$ 0. The transaction set of a pattern X, denoted by TS(X), is the set of transactions that include X. The count of transactions in TS(X) is the support of X, denoted by s(X).

For a pattern X contained in a transaction t, that is, X ⊆t, the utility of X in t, represented by, u (X, t), is the sum of the utility of every component item of X in it, that is,

$$u(X,t) = \sum_{i \in X \subseteq t} u(i,t).$$

The utility of X, depicted by u (X), is the sum of the utility of X in every transaction carrying X, that is,

$$u(X) = \sum_{t \in TS(X)} u(X,t) = \sum_{t \in TS(X)} \sum_{i \in X} u(i,t)$$

A pattern is a high utility pattern if its utility is no less than a user specified minimum utility threshold. High utility pattern mining is to discover all high utility patterns, that is,

$$HUPset = \{X | X \subseteq I, u(X) \geq minU\}$$

Consider an example of a super market database.

Table 1: Database D and eXternal Utility Table (XUT)

(a) D: shopping transactions

TID	ITEM						
	a	b	c	d	e	f	g
t_1	1		1		1		
t_2	6	2	2			5	
t_3	1	1	1	2	6		5
t_4	3	1		4	3		
t_5	2	1		2		2	

(b) XUT: prices

ITEM	PRICE
a	1
b	3
c	5
d	2
e	2
f	1
g	1

Table 1 (a) lists the quantity of each product in each shopping transaction. Here, I = {a, b, c, d, e, f, g} and D ={t_1, t_2, t_3, t_4, t_5}. Table 1 (b) lists the price (weight) of each product. For a transaction t_4, iu (a,t_4) = 3, iu (b,t_4) = 1, iu (d,t_4) = 4, iu (e,t_4) = 3 and eu(a) = 1, eu(b) = 3, eu(d) = 2, and eu(e) = 2. Here, u (i, t) is the product of iu (i, t) and eu (i). Thus, u (a, t_4) = 3, u (b, t_4) = 3, u (d, t_4) = 8 and u (e, t_4) = 6. Suppose the manager wants to know every combination of products with sales revenue no less than 30, that is, minimum utility = 30. Since TS ({a, b}) = {t_2, t_3, t_4, t_5}, we have, u ({a, b}) = u ({a, b}, t_2) + u ({a, b}, t_3) + u ({a, b}, t_4) + u ({a, b}, t_5) = u (a, t_2) + u (b, t_2) + u (a, t_3) + u (b, t_3) + u (a, t_4) + u (b, t_4) + u (a, t_5) + u (b, t_5) = 27. Similarly, u ({a, c}) = 28, u ({b, c}) = 24, u ({a, b, c}) = 31, u ({a, b, c, d}) = 13, and so on. Therefore, HUPset = {{a, b, c}, {a, b, d}, {a, d, e}, {a, b, d, e}, {b, d, e}, {d, e}, {a, b, c, d, e, g}}.

A standard method to mine high utility patterns is to enumerate each subset of I and test those subsets have a utility over the threshold. However such an enumeration is infeasible due to the large number of subsets of I. So, a high utility pattern growth approached is introduced, in which a reverse set enumeration tree is used and a pruning technique is used to reduce the number of patterns to be enumerated. The pattern growth approach can be viewed as searching a reverse set enumeration tree in a depth-first manner. The following figure represents a reverse set enumeration tree.

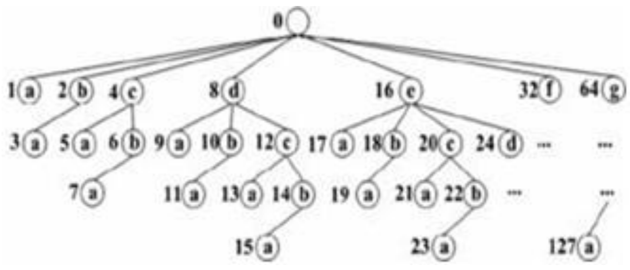


Figure 1: Reverse set enumeration tree

The construction of the reverse set enumeration tree follows an imposed ordering Ώ of items. The root is labeled by no item, each node N other than the root is labeled by an item, denoted by, item (N). The path from the node N to the root represent a pattern, denoted by pat (N). Child nodes of N are labeled by items listed before item (N) in Ώ. The imposed ordering of items, denoted by Ώ, is an ordered sequence of all the items in I. For items i and j, i $\prec$ j denotes that i is listed before j.

It is computationally infeasible to enumerate all patterns, so a standard solution is to prune the search space. Here the pruning is based on utility upper bounding [11, 12]. With the pattern growth approach, the method is to approximate an upper bound on utilities of all viable patterns represented by nodes in the sub tree rooted at the node currently being explored, when growing the reverse set enumeration tree. If such an upper bound is less than minimum utility, the sub tree itself can be pruned as all the patterns in the sub tree are not high utility patterns.

A pattern Y represented by a node C in the sub tree rooted at a node N is a prefix extension of pattern X represented by N, which leads to a way to estimate an upper bound on the utility of Y. Given an ordering Ώ, a pattern Y is a prefix extension of a pattern X, if X is a suffix of Y, that is, if Y = W ∨ X for some W with W $\prec$ X in Ώ.

Given an ordering Ώ, a pattern Y is a full prefix extension of a pattern X w.r.t. a transaction t containing X, denoted as, Y = fpe(X, t), if Y is a prefix extension of X derived by adding exactly all the items in t that are listed before X in Ώ. The basic upper bounds are defined by, for a pattern X, the sum of the utility of full prefix extension of X w.r.t each transaction in TS(X), denoted by, $uB_{fpe}(X)$, is no less than the utility of any prefix extension Y of X, that is,

$$uB_{fpe}(X) = \sum_{t \in TS(X)} u(fpe(X,t),t) \geq u(Y)$$

The algorithm used for mining patterns in single phase is the d^2HUP, Direct Discovery of High Utility Pattern. It is an integration of the depth-first search of reverse set enumeration tree and the pruning strategy that reduces the number of patterns to be enumerated. The algorithm builds the transaction set (TS) by scanning the database (D) and external utility table (XUT) to compute support, utility and upper bound for each item i. The algorithm starts finding high utility patterns from the root of the reverse set enumeration tree by calling the depth-first strategy, DFS(N, TS(pat(N)), minU, Ω). For the node N currently being visited, DFS print pat (N) as a high utility pattern if its utility is no less than the threshold [1].

Along with mining high utility patterns in single phase, a method for mining top-k high utility item sets are also introduced, where k is the desired number of item sets to be mined. It uses the basic search procedure of HUI-miner and its utility-list structure [24]. Whenever an item set is generated, its utility is calculated by its utility list without scanning the original database. Pruning the search space for HUI mining is difficult because the super set of a low utility item set can be high utility. To handle this problem, the concept of Transaction Weighted Utility (TWU) model was introduced [25]. In this model, an item set is called high transaction weighted utilization item set (HTWUI) if its TWU is no less than the minimum utility, where the TWU of an item set indicates an upper bound on its utility. Therefore, a HUI must be a HTWUI.

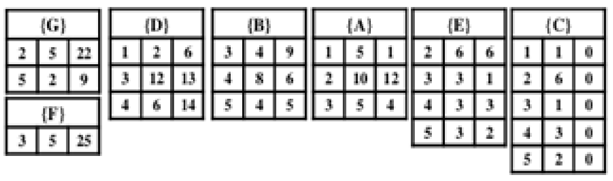
In this method, each item set is associated with a utility-list. The utility lists of item sets are called initial utility-lists, which can be constructed by scanning the database twice. In the first database scan TWU and utility values of each item are calculated. During the second scan, items in each transaction are sorted in order of TWU values and the utility-list of each item is constructed.

Table 2 shows an example database, where items in each transaction are arranged in the ascending order of TWU value. Figure 2 shows the utility-list of items for thedatabase in Table 2. The utility-list of an item X consists of one or more tuples. Each tuple represents the information of

X in a transaction T, and has three fields: Tid, iutil, rutil. Fields Tid and iutil respectively contains the identifier of T and utility of X in T. Field rutil indicates the remaining utility of X in T.

Table 2: Transactions for constructing utility-list

TID	Transaction	Transaction Utility (TU)
T_1	(D,1)(A,1)(C,1)	8
T_2	(G,5)(A,2)(E,2)(C,6)	27
T_3	(F,5)(D,6)(B,2)(A,1)(E,1)(C,1)	30
T_4	(D,3)(B,4)(E,1)(C,3)	20
T_5	(G,2)(B,2)(E,1)(C,2)	11



{G}		
2	5	22
5	2	9

{F}		
3	5	25

{D}		
1	2	6
3	12	13
4	6	14

{B}		
3	4	9
4	8	6
5	4	5

{A}		
1	5	1
2	10	12
3	5	4

{E}		
2	6	6
3	3	1
4	3	3
5	3	2

{C}		
1	1	0
2	6	0
3	1	0
4	3	0
5	2	0

Figure 2: Initial utility lists

The algorithm takes as input the parameter k and transactional database D in horizontal format. If the database have been transformed into vertical format such as initial utility-lists, the algorithm can directly use it for mining top-k HUIs. In top-k HUI mining, no minimum utility threshold is provided in advance. Therefore the minimum utility threshold is initially set to zero and the algorithm has to gradually raise the threshold to prune the search space. Such a threshold is an internal parameter of the algorithm, called border minimum utility threshold min_util_{Border}. The algorithm initially sets the threshold to zero and maintains a heap structure for maintaining the current top-k HUIs during the search. The algorithm then scans the database twice to build the initial utility lists. During the search, the algorithm updates the list of current top-k HUIs and gradually raises the min_util_{Border} threshold. When the algorithm terminates, the heap structure captures the complete set of top-k HUIs in the database. Initially, the heap structure is empty. When an item set X is found during the search procedure and its utility is no less than the min_util_{Border}, X is added to the heap structure. If there are more than k item sets already in the heap structure, min_util_{Border} can be raised to the utility of the k-th item set. After that item sets having a utility lower than the raised min_util_{Border} are removed from the heap structure.

The method discussed in this paper make use of mining high utility patterns in single phase and use the concept of utility-list for mining top-k high utility item sets.

4. Conclusions

Utility mining is a growing trend of data mining technology. Prior works on utility mining all employ a two-phase, candidate generation approach that is inefficient and not scalable with large databases. The two phase approach results in scalability issue due to the large number of candidates. This paper propose an algorithm for utility mining which finds high utility patterns in single phase. A high utility pattern growth approach is described, which is the combination of a pattern enumeration strategy and a pruning technique by utility upper bounding. It reduces the search space and scan time. This paper also concerns the problem of top-k high utility item sets mining, where k is the desired number of item sets to be mined. This method mines the item sets without setting minimum utility threshold.

References

[1] Junqiang Liu, Ke Wang, and Benjamin C.M. Fung, "Mining high utility patterns in one phase without generating candidates," in transactions on knowledge and data engineering, 2015.
[2] Vincent S. Tseng, Cheng-Wei Wu, Philip S. Yu, "Efficient algorithms for mining top-k high utility item sets," in transactions on knowledge and data engineering, 2015.
[3] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in SIGMOD.ACM, 1993, pp. 207–216.
[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in VLDB, 1994, pp. 487-499.

[5] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in SIGMOD.ACM, 2000, pp. 1–12.
[6] M. J. Zaki, "Scalable algorithms for association mining," IEEE TKDE, vol. 12, no. 3, pp. 372–390, 2000.
[7] A. Savasere, E. Omiecinski, and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," in VLDB. ACM, 1995, pp. 432–444.
[8] M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen, "A perspective on databases and data mining," in KDD. ACM, 1995, pp. 150–155.
[9] C. Bucila, J. Gehrke, D. Kifer, and W. M. White, "Dualminer: A dual-pruning algorithm for item sets with constraints," Data Mining and Knowledge Discovery, vol. 7, no. 3, pp. 241–272, 2003
[10] L. De Raedt, T. Guns, and S. Nijssen, "Constraint programming for item set mining," in SIGKDD, 2008, pp. 204–212.
[11] R. Bayardo and R. Agrawal, "Mining the most interesting rules," in SIGKDD. ACM, 1999, pp. 145–154.
[12] S. Morishita and J. Sese, "Traversing item set lattice with statistical metric pruning," in PODS. ACM, 2000, pp. 226–236.
[13] K. Chuang, J. Huang and M. Chen, "Mining Top-K Frequent Patterns in the Presence of the MemoryConstraint," The VLDB Journal, Vol. 17, pp. 1321-1344, 2008.
[14] G. Pyun and U. Yun, "Mining Top-K Frequent Patterns with Combination Reducing Techniques," Applied Intelligence, Vol. 41(1), pp. 76-98, 2014.
[15] T. Quang, S. Oyanagi, and K. Yamazaki, "ExMiner: An Efficient Algorithm for Mining Top-K Frequent Patterns," in Proc. of Int'l Conf. on Advanced Data Mining and Applications, pp. 436 – 447, 2006.
[16] J. Wang and J. Han, "TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Item sets," IEEE Transactions on Knowledge and Data Engineering, Vol. 17(5), pp. 652-664, 2005.
[17] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-K Closed Sequential Patterns," Knowledge and Information System, Vol. 7(4), pp. 438-457, 2005.
[18] P. Fournier-Viger, C. Wu, V. S. Tseng, "Mining Top-K Association Rules," in Proc. of Int'l Conf. on Canadian conference on Advances in Artificial Intelligence,pp. 61–73, 2012.
[19] 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.
[20] R. Chan, Q. Yang and Y. Shen, "Mining High-utility Item sets," in Proc. of IEEE Int'l Conf. on Data Mining, pp. 19-26, 2003.
[21] M. Zihayat and A. An, "Mining Top-K High Utility Item sets over Data Streams," Information Sciences, Vol. 285 (20), pp. 138–161, 2014.
[22] J. Yin, Z. Zheng, L. Cao, Y. Song and W. Wei, "Mining Top-K High Utility Sequential Patterns," in Proc. of IEEE Int'l Conf. on Data Mining, pp. 1259-1264, 2013.
[23] H. Ryang and U. Yun, "Top-K High Utility Pattern Mining with Effective Threshold Raising Strategies," Knowledge-Based Systems, Vol. 76, pp. 109-126, 2015.
[24] M. Liu and J. Qu, "Mining High Utility Item sets without Candidate Generation," in Proc. of ACM Int'l Conf. on Information and Knowledge Management, pp. 55-64, 2012.
[25] Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Item sets Mining Algorithm", in Proc. of the Utility-Based Data Mining Workshop, pp. 90-99, 2005.