Engineering Applications of Artificial Intelligence 68 (2018) 1–9

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Efficient algorithms for mining top-rank-*k* erasable patterns using pruning strategies and the subsume concept

Tuong Le [a], Bay Vo [b,c], Sung Wook Baik [a,*]

[a] Digital Contents Research Institute, Sejong University, Seoul, Republic of Korea
[b] Division of Data Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[c] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

ARTICLE INFO

Keywords:
Data mining
Erasable patterns
Pattern mining
Top-rank-*k*

ABSTRACT

Mining erasable patterns (EPs) is one of the emerging tasks in data mining which helps factory managers to establish plans for the development of new systems of products. However, systems usually face the problem of many EPs. Therefore, the problem of mining top-rank-*k* EPs, and an algorithm for mining these using the PID_List structure named VM, were proposed in 2013. In this paper, we propose two efficient methods, named the TEP (Top-rank-*k* Erasable Pattern mining) and TEPUS (Top-rank-*k* Erasable Pattern mining Using the Subsume concept) algorithms, for mining top-rank-*k* EPs. The TEP algorithm uses the dPidset structure to reduce the memory usage and a dynamic threshold pruning strategy to accelerate the mining process. The TEPUS algorithm is the extension of the TEP algorithm using the subsume concept and the index strategy to further speed up the mining time and reduce the memory usage. Finally, we conduct an experiment to compare the mining time, memory usage and scalability of TEP, TEPUS and two state-of-the-art algorithms (VM and dVM) for mining top-rank-*k* EPs. Our performance studies show that TEPUS outperforms TEP, VM and dVM.

1. Introduction

Data mining has attracted a lot of attention in recent years, due to the huge amounts of data and the need to turn this into useful information and knowledge. Since the problem of mining frequent patterns (FPs), which consists of extracting patterns frequently occurring in transaction datasets, was first introduced (Agrawal et al., 1993), it has played a key role in many important data mining tasks, such as association rule analysis (Lin et al., 2016; Sahoo et al., 2015), cluster analysis (Agarwal and Bharadwaj, 2015; Nanda and Panda, 2015), text mining (Indurkhya, 2015) and their applications (Nassirtoussi et al., 2014; Vairavasundaram et al., 2015). Frequent patterns are itemsets, subsequences, or substructures that appear in a dataset with a frequency no less than the user-specified minimum support threshold. Several algorithms have been proposed for frequent pattern mining (Deng, 2016; Deng and Lv, 2015; Lin et al., 2017). Besides the problem of mining FPs, those of mining top-*k* and top-rank-*k* FPs have also been proposed, and attracted many researchers using a variety of methods (Dam et al., 2016; Huynh et al., 2015). In addition, there are a lot of studies related to top-*k* pattern mining, such as those examining top-*k* high utility patterns (Dam et al., 2017; Duong et al., 2016; Ryang and Yun, 2015; Tseng et al., 2016) and top-*k* sequential patterns (Petitjean et al., 2016).

In 2009, an interesting variation of pattern mining was first presented, that of mining erasable patterns (EPs) (Deng et al., 2009). These patterns can help factory managers make new production plans, and there are now many methods to solve this problem such as MERIT (Fast Mining ERasable ITemsets) (Deng and Xu, 2012), MEI (Mining Erasable Itemsets) (Le and Vo, 2014) and EIFDD (Erasable Itemsets for very Dense Datasets) (Nguyen et al., 2015). MERIT first creates a WPPC-tree, then generates NC_Sets associated with erasable items from WPPC-tree. Using the NC_Set structure, MERIT reduces the memory usage for mining EPs. However, MERIT uses the union strategy, in which X's NC_Set is the subset of Y's NC_Set where $X \subset Y$. In addition, MERIT stores the value of a product's profit in each NC of NC_Set, which leads to data duplication. MEI was thus proposed which uses the dPidset structure to reduce the memory usage and mining time compared with MERIT. Next, EIFDD is an extension of the MEI algorithm which uses the subsume concept to reduce the mining time and memory usage for very dense datasets. dPidset is still used for mining EPs. In addition to the problem of mining EPs, several problems related to this, such as mining EPs with subset and

[*] Corresponding author.
E-mail addresses: tuonglc@sju.ac.kr (T. Le), vodinhbay@tdt.edu.vn (B. Vo), sbaik@sejong.ac.kr (S.W. Baik).

https://doi.org/10.1016/j.engappai.2017.09.010
Received 23 April 2017; Received in revised form 4 September 2017; Accepted 7 September 2017
0952-1976/© 2017 Elsevier Ltd. All rights reserved.

superset itemset constraints (Vo et al., 2017), and weighted erasable patterns (Lee et al., 2016, 2015; Yun and Lee, 2016), have also been developed.

The traditional approaches for mining Eps, such as MERIT, MEI and EIFDD, provide a very large number of patterns, which reduces the effectiveness of intelligent systems. These systems have to find all the patterns (mining phase) by using the traditional algorithms and then rank these to select a small number (ranking phase) by themselves. These two phases make the systems consume more time and resources, and they may even fail to run due to the huge memory consumption or lack of storage space. Therefore, the problem of mining top-rank-*k* EPs was presented in Deng (2013) to combine the mining and ranking phases into one. VM is the first algorithm to deal with this problem, and applies the PID_List structure and union PID_List strategy for mining top-rank-*k* EPs. This strategy makes this algorithm need more time to compute the union of PID_Lists, and requires a lot of memory usage to store the PID_Lists. To overcome this issue, Nguyen et al. (2014) proposed an enhanced VM algorithm, called dVM, for mining top-rank-*k* EPs using a new structure, the dPID_List. This structure uses the diff strategy to reduce the memory usage and computational operations. Although dVM outperforms VM in terms of mining time and memory usage, the resources required by dVM algorithm are still enormous. The dPidset structure (Le and Vo, 2014) was then proposed in 2014, and this is a very effective structure for mining EPs. Therefore, in this paper, we propose two algorithms for mining top-rank-*k* EPs using the dPidset structure, two pruning strategies and the subsume concept. The main contributions of this paper are as follows: (i) the dynamic threshold pruning strategy for mining top-rank-*k* EPs is first proposed. (ii) We then propose the TEP algorithm using the dynamic threshold pruning strategy for mining top-rank-*k* EPs. (iii) Finally, we proposed the TEPUS algorithm, which is an extension of the TEP algorithm with the subsume concept and the index strategy. Experiments were conducted to demonstrate the effectiveness of the proposed algorithms. The results show that the proposed algorithms outperform the VM and dVM algorithms in terms of mining time, memory usage and scalability for mining top-rank-*k* EPs.

The rest of the paper is organized as follows: Section 2 presents the related works, including the problem of mining top-rank-*k* EPs, dPidset structure and the subsume concept in EP mining. The TEP algorithm is then proposed in Section 3. Next, we apply the subsume concept and the index strategy to the TEPUS algorithm in Section 4. The experimental results are presented in Section 5 to compare the runtime, memory usage and scalability among the TEP, TEPUS, VM and dVM algorithms for mining top-rank-*k* EPs. Finally, Section 6 summarizes the results and offers some future research topics.

2. Related works

2.1. Mining top-rank-k EPs

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of all items, which are the abstract representations of components of products. A product dataset is denoted by $DB = P_1, P_2, \ldots, P_n\}$, where $P_i (1 \le i \le n)$ is a product presented in the form of $\langle Items, Val\rangle$, where *Items* are the items (or components) that constitute P_i and *Val* is the profit that the factory obtains by selling the product P_i. A set $X \subseteq I$ is also called a pattern, and a pattern with *k* items is called a *k*-pattern. The example product dataset in Table 1 will be used throughout this article.

Definition 1 (*Gain of a Pattern*). Let $X (\subseteq I)$ be a pattern. The gain of X is defined as:

$$g(X) = \sum_{\{P_k | X \cap P_k.Items \neq \emptyset\}} P_k.Val. \qquad (1)$$

Table 1
An example product dataset.

Product	Items	Val
P_1	*a, b*	1000
P_2	*a, b, e*	200
P_3	*c, e*	150
P_4	*b, d, e, f*	50
P_5	*c, d, e*	100
P_6	*d, e, f, h*	200
P_7	*d, h*	150
P_8	*d, f, h*	100

Table 2
Top six ranked EPs and their gains for the example dataset.

Rank	Gain	Patterns
1	250	*c*
2	350	*f*
3	450	*h*
4	500	*hf*
5	600	*d, fc, dh, df, dhf*
6	700	*e, ec, hc*

Definition 2 (*Rank of a Pattern*). Given a product dataset *DB*, the rank of a pattern X is as follows:

$$R_X = |\{g(Y) | Y \subseteq I \text{ and } g(Y) \le g(X)\}|. \qquad (2)$$

Definition 3 (*Top-rank-k EPs*). A pattern X ($\subseteq I$) is called a top-rank-*k* EP if and only if $R_X \le k$.

Given a transaction dataset *DB* and a threshold *k*, the problem of mining top-rank-*k* EPs is the task of finding the complete set of EPs whose ranks are no greater than *k*.

Example 1. The example dataset in Table 1 will be used throughout the article. According to Definition 1, there is a gain of $\{c\} = 250$, because two transactions, namely 3 and 5, contain *c*. Table 2 shows the top six ranked EPs and their gains in the example dataset.

2.2. dPidset structure

Deng (2013) proposed the VM algorithm using the PID_List (product identifiers) concept for mining top-rank-*k* EPs. A PID_List is the set of pairs $\langle PID, Val\rangle$, where *PID* is the product identifier and *Val* is the gain of this product (the gain obtained by selling this product). VM uses the union strategy in which $PID\_List(XAB)$ is determined by $PID\_List(XA) \cup PID\_List(XB)$. This strategy requires a lot of memory and operations. Therefore, Nguyen et al. (2014) proposed $dPID\_List$ for mining the top-rank-*k* EPs. In this, $PID\_List(XAB)$ is determined by $dPID\_List(XA) \setminus dPID\_List(XB)$ to reduce both memory usage and mining time. However, PID_List and dPID_List store each product's profit (*Val*) in a pair $\langle PID, Val\rangle$. This leads to data duplication because a pair $\langle PID, Val\rangle$ can appear in many PID_Lists and dPID_Lists. From that reason, Le and Vo (2014) proposed the dPidset structure to reduce memory usage by applying an index of gain for efficiently mining EPs. In this section, we summarize this as follows.

Definition 4 (*The Pidset of a Pattern*). The pidset of pattern X is denoted as:

$$p(X) = \bigcup_{A \in X} p(A) \qquad (3)$$

where: A is an item in pattern X; and $p(A)$ is the pidset of item A, i.e., the set of product identifiers (IDs) which have item A.

Definition 5 (*The Gain of a Pattern Based on the Pidset*). The gain of pattern X, denoted by $g(X)$, is computed easily as follows:

$$g(X) = \sum_{P_i \in p(X)} Val(P_i). \qquad (4)$$