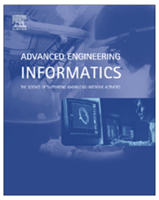
Advanced Engineering Informatics 29 (2015) 16–27

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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



Efficient updating of discovered high-utility itemsets for transaction deletion in dynamic databases ☆

Chun-Wei Lin [a,b,*], Tzung-Pei Hong [c,d], Guo-Cheng Lan [e], Jia-Wei Wong [d], Wen-Yang Lin [c]

[a] Innovative Information Industry Research Center (IIIRC), School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, HIT Campus Shenzhen University Town Xili, Shenzhen, PR China
[b] Shenzhen Key Laboratory of Internet Information Collaboration, School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, HIT Campus Shenzhen University Town Xili, Shenzhen, PR China
[c] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, ROC
[d] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, ROC
[e] Department of Mathematics and Computer Sciences, Fuqing Branch of Fujian Normal University, Fuzhou, Fujian, PR China

A R T I C L E I N F O

Article history:
Received 14 March 2014
Received in revised form 21 August 2014
Accepted 26 August 2014
Available online 18 September 2014

Keywords:
Utility mining
Pre-large concept
Transaction deletion
Two-phase approach
Dynamic databases

A B S T R A C T

Most algorithms related to association rule mining are designed to discover frequent itemsets from a binary database. Other factors such as profit, cost, or quantity are not concerned in binary databases. Utility mining was thus proposed to measure the utility values of purchased items for finding high-utility itemsets from a static database. In real-world applications, transactions are changed whether insertion or deletion in a dynamic database. An existing maintenance approach for handling high-utility itemsets in dynamic databases with transaction deletion must rescan the database when necessary. In this paper, an efficient algorithm, called PRE-HUI-DEL, for updating high-utility itemsets based on the pre-large concept for transaction deletion is proposed. The pre-large concept is used to partition transaction-weighted utilization itemsets into three sets with nine cases according to whether they have large (high), pre-large, or small transaction-weighted utilization in the original database and in the deleted transactions. Specific procedures are then applied to each case for maintaining and updating the discovered high-utility itemsets. Experimental results show that the proposed PRE-HUI-DEL algorithm outperforms a batch two-phase algorithm and a FUP2-based algorithm in maintaining high-utility itemsets.

1. Introduction

In various mining techniques [1–8], association rule mining is the most popular method used for knowledge discovery to find the relationships among items or products. The most common implementations use the Apriori algorithm [9] for generating and testing the candidate itemsets level by level. All frequent itemsets are first found based on a user-defined minimum support threshold and then the association rules are derived from the discovered frequent itemsets based on the user-defined minimum confidence threshold. In association rule mining, each item is treated as a binary variable to discover relationships among itemsets or products. The frequency of an itemset is, however, insufficient for identifying highly profitable itemsets with small sold quantities.

Utility mining [7,10,11] was thus proposed to solve the limitations of frequent itemsets. It may be thought of as frequent itemset mining with sold quantities and item profits concerned. In practice, the utility value of an itemset can be cost, profit, or some other term defined by the user. For example, itemsets with good profits or those with low pollution during manufacturing may be of interest. Liu et al. designed a two-phase algorithm [12] for efficiently extracting high-utility itemsets based on transaction-weighted utilization (TWU) to keep the downward closure property. The TWU is used as an effective upper bound to reduce the generation of candidates for later processing. An additional database scan is performed to determine the real utility values of the remaining candidates to identify high-utility itemsets. Most approaches [7,10,13,11] for handling high-utility itemsets are, however, processed in batch mode with a static database.

In real-world applications, databases tend to be large and dynamic since their contents are frequently changed whether the number of transactions are inserted or deleted. Previous

☆ Handled by C.-H. Chen.
* Corresponding author at: Innovative Information Industry Research Center (IIIRC), School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, HIT Campus Shenzhen University Town Xili, Shenzhen, PR China.
E-mail addresses: jerrylin@ieee.org (C.-W. Lin), tphong@nuk.edu.tw (T.-P. Hong), rrfoheiay@gmail.com (G.-C. Lan), jwwong.alex@gmail.com (J.-W. Wong), wylin@nuk.edu.tw (W.-Y. Lin).

http://dx.doi.org/10.1016/j.aei.2014.08.003
1474-0346/© 2014 Elsevier Ltd. All rights reserved.

discovered information may become invalid, or some new information may emerge in the updated database. Dynamic data mining can be referred as an updating technique to find the up-to-date information without re-scanning the original database and re-mining the desired information each time. Hong and Lin et al. proposed maintenance approaches for respectively maintaining the discovered frequent itemsets in dynamic databases based on Fast UPdated (FUP) [14,15] and pre-large concepts [16,17]. Lin et al. then extended the FUP concept to handle transaction insertion [18] for maintaining discovered high-utility itemsets.

Transactions are also frequently deleted in real-world applications. The FUP2 concept [19,20] for transaction deletion has been adopted for updating discovered high-utility itemsets [21]. The database, however, must be rescanned if a high-utility itemset was not large or pre-large both in the original database and in the deleted transactions. In this paper, a maintenance algorithm, called PRE-HUI-DEL, for transaction deletion based on the pre-large concept [22] and the TWU model [12] is proposed to maintain and update discovered high-utility itemsets in dynamic databases. The proposed PRE-HUI-DEL algorithm first partitions transaction-weighted utilization itemsets into three sets with nine cases according to whether they have large (high), pre-large, or small transaction-weighted utilization in the original database and in the deleted transactions. Specific procedures are then applied to each case to maintain and update the discovered transaction-weighted utilization itemsets. The major contributions of this paper are as follows:

1. Traditional utility mining processes a database in batch mode no matter whether transactions are deleted. For the original database, information was already discovered by data mining approaches. It is thus not efficient to waste the discovered information for updating the whole database with a small number of deleted transactions. In this paper, an efficient approach is proposed to handle transaction deletion for maintaining the discovered high-transaction-weighted utilization itemsets.
2. A two-phase model is used as an effective upper bound to reduce the generation of candidates, thus speeding up the processing time for updating the discovered information.
3. The upper bound utility and the lower bound utility are defined as the effective thresholds for respectively deriving high (large) and pre-large transaction-weighted utilization itemsets. Based on the two defined thresholds, the original database must be rescanned only if the transaction-weighted utilization of deleted transactions is more than the safety bound calculated from the two thresholds.
4. In the proposed algorithm, only a small number of itemsets must be rescanned to maintain the transaction-weighted utilization itemsets, reducing the computational load compared to those of the batch approach [12] and the FUP2-based approach [21].

2. Review of related work

In this section, association rule mining, the pre-large concept, and high-utility mining are briefly reviewed.

2.1. Association rule mining

Traditionally, data mining techniques [1,3] are used to derive desired information from a database. The most common approach is to find the association rules [9] from a binary database based on the fact that the presence of certain items in a transaction imply the presence of other items. Agrawal and Srikant proposed the Apriori algorithm [9] to level-wise discover association rules from a static database. Apriori uses the downward closure property to prune unpromising candidates, thus improving the efficiency of discovering frequent itemsets. The Apriori algorithm consists of two main parts for generating association rules. It first uses the generate-and-test approach to find all frequent itemsets, whose counts are larger than or equal to a user-specified threshold (called the minimum support). Each frequent itemset is then level-wise combined to form association rules whose confidence values are larger than or equal to the user-specified threshold (called the minimum confidence).

In real-world applications, databases grow over time and association rules are mined in batch mode. Some new association rules may be generated and some old ones may become invalid when transactions are inserted or deleted. Traditional batch mining algorithms solve this problem by rescanning the updated database when transactions are inserted or deleted, discarding previously discovered knowledge. Cheung et al. thus proposed the FUP [14] and FUP2 [19] algorithms to respectively handle transaction insertion and transaction deletion for maintaining and updating frequent itemsets. Hong et al. then respectively applied the FUP and FUP2 concepts to maintain the FP-tree structure for handling transaction insertion [15] and transaction deletion [20]. More related works for mining association rules for transaction insertion are reviewed elsewhere [23].

2.2. Pre-large concept

Most data mining techniques have been proposed to mine the desired information based on a minimum threshold. An itemset is considered as a frequent itemset if its ratio in database is larger than or equal to the minimum support threshold. When the ratio of an itemset is nearly close to but smaller than the minimum support threshold, it is still considered as an infrequent itemset. For example, a minimum support threshold is set at 50%, an itemset is concerned as an infrequent itemset if its support ratio is 49%, which is lower than 50%. When the transactions are changed in the database whether insertion or deletion, new information may be arisen or discovered information may be missed. It is a trivial way to handle the above situation by keeping all information from the transactional database. However, it is not an efficient mechanism to keep all information from a very large database for dynamic database based on the perspective of data mining. The FUP [14] and FUP2 [19] concepts were respectively proposed to update the discovered information with transaction insertion and transaction deletion. Although FUP2 concept can be used to efficiently update the discovered information, the original database is still required to be rescanned for handling the itemsets in case 4. Pre-large concept was proposed to keep more unpromising information as the buffer to avoid the limitations of database rescan for dynamic data mining with transaction insertion [16] and transaction deletion [22]. A pre-large itemset is not really large (frequent), but has a highly probability of becoming large (frequent) after data insertion [16] or deletion [22]. Two support thresholds are used to respectively find large and pre-large itemsets for reducing the number of database rescans. The pre-large itemset acts like a buffer to reduce the movement of an itemset directly from small to large and vice versa. Lin et al. then adopted the FP-tree structure [24] and the pre-large concept for maintaining discovered frequent itemsets [17,25,26] without candidate generation. Algorithms [16,22] based on this concept rescan the database only when a certain number of transactions are changed (inserted or deleted), thus improving the performance of knowledge maintenance. When transactions are deleted from the database, nine cases arise, as shown in Fig. 1 [22].

In Fig. 1, **cases 2**, **3**, **4**, **7**, and **8** do not change the final frequent itemsets. **Case 1** may remove some existing frequent itemsets, and