Expert Systems with Applications 42 (2015) 156–164

Contents lists available at ScienceDirect

Expert Systems with Applications

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

# An efficient and effective algorithm for mining top-rank-*k* frequent patterns

Quyen Huynh-Thi-Le [a], Tuong Le [b,c], Bay Vo [b,c,*], Bac Le [a]

[a] Department of Computer Science, University of Science, VNU-HCM, Viet Nam
[b] Division of Data Science, Ton Duc Thang University, Ho Chi Minh, Viet Nam
[c] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh, Viet Nam

ARTICLE INFO

*Article history:*
Available online 7 August 2014

*Keywords:*
Data mining
Pattern mining
Top-rank-*k* frequent patterns
N-list

ABSTRACT

Frequent pattern mining generates a lot of candidates, which requires a lot of memory usage and mining time. In real applications, a small number of frequent patterns are used. Therefore, the mining of top-rank-*k* frequent patterns, which limits the number of mined frequent patterns by ranking them in frequency, has received increasing interest. This paper proposes the *i*NTK algorithm, which is an improved version of the NTK algorithm, for mining top-rank-*k* frequent patterns. This algorithm employs an N-list structure to represent patterns. The subsume concept is used to speed up the process of mining top-rank-*k* patterns. The experiments are conducted to evaluate *i*NTK and NTK in terms of mining time and memory usage for eight datasets. The experimental results show that *i*NTK is more efficient and faster than NTK.



## 1. Introduction

An expert system is an intelligent system that solves the complex problems based on knowledge throughout inference procedures. Generally, there are three components in an expert system including knowledge base, inference engine and user interface (Jackson, 1999). The central of expert systems is the knowledge base, because it contains the problem solving knowledge of the particular application (Ahmed, 2008). Therefore, the reduction of this knowledge space plays a big role in the implemented performance of expert systems. Association rules are important of the knowledge (Daniel & Viorel, 2004; Guil, Bosch, Túnez, & Marín, 2003) which represent the relationships between items in a dataset. To generate association rules, traditional approaches first mine frequent patterns which are itemsets, subsequences, and substructures that appear in large transactions or relational datasets with a frequency no less than a given threshold. After that, the system uses these frequent patterns and the minimum confidence to find all rules. Two above phrases require a lot of memory usage and mining time. Therefore, the reduction of time to mine frequent patterns is very useful to enhance expert systems.

Currently, there are many forms of patterns such as frequent, subsequences, and substructure patterns. Mining frequent patterns is an indispensable component in many data mining tasks such as association rule mining (Agrawal, Imielinski, & Swami, 1993; Vo, Hong, & Le, 2012, 2013; Vo, Coenen, Le, & Hong, 2013; Vo, Le, Coenen, & Hong, 2014; Vo, Le, Hong, & Le, 2014a,b), sequential pattern mining (Agrawal & Srikant, 1995; Pham, Luo, Hong, & Vo, 2014), and classification (Liu, Hsu, & Ma, 1998; Nguyen, Vo, Hong, & Thanh, 2012; Nguyen, Vo, Hong, & Thanh, 2013). Since the introduction of frequent pattern mining (Agrawal et al., 1993), various algorithms (Agrawal & Srikant, 1994; Han, Dong, & Yin, 1999; Han, Pei, & Yin, 1999; Zaki, 2000; Zaki & Gouda, 2003) have been proposed for efficiently performing the task. These algorithms can be partitioned into two main categories: using the traditional horizontal dataset format such as two important algorithms, Apriori and FP-growth (Agrawal & Srikant, 1994; Han, Dong et al., 1999; Han, Pei et al., 1999) and using the vertical dataset format such as Eclat (Zaki, 2000).

In general, mining frequent patterns uses a minimum support threshold (*min_sup*) to generate correctly and completely frequent patterns. However, setting this threshold is an interesting problem. Whether this threshold is too large or too small, it also influences the number of generated frequent patterns in a dataset. In addition, the number of produced frequent patterns is very large, while applications such as expert systems, recommendation systems and so on, only use a small number of frequent patterns. From

* Corresponding author at: Division of Data Science, Ton Duc Thang University, Ho Chi Minh, Viet Nam.
*E-mail addresses:* lequyenk41@gmail.com (Q. Huynh-Thi-Le), lecungtuong@tdt.edu.vn (T. Le), vodinhbay@tdt.edu.vn (B. Vo), lhbac@fit.hcmus.edu.vn (B. Le).

http://dx.doi.org/10.1016/j.eswa.2014.07.045
0957-4174/

above problems, Han, Wang, Lu, and Tzvetkov (2002) proposed top-$k$ frequent closed pattern mining, where $k$ is the number of frequent closed patterns to be mined. Then, the authors proposed the TFP algorithm to solve this task. Unlike frequent patterns, frequent closed patterns have length no less than the minimal length of each pattern (*min_l*). Although TFP implements effectively its mission, but like *min_sup*, set the value *min_l* is not a simple problem for users. Therefore, a new direction of research was proposed, that is the problem of top-rank-$k$ frequent pattern mining. To solve this problem, FAE (Deng & Fang, 2007) and VTK algorithms (Fang & Deng, 2008) are proposed. A top-rank-$k$ of frequent patterns is selected based on rank order of frequency. Recently, Deng (2014) proposed NTK algorithm for mining top-rank-$k$ frequent patterns based on the idea of PPC-tree (the Pre-order and Post-order Code tree). NTK is efficient due to its patterns presentation based on Node-list structure. The experimental results show that NTK is more effective than FAE and VTK.

Considering carefully Node-list structure, we found that N-list (Deng, Wang, & Jiang, 2012) better than Node-list because the length of the Node-list of a pattern is greater than the length of its N-list. Hence, the time required to join two Node-lists is longer than that of N-lists. In addition, NTK must generate and test all candidates in each loop of the algorithm. Therefore, this paper presents an efficient method for mining top-rank-$k$ frequent patterns called *i*NTK. Unlike NTK, *i*NTK uses N-list structure with an improved N-list intersection function to reduce the run-time and memory-consuming. Moreover, *i*NTK employs the subsume index concept to directly mine frequent patterns without generating candidates in a number of cases.

The rest of this paper is organized as follows. Section 2 presents the related work for mining top-rank-$k$ frequent patterns. Section 3 introduces the basic concepts. The *i*NTK algorithm for mining top-rank-$k$ frequent patterns is described in Section 4. Section 5 compares the performance of the *i*NTK and NTK algorithms. Section 6 summarizes the study and gives some topics for future research.

## 2. Related work

Since mining top-rank-$k$ frequent patterns is proposed, a number of algorithms such as FAE, VTK and NTK were built to solve this problem. Besides, mining top-rank-$k$ erasable itemsets is also proposed (Deng, 2013; Nguyen, Le, Vo, & Le, 2014).

FAE is the first algorithm (Deng & Fang, 2007) to solve the problem of mining top-rank-$k$ frequent patterns. FAE is an acronym for "Filtering and Extending"; it uses heuristic rules to reduce the search space, filters undesired patterns and selects useful patterns to generate the next patterns. Next, VTK (Fang & Deng, 2008) (Vertical Mining of top-rank-$k$ frequent patterns) is more efficient than FAE because it does not need to scan the entire dataset to calculate the support of frequent patterns.

Recently, NTK algorithm was built for mining top-rank-$k$ frequent patterns (Deng, 2014). This algorithm was proven to be more effective than FAE and VTK because it uses Node-list, a data structure that has been effectively used in frequent pattern mining (Deng & Wang, 2010). In NTK, first a tree construction algorithm is used to build a PPC-tree. Then, Node-list structure associated with frequent 1-patterns is generated. Unlike FP-tree-based approaches, this approach does not build additional trees repeatedly; it mines frequent patterns directly using Node-list.

In 2010, Node-list is first proposed (Deng & Wang, 2010). After that, N-list, like Node-list structure, has also been proposed (Deng et al., 2012) to mine frequent patterns. Both of them are generated from a PPC-tree and a list of nodes sorted in pre-order ascending order. Besides, the Node-list and N-list of a pattern contains $t$ items can be produced from two patterns contains $(t-1)$ items. The difference between them is that Node-list is constructed by the suffix nodes while N-list is constructed by prefix nodes, and the length of Node-list of a pattern is greater than the length of N-list of a pattern. Therefore, Node-list used in NTK requires a lot of time and memory. In Vo, Coenen et al., 2013; Vo, Hong et al., 2013; Vo, Le, Coenen et al., 2014; Vo, Le, Hong et al., 2014a,b, N-list and subsume index (Song, Yang, & Xu, 2008) of frequent 1-pattern was used for mining frequent itemsets effectively. NSFI algorithm was proven more outperforms than the PrePost. In this paper, *i*NTK, an improvement algorithm of NTK, is proposed. This algorithm uses N-list structure and subsume index of 1-patterns to enhance the mining time and the memory usage.

## 3. Problem definition

### 3.1. Frequent patterns

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items, and $DB = \{T_1, T_2, \ldots, T_n\}$ be a set of transactions, where $T_i$ $(1 \leqslant i \leqslant n)$ is a transaction that has a unique identifier and contains a set of items. Given a pattern $P$ and a transaction $T$, it is said that $T$ contains $P$ if and only if $P \subseteq T$.

**Definition 1** (*support of a pattern*). Given a $DB$ and a pattern $P$ $(\subseteq I)$, the support of pattern $P$ ($SUP_P$) in $DB$ is the number of transactions containing $P$.

A pattern $P$ is a frequent pattern if support of $P$ is no less than a given *min_sup*.

### 3.2. Problem of mining top-rank-k frequent patterns

Deng and Fang (2007) described the problem of mining top-rank-$k$ patterns as follows.

**Definition 2** (*rank of a pattern*). Given a $DB$ and a pattern $X$ $(\subseteq I)$, the rank of $X$ ($R_X$) is defined as $R_X = |\{SUP_Y | Y \subseteq I \text{ and } SUP_Y \geqslant SUP_X\}|$, where $|Y|$ is the number of items in $Y$.

**Definition 3** (*top-rank-k frequent patterns*). Given a $DB$ and a threshold $k$, a pattern $P$ $(\subseteq I)$ belongs to a top-rank-$k$ frequent pattern ($TR_k$) if and only if $R_P \leqslant k$.

Given a $DB$ and a threshold $k$, top-rank-$k$ frequent pattern mining is the task of finding the set of frequent patterns whose ranks are no greater than $k$. That means that $TR_k = \{P | P \subseteq I \text{ and } R_P \leqslant k\}$.

**Example 1.** Dataset $DB_E$ in Table 1 is used throughout the article. According to Definition 1, $SUP_{\{c\}} = 5$ because five transactions, namely 2, 3, 4, 5, and 6, contain $c$. Table 2 shows the ranks and supports of all patterns in $DB_E$. According to Table 2, $SUP_{\{c\}}$ is the largest, and therefore $R_{\{c\}} = 1$.

### 3.3. N-list structure

Deng et al. (2012) presented the PPC-tree, an FP-tree-like structure (Han, Dong et al., 1999; Han, Pei et al., 1999), the PPC-tree construction algorithm, and the N-list structure as follows.

**Table 1**
Example dataset ($DB_E$).

| TID | Items |
|---|---|
| 1 | *a*, *b* |
| 2 | *a*, *b*, *c*, *d* |
| 3 | *a*, *c*, *e* |
| 4 | *a*, *b*, *c*, *e* |
| 5 | *c*, *d*, *e*, *f* |
| 6 | *c*, *d* |