Engineering Applications of Artificial Intelligence 72 (2018) 136–149

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Efficiently updating the discovered high average-utility itemsets with transaction insertion

Jerry Chun-Wei Lin[a,b,*], Shifeng Ren[a], Philippe Fournier-Viger[c], Jeng-Shyan Pan[d], Tzung-Pei Hong[e,f]

[a] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
[b] Department of Computing, Mathematics, and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
[c] School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
[d] Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fujian, China
[e] Department of Computer Science and Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
[f] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan

ARTICLE INFO

Keywords:
High average-utility itemsets
FUP
Dynamic
Insertion
Incremental

ABSTRACT

High-utility itemset mining (HUIM) is an extension of frequent-itemset mining (FIM) but considers the unit profit and quantity of items to discover the set of high-utility itemsets (HUIs). Traditionally, the utility of an itemset is the summation of the utilities of the itemset in all the transactions regardless of its length. This approach is, however, inappropriate in real-world applications since the utility of the itemset increases along with the number of items within it. High average-utility itemset mining (HAUIM) was designed to provide more reasonable utility measure by taking the size of the itemset into account. Existing algorithms can only handle, however, the static database and unsuitable for the dynamic environment since the size of data is frequently changed in real-life situations. In this paper, an incremental high-average utility pattern mining (IHAUPM) algorithm is presented to handle the incremental database with transaction insertion. The well-known fast updated (FUP) concept in the FIM is modified to adopt the designed algorithm, thus efficiently updating the discovered HAUIs. Based on the designed model for HAUIM with transaction insertion, the proposed IHAUPM algorithm can easily only handle the inserted transactions. Experiments are carried on six datasets and the results showed that the designed algorithm has better performance than the state-of-the-art algorithms performing in the batch manner.

1. Introduction

Data mining or knowledge discovery can be referred to discover useful, potential, and implicit information from a very large database (Agrawal and Srikant, 1994; Agrawal et al., 1993; Fournier-Viger et al., 2017; Han et al., 2004; Srikant and Agrawal, 1995). The discovered information can thus be used to aid managers or decision makers for making the significant or efficient strategies. Based on different requirements in various applications, varied types of knowledge are respectively presented in different ways. Apriori algorithm (Agrawal and Srikant, 1994) was the first algorithm to mine the correlations between the purchase itemsets, which can be referred as association-rule mining (ARM). It uses the level-wise approach to generate-and-test candidates at each level for finding the relationships among the items. The set of frequent itemsets (FIs) is first discovered by the Apriori algorithm based on the pre-defined minimum support threshold. After that, the set of FIs is then combined together, forming the set of association rules (ARs) based on the pre-defined minimum confidence threshold. This approach requires the amounts of computation and easily causes the memory leakage if the size of transactions in the database is too long, the frequent pattern (FP)-growth (Han et al., 2004) algorithm was then further designed to mine the FIs based on its designed FP-tree structure. Several algorithms were widely studied to enhance the mining performance of the A priori-based (Agrawal et al., 1993; Chang and Lin, 2005; Park et al., 1995; Srikant and Agrawal, 1995) and tree-based (Agrawal et al., 2001; Chi and Lain, 2004; Han et al., 2004) algorithms.

Since the size of database is dynamically changed in real-life situations, such as some transactions could be inserted (Cheung et al., 1996) into the original one; the discovered information is necessary

[*] Corresponding author at: School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China .
E-mail addresses: jerrylin@ieee.org (J.C.-W. Lin), renshifeng@stmail.hitsz.edu.cn (S. Ren), philfv@hitsz.edu.cn (P. Fournier-Viger), jengshengpan@fjut.edu.cn (J.-S. Pan), tphong@nuk.edu.tw (T.-P. Hong).

https://doi.org/10.1016/j.engappai.2018.03.021
Received 30 September 2016; Received in revised form 31 August 2017; Accepted 23 March 2018

to be updated since some new information may be arisen but some information may be lost. Most algorithms of FIM or ARM can only, however, handle the static database. When the data is changed in the database, the algorithms must be performed in the batch manner, which indicates that the already discovered information is useless and updated database is required to be scanned and examined to mine the required information. To handle the situation of transaction insertion, Cheung et al. proposed the Fast UPdated (FUP) (Cheung et al., 1996) algorithm to adopt the pruning techniques used in the DHP (Direct Hashing and Pruning) algorithm (Park et al., 1997) for updating the discovered information. It divides the original databases and new inserted transactions into four parts and each part is performed by its own algorithm to update the required information. Since some unpromising itemsets in the FUP concept can be eliminated, the updating performance can thus be significantly improved.

In real-world situations, the frequent itemsets in FIM or ARM may only contribute a small portion to the overall profit, while non-frequent ones may contribute a large portion to the profit. High-utility itemset mining (HUIM) (Liu et al., 2005b; Yao et al., 2004, 2006) was thus proposed to concern more factors such as unit profit and quantity of the items to reveal the high-utility itemsets (HUIs), which is an extension of FIM and widely applied in real-world situations. The utility of an itemset is to sum up the utilities of the itemset in all transactions regardless of its length (Liu et al., 2005a; Yao et al., 2004). It is based on the measure of local transaction utility (quantity) and external utility (unit profit of items) to evaluate whether an itemset is a HUI. Liu et al. (2005b) then proposed the transaction-weighted utilization (TWU)-model to overestimate the upper value of the designed high-transaction-weighted utilization itemsets (HTWUIs), thus maintaining the transaction-weighted downward closure (TWDC) property for mining the complete HUIs. To speed up mining performance, Lin et al. then presented a high utility pattern (HUP)-tree algorithm (Lin et al., 2011) for mining the set of HUIs. Several algorithms were also extensively developed to mine the HUIs (Chan et al., 2003; Krishnamoorthy, 2015; Tseng et al., 2013; Yao et al., 2006).

In HUIM, the itemset within more number of items may increase its utility regardless of its length. Thus, it is unfair to judge whether the itemset is a HUI with different lengths according to the same user-specified minimum threshold. To solve this limitation, Hong et al. proposed a high average-utility itemset mining (HAUIM) (Hong et al., 2011) to alleviate the effect of the length of the itemset and identify a better utility effect by considering the size of the itemset than the original utility standard. The average-utility of an itemset is defined as the total utility of an itemset divided by its number of items within it. An itemset is called a high average-utility itemset (HAUI) if its average utility is no less than pre-defined minimum high average-utility count. Thus, HAUIM is a totally different issue compared to the traditional HUIM since the new downward closure property, upper-bound model, pruning strategy, as well as the mining procedure are required to be developed. Lin et al. presented a high average-utility pattern (HAUP)-tree algorithm (Lin et al., 2011) to mine the set of HAUIs based on the efficient tree structure. The projection-based algorithm called PAI (Lan et al., 2012) was also designed to speed up mining performance. Lin et al. (2016d) then designed a average-utility (AU)-list structure to efficiently mine the HAUIs from a static database, which is the state-of-the-art algorithm for mining HAUIs.

Since the previous works of HAUIM can only handle the static database, but in real-life situations, database size is dynamically changed; for example, some transactions are frequently inserted into the original database. As we mentioned above, some information may be thus arisen, but some information may be lost in the updated progress. Although the batch processing of the updated database can solve this problem, it causes, however, multiple database scans and the already discovered information become invalid and useless. In this paper, we propose an incremental algorithm to efficiently update the discovered HAUIs when some transactions are frequently inserted into the original database. Major contributions are summarized as follows.

- We modified FUP concept (Cheung et al., 1996) used in the FIM and ARM for updating the set of the discovered HAUIs. We have also maintained the correctness and completeness of the updated HAUIs to ensure that all the required information is completely discovered and updated.
- We adopted the efficient HAUP-tree structure to maintain the necessary information for later mining progress. An incremental IHAUPM algorithm is developed to divide the original database and new transactions into four cases for maintenance. The itemsets of each case can be correctly maintained and processed.
- The designed algorithm is sometimes unnecessary to multiply rescan the original database for updating the discovered HAUIs compared to the existing state-of-the-art algorithms running in the batch manner.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. The preliminaries and problem statement of the designed algorithm are stated in Section 3. The developed models and designed algorithm are described in Section 4. An illustrated example is provided in Section 5. Experiments are conducted and evaluated in Section 6. The conclusion and future work are finally given in Section 7.

2. Related work

In this session, the related works of incremental concept, the high-utility itemset mining (HUIM), and the high average-utility itemset mining (HAUIM) are respectively reviewed and discussed.

2.1. Incremental concept

Data mining techniques are used to extract meaningful and implicit patterns or rules from a very large dataset, and the most common approach is called association-rule mining (ARM). Agrawal and Srikant first proposed the Apriori algorithm (Agrawal and Srikant, 1994) to mine association rules (ARs) from a set of transactions. It applies the level-wise approach to first mine the frequent itemsets (FIs) based on the minimum support threshold and performs the combinational mechanism from the mined FIs to retrieve the ARs based on the minimum confidence threshold. Since the Apriori algorithm applies the level-wise approach, thus the huge amounts of runtime and memory usage are necessary. To solve this problem, Han et al. proposed the Frequent-Pattern-tree structure (FP-tree) structure (Han et al., 2004) and FP-growth algorithm for efficiently mining FIs without generation of candidate itemsets. To further improve the efficiency of FP-growth, Grahne and Zhu (2005) proposed a novel FP-array technique to greatly reduce traversing time of the FP-tree especially in the sparse datasets. Another NFP-tree structure (Chi and Lain, 2004) was also designed to reduce the number of tree nodes, as well as the size of Header_Table by keeping two counts of each tree node. Thus, fewer tree nodes are traversed in the tree structure to mine the FIs.

The above algorithms can only process the static database. In real-life situations, transactions usually grow over time and the mined information must be re-evaluated. For example in ARM, some new ARs may be generated and some old ones may become invalid when new transactions are inserted into the original database. The conventional algorithms are processed in the batch manner to re-process the entirely updated database whether the transactions are inserted, deleted or modified from the original database. This procedure requires the amounts of computational cost and waste the discovered knowledge. Cheung et al. thus proposed the Fast UPdated (FUP) concept (Cheung et al., 1996) to effectively handle the incremental database with transaction insertion. The FUP concept can be used to divide the original database and newly inserted transactions into four cases and each case is performed by its own procedure to maintain and update the discovered information. After all items in four cases are respectively performed, the up-to-date information is thus correctly maintained. The FUP concept can be