Expert Systems with Applications 42 (2015) 2371–2381

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Pruning strategies for mining high utility itemsets

Srikumar Krishnamoorthy *

Indian Institute of Management, Ahmedabad, India

ARTICLE INFO

Article history:
Available online 8 November 2014

Keywords:
High utility itemsets
Frequent itemsets
Data mining

ABSTRACT

High utility itemset mining problem involves the use of internal and external utilities of items (such as profits, margins) to discover interesting patterns from a given transactional database. It is an extension of the basic frequent itemset mining problem and is proven to be considerably hard and intractable. This is due to the lack of inherent structural properties of high utility itemsets that can be exploited. Several heuristic methods have been suggested in the literature to limit the large search space. This paper aims to improve the state-of-the-art and proposes a high utility mining method that employs novel pruning strategies. The utility of the proposed method is demonstrated through rigorous experimentation on several real and synthetic benchmark sparse and dense datasets. A comparative evaluation of the method against a state-of-the-art method is also presented. Our experimental results reveal that the proposed method is very effective in pruning unpromising candidates, especially for sparse transactional databases.

1. Introduction

High utility itemset mining is one of the important problems that has received significant attention in the last several years. This is largely due to its potential applicability in several business and scientific applications. A high utility mining system provides flexibility to a decision maker to incorporate her/his notion of item utilities (profit, margin and so on) into the itemset mining process. As a result, the discovered patterns are highly likely to be of interest to the decision maker. On the other hand, traditional frequent set mining (Agrawal et al., 1994) methods largely rely on item frequencies. The high utility mining, therefore, can be considered as an extension or generalized version of frequent set mining. While the former method uses a generalized utility function, the latter method uses item support or frequencies as the utility function during the mining process.

Frequent set mining methods leverage the anti-monotonic property of support for efficient mining. However, a high utility itemset do not satisfy anti-monotone property (Liu, Liao, & Choudhary, 2005; Yao & Hamilton, 2006). This makes the high utility mining problem considerably hard and intractable. Several algorithms for high utility mining have been proposed in the literature. These algorithms can be broadly classified as level-wise candidate generation and test approach (Liu et al., 2005; Li, Yeh, & Chang, 2008; Yao & Hamilton, 2006), tree-based approach (Ahmed, Tanbeer, Jeong, & Lee, 2009; Ahmed, Tanbeer, Jeong, & Lee, 2011; Tseng, Shie, Wu, & Yu, 2012) and depth-first approach (Liu & Qu, 2012). Most of these algorithms use the concept of *transaction weighted utility* (Liu et al., 2005) to minimize the number of utility computations made during the mining process. However, the *transaction weighted utility* is known to overestimate the true utility of an itemset. This leads to a lot of wasted utility computations for itemsets that eventually do not satisfy the minimum utility threshold. This paper aims to address the foregoing limitations with the help of a new high utility mining system that employs several pruning strategies.

The key contributions of this paper are as follows: We present a method for efficiently discovering high utility itemsets. The proposed method employs two novel pruning strategies, namely partitioned utility pruning and lookahead utility pruning. We demonstrate the usefulness of the proposed method through rigorous experimental evaluation on several real and synthetic benchmark sparse and dense datasets. Furthermore, we present comparative evaluation of the method against a state-of-the-art utility mining system and report our findings.

The rest of the paper is organized as follows. Section 2 describes the related work on high utility mining. Section 3 defines the problem and describes the key definitions and notations used in this paper. Section 4 presents the proposed data structure and pruning strategies. Section 5 outlines our high utility mining method. Subsequently, Section 6 provides detailed experimental analysis and results. Finally, Section 7 gives concluding remarks.

* Tel.: +91 79 6632 4834.
E-mail address: srikumark@iimahd.ernet.in

http://dx.doi.org/10.1016/j.eswa.2014.11.001
0957-4174/

2. Related literature

Liu et al. (2005) use the concept of *transaction weighted utility* (TWU) for mining high utility itemsets. Their two-phase algorithm mines high utility itemsets in a two step process. In the first step, the algorithm exploits the anti-monotonic property of TWU of itemsets to mine all high TWU itemsets. Then, in the second step, actual utilities of itemsets are computed and low utility itemsets are discarded. The algorithm suffers from scalability issues due to its iterative level-wise candidate generation and test methodology.

UMining and UMining_H algorithms proposed by Yao and Hamilton (2006) utilizes two pruning strategies, namely utility upper bound and support upper bound. While the former strategy is guaranteed to generate all itemsets, the latter strategy is heuristically oriented and may erroneously prune genuine itemsets. The pruning properties exploited by the algorithm require a level-wise mining. Therefore, it may not be easily translatable to more efficient depth oriented utility mining approaches. It also suffers from scalability issues due to its level-wise candidate generation, prune and test methodology. FUM and DCG+ (Li et al., 2008) are level-wise utility mining algorithms that use an isolated items discarding strategy (IIDS) to limit the number of candidates generated and tested.

GPA algorithm (Lan, Hong, & Tseng, 2012) presents an extension to the basic concepts of Two-phase algorithm (Liu et al., 2005). The algorithm follows a level wise mining approach and iteratively generates a tighter utility upper bound. It removes unwanted items iteratively before performing utility computation to determine a tighter upper bound. In addition, the algorithm employs a transaction size reduction strategy to combine duplicate transactions generated during the mining process. The authors demonstrate that GPA algorithm performs much better than a Two-phase algorithm both on pruning and execution efficiency.

PB algorithm (Lan, Hong, & Tseng, 2014) uses special indexing structures and projection based methods to efficiently mine high utility itemsets. The authors demonstrate that PB algorithm is much more computationally efficient compared to Two-phase and CTU-PRO (Erwin, Gopalan, & Achuthan, 2007).

Tree based algorithms that mine high utility itemsets without expensive candidate generation and test methodology include IHUP (Ahmed et al., 2009), HUC-Prune (Ahmed et al., 2011), and UP-Growth+ (Tseng et al., 2012). IHUP (Ahmed et al., 2009) is experimentally proven to be better than Two-phase (Liu et al., 2005), FUM (Li et al., 2008), and DCG+ (Li et al., 2008).

HUI-Miner is one of the recent and most efficient depth-first algorithms proposed by Liu and Qu (2012). The authors introduced a new data structure called utility lists which is similar to the tid lists used in Eclat algorithm for mining frequent itemsets (Zaki, 2000). The information captured in utility lists during the mining process is exploited to limit the overall search space. The authors demonstrate that their method is superior to IHUP (Ahmed et al., 2009; Tseng, Wu, Shie, & Yu, 2010) and UP-Growth+ (Tseng et al., 2012). The authors show almost two orders of magnitude improvement over other methods in the literature.

Liu, Wang, and Fung (2012) proposed d^2HUP algorithm for efficiently mining high utility itemsets. The authors used the concepts of irrelevant item filtering and lookahead pruning for efficient mining. The concept of irrelevant item filtering involves iteratively eliminating the irrelevant items from the utility computation process. d^2HUP (Liu et al., 2012) and HUI-Miner (Liu & Qu, 2012) algorithms were introduced by different sets of authors at almost around the same time. However, one can observe that the key pruning strategies adopted by both of these algorithms are conceptually equivalent though the underlying data structures used are quite different. d^2HUP also uses the concept of lookahead strategy for efficient utility mining in the case of dense datasets. It is to be noted that the final high utility itemsets generated (with lookahead pruning) are not complete. An additional iteration of the generated itemsets will be required to enumerate the actual high utility itemsets and their utility values. The authors demonstrate that their algorithm is up to an order of magnitude faster than UP-Growth (Tseng et al., 2010).

An interesting extension to the basic utility mining problem is the high on-shelf utility mining problem. This problem was introduced by Lan, Hong, and Tseng (2011). The authors design a periodic total transaction utility table and a new pruning strategy based on on-shelf utility measure. The proposed method is shown to be better than a traditional high utility itemset mining on synthetic benchmark datasets.

This research work builds on the extant literature and presents new pruning strategies for efficiently mining high utility itemsets.

3. Definition and notation

We formally define the key terms in utility mining using the standard conventions followed in the literature (Ahmed et al., 2009; Liu & Qu, 2012; Liu et al., 2005; Yao & Hamilton, 2006).

Let $I = \{i_1, i_2 \ldots i_m\}$ be a set of distinct items. A set $X \subseteq I$ is called an itemset. A transaction $T_j = \{x_l | l = 1, 2 \ldots N_j, x_l \in I\}$, where N_j is the number of items in transaction T_j. A transaction database D has set of transactions, $D = \{T_1, T_2 \ldots T_n\}$, where n is the total number of transactions in the database. A sample transaction database D is given in Table 1.

Definition 1. Each item $x_i \in I$ is assigned an external utility value (e.g. profit), referred as $EU(x_i)$. For example, in Table 1, $EU(B) = 2$.

Definition 2. Each item $x_i \in T_j$ is assigned an internal utility value, referred as $IU(x_i, T_j)$. For example, in Table 1, $IU(B, T_1) = 1$.

Definition 3. The utility of an item $x_i \in T_j$, denoted as $U(x_i, T_j)$ is computed as the product of external and internal utilities of item in the transaction, T_j. That is,

$$U(x_i, T_j) = EU(x_i) * IU(x_i, T_j) \quad (1)$$

For example, in Table 1, $U(B, T_1) = EU(B) * IU(B, T_1) = 2 * 1 = 2$.

Definition 4. The utility of an itemset X in transaction T_j ($X \subseteq T_j$) is denoted as $U(X, T_j)$.

$$U(X, T_j) = \sum_{x_i \in X} U(x_i, T_j) \quad (2)$$

For example, in Table 1, $U(ABD, T_1) = 13$.

Table 1
Purchase history.

TID	Transaction	Purchase qty (IU)	Utility (U)	Transaction utility (TU)
T_1	A, B, D, F, G	1, 1, 2, 1, 2	1, 2, 10, 3, 2	18
T_2	D, H	1, 1	5, 2	7
T_3	A, B, F, G	2, 2, 2, 1	2, 4, 6, 1	13
T_4	B, C, D, E, F	1, 1, 1, 2, 2	2, 1, 5, 8, 6	22
T_5	D, F	2, 1	10, 3	13
T_6	B, C, E, F	1, 1, 2, 1	2, 1, 8, 3	14
T_7	A, G	3, 2	3, 2	5
T_8	C, E, G	2, 1, 3	2, 4, 3	9
T_9	A, B, C, D, E, G	2, 1, 2, 1, 1, 3	2, 2, 2, 5, 4, 3	18
T_{10}	B, D, E, G	2, 1, 2, 2	4, 5, 8, 2	19
T_{11}	F, G, I	1, 1, 1	3, 1, 1	5
T_{12}	B, C	1, 5	2, 5	7