Engineering Applications of Artificial Intelligence 32 (2014) 88–99

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

An efficient method for mining non-redundant sequential rules using attributed prefix-trees

Thi-Thiet Pham [a], Jiawei Luo [b], Tzung-Pei Hong [c,d], Bay Vo [e,*]

[a] Faculty of Information Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Viet Nam
[b] School of Information Science and Engineering, Hunan University, Changsha City, Hunan Province 410082, Republic of 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] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Viet Nam

ARTICLE INFO

Article history:
Received 27 February 2013
Received in revised form
7 January 2014
Accepted 26 February 2014
Available online 29 March 2014

Keywords:
Sequential pattern
Sequential generator pattern
Closed sequential pattern
Non-redundant sequential rule
Prefix-tree

ABSTRACT

Mining sequential generator patterns and mining closed sequential patterns are important to sequence mining. They address difficulties in the mining of sequential patterns by reducing the number of sequential rules, and the results are often used to generate non-redundant sequential rules. This paper proposes an efficient method for mining non-redundant sequential rules from an attributed prefix-tree. The proposed method has two phases. In the first phase, it builds a prefix-tree that stores all the sequential patterns from a given sequence database. Then in the second phase, it mines non-redundant sequential rules from this prefix-tree. In the prefix-tree building process, each node on the prefix-tree has a field that indicates whether this node is a minimal sequential generator pattern, and another field that indicates whether this node is a closed sequential pattern. By traversing the prefix-tree, non-redundant sequential rules can be easily mined from a minimal sequential generator pattern X to a closed sequential pattern Y such that X is a prefix of Y. In addition, a good pruning mechanism is proposed to reduce the search space and the execution time in the mining process. Experimental results show that the proposed method is more efficient than existing methods in mining non-redundant sequential rules.

1. Introduction

Mining sequential patterns from a sequence database (Agrawal and Srikant, 1995; Ayres et al., 2002; Gouda et al., 2010; Pei et al., 2004; Srikant and Agrawal, 1996; Zaki, 2000) may generate many sequential patterns especially when the support thresholds are low. These patterns may be irrelevant and a sequence of events that appear frequently in a database is thus insufficient for predicting events. Therefore, the sequential rule mining problem was proposed (Fournier-viger et al., 2011, 2012a, 2012b; Harms and Deogun, 2004; Lo et al., 2009; Spiliopoulou, 1999; Van et al., 2011; Zang et al., 2010) and sequential rules are used to allow better prediction. Sequential rules express the relationships between sequential patterns from a sequence database (Spiliopoulou, 1999; Lo et al., 2009; Van et al., 2011) and can be considered as a natural extension of original sequential patterns, just as association rules are a natural extension of frequent itemsets (Srikant and Agrawal, 1996). They have been applied to many application areas, including the stock market (Berry and Linoff, 1997; Brin et al., 1997; Hsieh et al., 2006; Olaniyi et al., 2011; Wu et al., 2012), DNA sequence analysis (Chang et al., 2012); web usage behavior analysis (Jeong et al., 2011; Wang and Lee, 2011), trading (Dong and Pei, 2007), e-learning (Faghihi et al., 2010), weather observation (Hamilton and Karimi, 2005), and software engineering (Lo and Khoo, 2006). Using sequential rules, the series of events that usually occurs after a series of previous ones can be predicted. Sequential rules are rather simple, but their information has many important implications, which can be used for decision-making, management and behavior analysis. Compared with sequential patterns, sequential rules can help users better understand the chronological order of the sequences present in a sequence database.

A sequential rule has the form $X \Rightarrow Y$ (Sup, $Conf$), where X and Y are sequential patterns, $X \cap Y = \emptyset$. A sequential rule can be created by splitting a sequential pattern into two parts: the prefix (pre) and the postfix ($post$). If pre is concatenated with $post$, denoted $pre++post$, then the result is the original sequential pattern.

* Corresponding author.
E-mail addresses: phamthithiet@hui.edu.vn (T.-T. Pham), luojiawei@hnu.edu.cn (J. Luo), tphong@nuk.edu.tw (T.-P. Hong), vdbay@it.tdt.edu.vn (B. Vo).

http://dx.doi.org/10.1016/j.engappai.2014.02.019
0952-1976/

However, generating a full set of sequential rules is very costly, even for a sparse dataset. In addition, a lot of low-quality rules that are almost meaningless are generated. To address these problems, mining non-redundant sequential rules has been proposed. A rule SR_1 is called a redundant rule if it can be inferred from another rule SR_2, where SR_1 and SR_2 have the same support and confidence. When a set of mined rules is considered as a composite filter, replacing a full set of rules with its corresponding non-redundant set of rules does not affect the accuracy of the filter. Several methods for mining non-redundant sequential rules have been proposed (Lo et al., 2009; Zang et al., 2010). These methods remove a significant number of redundant sequential rules but require a lot of time to check sequential generator patterns and closed sequential patterns to generate rules.

In this study, we would like to scan through all the sequential patterns to enumerate all sequential rules/non-redundant sequential rules, which satisfy the minimum confident threshold, from a sequence database. Meanwhile, tree structure is a good method commonly used to describe candidates (sequential patterns) in which each sequential pattern will be stored in one node in the tree. Once the tree structure storing the sequential pattern is built, we can apply the procedure traversed on the tree (the width search tree or depth search tree) to enumerate the candidate list (sequential rule/non-redundant sequential rule), but for mining sequential rules/non-redundant sequential rules from a sequence database, building a prefix-tree is very time consuming (may take more time than generating sequential rules/non-redundant sequential rules). For the above reason, this paper has proposed a method for mining non-redundant sequential rules by using the attributed Prefix-trees, and applying the child–parent relationship of the prefix-tree structure for mining non-redundant sequential rules, which greatly reduces the mining time required.

The paper would like to design the algorithm in two stages to reuse the tree-building results in the first stage for mining non-redundant sequential rules with different *minConf* values as *minConf*=0%, 50%, 70%, and so on. Hence we only build the prefix-tree one time. In the second stage, by passing on the prefix-tree, non-redundant sequential rules can be easy mined with different *minConf* thresholds. Two stages are proposed as follows: (1) building a prefix-tree that stores all sequential patterns in a sequence database and (2) generating non-redundant sequential rules from this prefix-tree. In the prefix-tree building process, each node on the prefix-tree has a field that indicates whether this node is a minimal sequential generator pattern (*IsmSGP*), and another field that indicates whether this node is a closed sequential pattern (*IsCSP*). By traversing the prefix-tree, non-redundant sequential rules can be easily mined from a sequence *X* at the node whose *IsmSGP* value is *true* (*T*) to a sequence *Y* at the node whose *IsCSP* value is *true* (*T*) such that *X* is a prefix of *Y*. In addition, to prune the search space in the mining process, the proposed algorithm does not generate rules from the root node of the subtree to its itemset-extended nodes set, but derives rules for sequences on the subtree whose nodes are sequence-extended nodes of the root node of the subtree. Experimental results show that the number of non-redundant sequential rules is less than that of the full sequential rules. Besides, the execution time for mining non-redundant sequential rules using the prefix-tree structure is much less than that using closed sequential patterns and generator sets (Zang et al., 2010).

In this study, the database is assumed to be binary, meaning that an item may/may not appear in the chain of transactions, not concerned about the weight of the item, such as price and number of items which purchased in each transaction of the attached corresponding sequences. The sequential currently mining algorithms popular currently are working on this kind of database. In fact, there are plenty of algorithms proposed for mining association rules from quantitative databases but they posed to resolve for the other problems such as: mining high utility itemsets (Le et al., 2011; Shie et al., 2013; Wu et al., 2013), mining frequent weighted itemsets (Vo et al., 2013), etc.

The rest of this paper is organized as follows. Section 2 reviews works related to mining sequential rules and non-redundant sequential rules. Section 3 presents problem definitions related to this paper. The proposed method for mining non-redundant sequential rules is described in Section 4. Section 5 describes the experimental results. The conclusion and future work are given in Section 6.

2. Related work

Spiliopoulou (1999) proposed a method for generating a full set of sequential rules from a set of discovered frequent sequences in a sequence database. Harms and Deogun (2004) then proposed the Minimal Occurrences With Constraints and Time Lags (*MOWCATL*) method for mining frequent sequential association rules from multiple sequential databases. The method combines the concept of association rules with frequent episodes, time lags between the occurrences of an antecedent sequence of a mined rule, and event constraints to find relationships between sequences in multiple datasets. However, this method is only used for mining episode rules that can be considered as a variant of sequential rules with constraints. Fournier-viger et al. (2011, 2012a, 2012b) then proposed the *RuleGrowth*, *CMRULES* and *TRuleGrowth* algorithms for generating sequential rules common to sequences in sequence databases.

Based on the method of Spiliopoulou (1999), Lo et al. (2009) proposed a generalized algorithm, called the *Full* algorithm, to mine a full set of sequential rules, Van et al. (2011) improved the *Full* algorithm (Lo et al., 2009) and proposed the *MSR-ImpFull* algorithm for finding sequential rules between pairs of sequential patterns. The algorithm was improved by sorting all sequential patterns in an ascending order of sizes. Sequences that are prefixes of a sequence *X* only appear before *X* in the list of sequential patterns. Based on the property of prefix-tree structure, the authors also presented another algorithm, called *MSR-PreTree* (Van et al., 2011), which could directly generate all sequential rules by traversing the prefix-tree.

Lo et al. (2009) introduced several rule sets generated by using the composition of various types of pattern sets including generators, projected-database generators, closed sequences, and projected-database closed sequences. They also proposed a compressed set of non-redundant rules that were generated from two pattern sets: the projected database closed pattern set (*LS-Closed*) and the closed pattern set (*CS-Closed*) (Lo et al., 2009). The antecedent of a rule is a sequence in an *LS-Closed* set, and the consequence is a sequence in a *CS-Closed* set. The authors proved that the compressed set of non-redundant rules was complete and tight, and proposed an algorithm for mining this set.

From the basis of description for redundant rules (Spiliopoulou, 1999) and a theory of non-redundant rules (Lo et al., 2009), Zang et al. (2010) then proposed a method for mining non-redundant sequential association rules from frequent closed sequences and sequence generator sets. They also proposed an algorithm called *CSGM* (closed sequence and sequence generator mining), which were extended from *PrefixSpan* (Pei et al., 2004) and *CloSpan* (Yan et al., 2003), for mining frequent closed sequences and sequence generators at the same time. Zang et al. (2010) then were based on the achieved results, which are the set of closed sequences and the set of sequence generators generated from the *CSGM* method, to propose a method for mining non-redundant sequential association rules. This method (Zang et al., 2010) removes a significant