Advanced Engineering Informatics 30 (2016) 233–243

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Full length article

An efficient algorithm to mine high average-utility itemsets

Jerry Chun-Wei Lin [a,*], Ting Li [a], Philippe Fournier-Viger [b], Tzung-Pei Hong [c,d], Justin Zhan [e], Miroslav Voznak [f]

[a] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China
[b] School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China
[c] Department of Computer Science and Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
[d] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan
[e] Department of Computer Science, University of Nevada, Las Vegas, USA
[f] Department of Telecommunications, VSB-Technical University of Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 24 December 2015
Received in revised form 26 March 2016
Accepted 3 April 2016
Available online 15 April 2016

Keywords:
High average-utility itemsets
List structure
Data mining
HAUIM

ABSTRACT

With the ever increasing number of applications of data mining, high-utility itemset mining (HUIM) has become a critical issue in recent decades. In traditional HUIM, the utility of an itemset is defined as the sum of the utilities of its items, in transactions where it appears. An important problem with this definition is that it does not take itemset length into account. Because the utility of larger itemset is generally greater than the utility of smaller itemset, traditional HUIM algorithms tend to be biased toward finding a set of large itemsets. Thus, this definition is not a fair measurement of utility. To provide a better assessment of each itemset's utility, the task of high average-utility itemset mining (HAUIM) was proposed. It introduces the average utility measure, which considers both the length of itemsets and their utilities, and is thus more appropriate in real-world situations. Several algorithms have been designed for this task. They can be generally categorized as either level-wise or pattern-growth approaches. Both of them require, however, the amount of computation to find the actual high average-utility itemsets (HAUIs). In this paper, we present an efficient average-utility (AU)-list structure to discover the HAUIs more efficiently. A depth-first search algorithm named HAUI-Miner is proposed to explore the search space without candidate generation, and an efficient pruning strategy is developed to reduce the search space and speed up the mining process. Extensive experiments are conducted to compare the performance of HAUI-Miner with the state-of-the-art HAUIM algorithms in terms of runtime, number of determining nodes, memory usage and scalability.

1. Introduction

Mining frequent itemsets (FIs) or association rules (ARs) in transactional databases is a fundamental task in knowledge discovery in databases (KDD) [2,3,6]. Many algorithms have been designed to mine FIs or ARs. The most common ways of deriving FIs or ARs from a database are to use a level-wise [3] or a pattern-growth approach [8,14]. Apriori [3] is the first algorithm to mine FIs in a level-wise manner. It relies on a minimum support threshold in the first phase to mine FIs, and then use the discovered FIs in the second phase to derive ARs satisfying a minimum confidence threshold. The pattern-growth approach was introduced by Han et al. [8] for mining FIs without candidate generation. FP-growth initially builds an FP-tree structure using frequent 1-itemsets. Then, during the mining process, conditional FP-trees are recursively generated, and each tree contains a designed index table (Header_Table) for mining the FIs.

Traditional frequent itemset mining (FIM) and association rule mining (ARM) algorithms only consider occurrence frequencies of items in binary databases. Other important factors such as quantities, profits, and weights of items are not taken into account by traditional FIM and ARM algorithms. Another problem is that FIs and ARs found a transaction database may only contribute a small portion of the overall profit generated by the sale of items, and infrequent itemsets may contribute a large amount of the profit. For example, the sale of diamonds may be less frequent than that of clothing or shoes in a shopping mall, but diamonds generally contribute a much higher profit per unit sold. It is thus obvious that only considering the occurrence frequency is insufficient to

* Corresponding author.
E-mail addresses: jerrylin@ieee.org (J.C.-W. Lin), tingli@ikelab.net (T. Li), philfv@hitsz.edu.cn (P. Fournier-Viger), tphong@nuk.edu.tw (T.-P. Hong), justin.zhan@unlv.edu (J. Zhan), miroslav.voznak@vsb.cz (M. Voznak).

http://dx.doi.org/10.1016/j.aei.2016.04.002
1474-0346/© 2016 Elsevier Ltd. All rights reserved.

identify highly profitable itemsets or itemsets that are generally more important to the user. Thus, high-utility itemset mining (HUIM) [11–13,22] has emerged as a critical issue in recent decades, as it can reveal the profitable itemsets in real-world situations. HUIM can be considered as an extension of FIM that considers additional information such as quantities and unit profits of items, to better assess how "useful" an itemset is to the user. An item/set is considered as a high-utility itemset (HUI) if its utility is no less than a user-defined minimum utility threshold. Since the downward closure (DC) property used in traditional FIM and ARM does not hold in traditional HUIM, Liu et al. [12] designed a two-phase approach and developed a transaction-weighted downward closure (TWDC) property to reduce the search space by pruning unpromising itemsets early. Several level-wise and pattern-growth algorithms have been proposed to efficiently mine HUIs, using the two-phase approach [4,5,7].

In traditional HUIM, the utility of an item/set is defined as the sum of its utilities in the database. An important problem with this definition is that it does not take itemset length into account. Thus, this definition is not a fair measurement of utility. To provide a better assessment of each itemset's utility, the task of high average-utility itemset mining (HAUIM) was proposed by Hong et al. [9]. The proposed average utility measure estimates the utility of an itemset by considering its length. It is defined as the sum of the utilities of the itemset in transactions where it appears, divided by the number of items that it contains. This measure addresses the bias of traditional HUIM toward larger itemsets, by considering the length of itemsets, and can thus more objectively assess the utility of itemsets. As for traditional HUIM, level-wise and pattern-growth algorithms have been designed for HAUIM. Level-wise algorithms [9] require to generate numerous candidates for mining the actual high average-utility itemsets (HAUIs). Pattern-growth algorithms [15] require to recursively build conditional trees for mining HAUIs, which is quite time-consuming. In this paper, we first design an efficient average-utility (AU)-list structure and develop an algorithm named HAUI-Miner for mining HAUIs using a single phase. The key contributions of this paper are threefold.

1. We first design an efficient HAUI-Miner algorithm to mine high average-utility itemsets (HAUIs). It relies on a novel condensed average-utility (AU)-list structure. This structure only keeps information required by the mining process, thus compressing very large databases into a condensed structure.
2. An efficient pruning strategy is developed to reduce the search space, represented as an enumeration tree, by pruning unpromising candidates early. Using this strategy, building the AU-lists of extensions of a processed node in the enumeration tree can be avoided to reduce the amount of computation.
3. Substantial experiments are conducted to compare the performance of the designed HAUI-Miner algorithm with the state-of-the-art algorithms, in terms of runtime, number of determining nodes, memory consumption, and scalability.

2. Related work

High-utility itemset mining (HUIM) [12,13,22], an extension of frequent itemset mining, is based on the measurement of internal utility and external utility. The internal utility of an item is its purchase quantity in a transaction, and the external utility of an item can be viewed as its unit profit, importance or weight. The utility of an item/set in a database is calculated as the total purchase quantity of the itemset in the database, multiplied by its unit profit (external utility). The purpose of HUIM is to discover the complete set of high-utility itemsets (HUIs), that are itemsets having a utility no less than a minimum utility threshold. Yao et al. [22] proposed a framework for mining HUIs based on mathematical properties of the utility measure. Two pruning strategies were designed to reduce the search space for discovering HUIs respectively based on utility upper bounds and expected utility upper bounds. Since the downward closure (DC) property of ARM does not hold in traditional HUIM, Liu et al. [12] then designed a transaction-weighted downward closure (TWDC) property and developed the transaction-weighted utilization (TWU) model. This latter provides upper bounds on the utilities of potential HUIs, which can be used to reduce the combinatorial explosion of the search space in traditional HUIM. However, the TWU model still requires to generate numerous candidates to obtain the actual HUIs. Pattern-growth algorithms have been proposed to compress the database into a condense tree structure using the TWU model. Lin et al. [16] designed a high-utility pattern (HUP)-tree algorithm to recursively mine high-utility itemsets using the proposed tree structure. Tseng et al. developed the UP-Growth [20] and UP-Growth+ [21] algorithms to efficiently discover HUIs based on different pruning strategies. The aforementioned approaches all rely on the TWU model and its TWDC property for discovering HUIs. The search space is, however, very large when using the TWU model, and it is thus very time-consuming to discover the actual HUIs. As an alternative to the pattern-growth mechanism, Liu et al. [13] developed the list-based HUI-Miner algorithm to discover HUIs without candidate generation. The developed utility-list structure is an efficient structure for maintaining the information required for mining HUIs using a limited amount of memory. Fournier-Viger et al. [7] extended HUI-Miner with a structure named EUCS to store information about the relationships between 2-itemsets, thus speeding up the discovery of HUIs. Several extensions of the task of HUIM have been proposed such as discovering up-to-date HUIs [17] and top-k HUIs [23].

Similarly to traditional HUIM, several HAUIM algorithms have been designed using the TWU model. Lin et al. [15] first developed the HAUP-tree structure and the HAUP-growth algorithm for mining HAUIs. In the HAUP-tree, each node at the end of a path stores the average-utility upper bound of the corresponding item as well as the quantities of the preceding items in the same path. This approach can thus be used to speed up the discovery of HAUIs. Lan et al. [10] proposed a projection-based average-utility itemset mining (PAI) algorithm to reveal HAUIs using a level-wise approach. Based on the proposed upper-bound model, the number of unpromising candidates can be greatly reduced compared to previous work based on the TWU model. Lu et al. [18] proposed the HAUI-tree algorithm to further reduce the number of unpromising candidates for mining the actual HAUIs using a designed enumeration tree structure. However, mining HAUIs using the designed algorithm is still very time-consuming since the upper-bounds used by these algorithms are loose, and thus numerous unpromising candidates need to be generated, and the recursive process for building the complete enumeration tree remains costly.

3. Preliminaries and problem statement

3.1. Preliminaries

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items. A quantitative database is a set of transactions $D = \{T_1, T_2, \ldots, T_n\}$, where each transaction $T_q \in D$ $(1 \leqslant q \leqslant m)$ is a subset of I and has a unique identifier q, called its *TID*. Besides, each item i_j in a transaction T_q has a purchase quantity denoted as $q(i_j, T_q)$. A profit table PT indicates the unit profit value of each item in the database as $PT = \{pr(i_1), pr(i_2), \ldots, pr(i_m)\}$, where profit values are positive integers. A set of k distinct items $X = \{i_1, i_2, \ldots, i_k\}$ such that $X \subseteq I$ is said to