Decision Support Systems 46 (2009) 501–511

Contents lists available at ScienceDirect

Decision Support Systems

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

A change detection method for sequential patterns

Chieh-Yuan Tsai *, Yu-Chen Shieh

Department of Industrial Engineering and Management, Yuan Ze University, Taiwan

ARTICLE INFO

Article history:
Received 29 August 2007
Received in revised form 4 September 2008
Accepted 15 September 2008
Available online 25 September 2008

Keywords:
Sequential patterns
Change mining
Pattern matching
Customer behaviors

ABSTRACT

Recent trends in customer-oriented markets drive many researchers to develop sequential pattern mining algorithms to explore consumer behaviors. However, most of these studies concentrated on how to improve accuracy and efficiency of their methods, and seldom discussed how to detect sequential pattern changes between two time-periods. To help business managers understand the changing behaviors of their customers, a three-phase sequential pattern change detection framework is proposed in this paper. In phase I, two sequential pattern sets are generated respectively from two time-period databases. In phase II, the dissimilarities between all pairs of sequential patterns are evaluated using the proposed sequential pattern matching algorithm. Based on a set of judgment criteria, a sequential pattern is clarified as one of the following three change types: an emerging sequential pattern, an unexpected sequence change, or an added/perished sequential pattern. In phase III, significant change patterns are returned to managers if the degree of change for a pattern is large enough. A practical transaction database is demonstrated to show how the proposed framework helps managers to analyze their customers and make better marketing strategies.

1. Introduction

Sequential pattern mining is the technique that explores frequently occurring patterns related to time from a large-scale database. It has been applied to web log mining [5], customer purchase behavior analysis [10], DNA sequence analysis [11], project team coordination [22], retailing management [8], and so on. Previous studies related to sequential pattern mining mainly focused on how to build accurate models or how to discover interesting rules in efficient ways. AprioriAll [1], GSP [29], PrefixSpan [27], FFS [32], and SPAM [3] are some well known algorithms that efficiently identify sequential patterns. In addition, other efforts have concentrated on sequential pattern mining in multi-databases [18] with constraints [7], time-intervals [9], and fuzzy sets [6]. Relatively few attempts have been made to analyze sequential pattern changes in databases collected over time [19]. However, some popular patterns at one time-period may not be valid in another time-period [4]. For example, the sequential pattern "Computer→Memory→Color_Printer" is frequent in the last year. However, this pattern might not be popular in this year but change to "Computer→Memory→Multifunctional_Printer." If managers cannot capture this dynamic behavior change in time and provide appropriate products or services to their customer, customer attrition will be unavoidable.

There have been many existing works focusing on dynamic aspects or comparison between two different datasets. They can be broadly classified as four research trends [4,23,28]. The first trend is to improve rule matching accuracy when new transaction data are constantly added to the original transaction database. El-Sayed et al. [14] introduced an efficient strategy for discovering frequent patterns in sequence databases that requires only two scans of the database. Koh and Shieh [17] proposed an algorithm of AFPIM (Adjusting FP-tree for Incremental Mining) based on adjusting FP-tree structures to maintain frequent itemsets discovered in a changing database. The algorithm only scans changed data in the database, and adjusts FP-tree structures to enhance the efficiency of mining association rules. Masseglia et al. [24] proposed an efficient algorithm, called incremental sequence extraction (ISE), to compute frequent sequences in an updated database. The algorithm minimizes computational costs by re-using the minimal information from the original database. Lee et al. [21] proposed a SWF (sliding-window filtering) algorithm, which uses a filtering thresho ld to deal with the candidate itemset generation. The algorithm has not only significantly reduced I/O and CPU costs but also effectively controlled memory utilization. The second research trend is to discover emerging patterns. Emerging pattern mining gauges significant changes or differences from one database to another [12]. Fan and Ramamohanarao [15] developed an algorithm which applied the minimum support, the minimum growth rate and chi-squared values to extract interesting emerging patterns. Terlecki and Walczak [30] presented the relations between rough set and jumping emerging patterns. They proposed practical applications of these observations to the minimal reduced problem and to test whether a given attribute set is differentiating.

* Corresponding author. Department of Industrial Engineering and Management, Yuan Ze University, No. 135, Yuantung Rd., Chungli City, Taoyuan County, Taiwan. Tel.: +886 3 463 8800x2512; fax: +886 3 463 8907.
E-mail addresses: cytsai@saturn.yzu.edu.tw (C.-Y. Tsai), f8902045@hotmail.com (Y.-C. Shieh).

0167-9236/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2008.09.003

The third research trend is subjective interestingness mining. Interestingness mining is used to find unexpected rules with respect to the user's existing knowledge. Padmanabhan and Tuzhilin [25] developed methods to generate unexpected patterns with respect to managerial intuition by eliciting managers' beliefs about the domain and used these beliefs to seed the search for unexpected patterns in data. The methods provide managers with more relevant patterns from data and aid in effective decision making. Liu et al. [23] proposed a new approach to assist users in finding interesting rules from discovered association rules and analyze subjective interestingness. The approach first asks users to specify his/her existing knowledge such as beliefs or concepts, and then analyzes the discovered association rules to conformity and various interestingness criteria. Padmanabhan and Tuzhilin [26] presented a method for discovering unexpected patterns in data. The method combines independent concepts of minimality and unexpectedness to discover a minimal set of unexpected patterns. Lanquillon [20] proposed the concepts of added rules and perished rules. An added rule is the rule that is hard to be found in the past database, while a perished rule is the rule that is difficult to be found in the present database.

The fourth research trend is to discover regularity from time series data. Han et al. [16] presented several algorithms for efficient mining of partial periodic patterns by exploring some interesting properties related to partial periodicity. The algorithms shows that mining partial periodicity needs only two scans over a time series database to efficiently mine long periodic patterns. Yu et al. [31] proposed a granulation-based method to find similarities between two time series. The granulation-based method deals with problematic temporal data mining tasks such as similar subsequence searching, clustering and indexing etc. Altiparmark et al. [2] proposed a novel approach for information mining between time series data. They utilized frequent itemset mining as well as clustering and declustering techniques with novel distance metrics for measuring similarities between items series data.

Although these researches can efficiently detect the dynamic changes between two datasets, their discussions are limited to association rules only. None of them discusses the change of sequential patterns in two time-periods. To bridge the gap, this study proposes a change detection framework to observe the dynamic alternation of sequential patterns between two time-periods. The rest of this paper is organized as follows. Section 2 defines the dissimilarity measurement for sequential patterns. Section 3 details each phase of the proposed change detection framework. With the framework, a sequential pattern can be classified as one of the three change types: an emerging sequential pattern, an unexpected sequence change, or an added/perished sequential pattern. Section 4 provides an implementation case to demonstrate the benefit of the proposed framework. A summary and future perspective for the study is concluded in Section 5.

2. The dissimilarity measurement for sequential patterns

In this section, a formal representation for sequential patterns is illustrated first. Then, the dissimilarity measurement for two sequential patterns in different time-periods is defined. The measurement is then derived using the proposed sequential pattern matching algorithm. Finally, the minimal difference value between one and a set of sequential patterns are formulated.

2.1. Sequential patterns

Let $I=\{i_1, i_2, \ldots, i_n\}$ be a set of items. An itemset is a non-empty set of items. A sequence is an ordered list of itemsets, represented as $<s_1 s_2 \ldots s_l>$ where s_j is an itemset, i.e., $s_j \subseteq I$ for $1 \le j \le l$. s_j is also called an element of the sequence and denoted as $(x_1 x_2 \ldots x_m)$ where $x_k \in I$ for $1 \le k \le m$. The number of instances of items in a sequence is called the length of the sequence. A sequence with length l is called a l-sequence. A sequence $a=<a_1 a_2 \ldots a_n>$ is called a subsequence of $b=<b_1 b_2 \ldots b_m>$, and b is called a super sequence of a, denoted as $a \subseteq b$, if there exist integers $1 \le j_1 < j_2 < \ldots < j_n \le m$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, …, $a_n \subseteq b_{j_n}$. A sequence database D is a set of tuples <sid, s>, where sid is a sequence-id and s is a sequence. A tuple <sid, s> is said to contain a sequence a, if a is a subsequence of s, i.e., $a \subseteq s$. The number of tuples in a sequence database D containing sequence a is called the support of a, denoted as Sup(a)1. Given a sequence database D and user specified minimum support $\gamma_{\min}$, a sequence a is a sequential pattern in D if $\mathrm{Sup}(a) \ge \gamma_{\min}$. The sequential pattern mining problem is to find the complete set of sequential patterns with respect to D and $\gamma_{\min}$.

2.2. Dissimilarity measurement

Let D^t and D^{t+k} be the databases (or datasets) at time-period t and $t+k$ respectively, where D^t and D^{t+k} are disjoint. S^t and S^{t+k} represent the discovered sequential pattern sets from D^t and D^{t+k} respectively. s_i^t and s_j^{t+k} indicate individual sequential patterns in the corresponding sequential pattern sets S^t and S^{t+k} respectively, where $i=1,2,\ldots,|S^t|$ and $j=1,2,\ldots,|S^{t+k}|$. $\mathrm{Sup}(s_i^t)$ represents the support of s_i^t in D^t and $\mathrm{Sup}(s_j^{t+k})$ represents the support of s_j^{t+k} in D^{t+k}.

The dissimilarity between s_i^t in S^t and s_j^{t+k} in S^{t+k} is defined based on the following three considerations. First, the higher the number of mismatched elements between s_i^t and s_j^{t+k} is, the higher the dissimilarity is. For instance, we have s_i^t: $A \rightarrow B \rightarrow C \rightarrow D$ and s_{i+1}^t: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$ in S^t, and s_j^{t+k}: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$ in S^{t+k}. When s_j^{t+k} is compared with s_i^t, there are two among the six elements mismatched (2/6). However, when s_j^{t+k} is compared with s_{i+1}^t, only one among six elements are mismatched (1/6). It is reasonable to claim that the dissimilarity between s_i^t and s_j^{t+k} is greater than the dissimilarity between s_{i+1}^t and s_j^{t+k}.

Second, the position of the element changed in a sequence is taken into consideration based on the following observation. Assume that we have a sequence s and a sequential pattern set D. To predict what element(s) will appear after s, we will match the elements in s with the prefix elements of all patterns in D. For example, D= {"A→B→C→D", "A→B→C→E", "A→B→F→G", "A→B→H→J"}. If s="A→B→C", the patterns "A→B→C→D" and "A→B→C→E" are matched. Thus, two elements (subsequences) "D" or "E" could appear next if s is inputted. On the other hand, if s="A→B", the patterns "A→B→C→D", "A→B→C→E", "A→B→F→G", "A→B→H→J" are matched. Thus, four subsequences "C→D", "C→E", "F→G", or "H→J" could appear next. It is clear that if number of elements in s is fewer, the number of patterns matched is higher. This also implies that the prediction accuracy is low since there are many possible outcomes when s is known. Based on this observation, the change between two patterns appears in the rare position is prefer, because the number of matched elements is more. Therefore, if the element change position from s_i^t and s_j^{t+k} is later than the one from s_i^t and s_{j+1}^{t+k}, the dissimilarity between s_i^t and s_j^{t+k} should be smaller than the dissimilarity between s_i^t and s_{j+1}^{t+k}. For example, we have s_i^t: $A \rightarrow B \rightarrow C \rightarrow D$ in S^t, s_j^{t+k}: $A \rightarrow B \rightarrow C \rightarrow E$ and s_{j+1}^{t+k}: $A \rightarrow B \rightarrow E \rightarrow D$ in S^{t+k} The alternation from s_i^t to s_j^{t+k} is in position four (three common prefix elements between s_i^t and s_j^{t+k}, while the alternation from s_i^t and s_{j+1}^{t+k} is in position three (two common prefix elements between s_i^t and s_{j+1}^{t+k}). Thus, we claim that the dissimilarity between s_i^t and s_j^{t+k} is smaller than the dissimilarity between s_i^t and s_{j+1}^{t+k}.

Third, the dissimilarity between any mismatched elements should be evaluated. For example, we have s_i^t: $A \rightarrow B \rightarrow C \rightarrow D$ in S^t and s_j^{t+k}: $A \rightarrow B \rightarrow C \rightarrow E$ and s_{j+1}^{t+k}: $A \rightarrow B \rightarrow C \rightarrow F$ in S^{t+k}. In addition, element D contains items (p, q, x, y, z), element E contains items (m, p, q, x, y), and element F contains items (m, n, o, q, r). It is obvious that both the change from s_i^t to s_j^{t+k} and the change from s_i^t to s_j^{t+k} are one element mismatched. However, the dissimilarity between D in s_i^t and F in s_j^{t+k}