Sequential Pattern Mining with Various Constraints: An Enhanced Approach

Mansi Vithalani
Computer Science & Engineering, Parul Institute of Engg. & Technology, Vaghodia, Vadodara, India
mansi_vithalani@yahoo.com

Gordhan Jethava
Computer Science & Engineering, Parul Institute of Engg. & Technology, Vaghodia, Vadodara, India
g.jethava@gmail.com

Amit Ganatra
Computer Engineering Department, Charotar Institute of Technology, Changa, India.
amitganu@yahoo.com

Abstract

Sequential pattern mining is to find sequential purchasing behaviors for most customers from a large amount of customer transactions. Historical transaction data can be analyzed to discover customer purchasing habits. However, the size of the transaction database can be very large. It is very time consuming to find all the sequential patterns from a large database, and users may be only interested in some items. Sequential pattern mining based on constraint is now an important research issue of data mining, because it can reduce useless candidate generation as well as make the generated patterns meet the requirements of special users. Thus focus of this research paper is to find sequential patterns which satisfy various constraints and thus generating only interesting patterns and saving computational cost. We can push multiple constraints in sequential pattern mining to enhance the performance.

1. Introduction

Sequential pattern mining deals with data represented as sequences (a sequence contains sorted sets of items). Compared to the association rule problem, a study of such data provides “inter-transaction” analysis. It has been an active and important field of research and development since it was first introduced. Discovering sequential patterns from a large database of sequences is an important problem in the field of knowledge discovery and data mining. Briefly, given a set of data sequences, the problem is to discover subsequences that are frequent, in the sense that the percentage of data sequences containing them exceeds a user-specified minimum support.

Mining frequent sequential patterns has found a host of potential application domains, including the analysis of customer purchase patterns or Web access patterns, the analysis of sequencing or time related processes such as scientific experiments, natural disasters, and disease treatments, the analysis of DNA sequences, etc.

Motivation

The design of effective algorithms for mining frequent sequential patterns has been the subject of several studies in recent years. Sequential pattern mining algorithms, in general, can be categorized into three classes:

• Apriori-based, horizontal formatting method, with GSP Srikant and Agrawal (1996) as its representative;
• Apriori-based, vertical formatting method, such as SPADE (Zaki, 2001); and
• Projection-based pattern growth method, such as PSP (Pei et al., 2001) and SPAM (Ayres et al., 2002).

Although efficiency of mining the complete set of sequential patterns has been improved substantially, in many cases, sequential pattern mining still faces tough challenges in both effectiveness and efficiency. On the one hand, there could be a large number of sequential patterns in a large database. A user is often interested in only a small subset of such patterns. Presenting the complete set of sequential patterns may make the mining result hard to understand and hard to use. This brings in the effectiveness concern: “Can we only mine the sequential patterns that are highly interesting to users?”

On the other hand, although efficient algorithms have been proposed, mining a large amount of sequential patterns from large data sequence databases is inherently a computationally expensive task. If we can focus on only those sequential patterns interesting to users, we may be able to save a lot of computation cost by those uninteresting patterns. This opens a new
opportunity for performance improvement: “Can we improve the efficiency of sequential pattern mining by focusing only on interesting patterns?”

Fig.1 Pushing Constraints after pattern mining process

Fig.2 Incorporating constraints into pattern mining process

Ignoring small differences in the problem definition (e.g., form of input data, definition of a subsequence), a major common thread that runs through the vast majority of earlier work is the lack of user controlled focus in the pattern mining process. Typically, the interaction of the user with the pattern mining system is limited to specifying a lower bound on the desired support for the extracted patterns. The system then executes an appropriate mining algorithm and returns a very large number of sequential patterns, only some of which may be of actual interest to the user.

Despite its conceptual simplicity, this “unfocused” approach to sequential pattern mining suffers from two major drawbacks:

1. Disproportionate computational cost for selective users.
2. Overwhelming volume of potentially useless results.

The above discussion clearly demonstrates the need for novel pattern mining solutions that enable the incorporation of user-controlled focus in the mining process. To overcome these problems, researchers add constraints into sequence pattern mining to restrict mining sequences in small specific sets. Generally users specify a minimum support to collect patterns, the result of mining would return so many sequential patterns, and only a small part of these patterns may be meaningful to users.

2. Related Work

In [4] projection-based, sequential pattern-growth approach for efficient mining of sequential patterns was proposed. In this approach, a sequence database is recursively projected into a set of smaller projected databases, and sequential patterns are grown in each projected database by exploring only locally frequent fragments.

The generalization of the PrefixSpan algorithm to deal with gap constraints, using a new method to generate projected databases[8]. Studies on performance and scalability were conducted in synthetic and real-life datasets, and the respective results are presented.

From the data mining language, users can specify the interested items and the criteria of the sequential patterns to be discovered. Also, an efficient data mining technique is proposed to extract the sequential patterns according to the users’ requests. For designing a data mining language, two important issues need to be considered: the easy-to-use user interface and the efficient data mining language processing. This paper is concerned with these two issues. [10]

The use of Regular Expressions (REs) as a flexible constraint specification tool that enables user-controlled focus to be incorporated into the pattern mining process is represented in [5]. They develop a family of novel algorithms (termed SPIRIT – Sequential Pattern mining with Regular expression consTraints) for mining frequent sequential patterns that also satisfy user-specified RE constraints. The main distinguishing factor among the proposed schemes is the degree to which the RE constraints are enforced to prune the search space of patterns during computation.

3. Problem Statement

Sequential pattern mining is an important data mining problem with broad applications. However, it is also a difficult problem since the mining may have to generate or examine a combinatorially explosive number of intermediate subsequences.

Given an input database of sequences, we define a sequential pattern to be frequent if its support in the database exceeds a user-specified minimum support threshold. Prior work has focused on efficient techniques for the discovery of frequent patterns, typically ignoring the possibility of allowing and exploiting flexible structural constraints during the mining process.
A constraint C for sequential pattern mining is a boolean function C(a) on the set of all sequences. The problem of constraint-based sequential pattern mining is to find the complete set of sequential patterns satisfying a given constraint C.

4. Proposed Approach

From the application point of view, the following seven categories of constraints based on the semantics and the forms of the constraints can be used[6].

1. Item Constraint: An item constraint specifies subset of items that should or should not be present in the patterns. For example, when mining sequential patterns over a web log, a user may be interested in only patterns about visits to online bookstores.

2. Length Constraint: A length constraint specifies the requirement on the length of the patterns, where the length can be either the number of occurrences of items or the number of transactions. Length constraints can also be specified as the number of distinct items, or even the maximal number of items per transactions. For example, a user may want to find only long patterns (e.g., patterns consisting of at least 50 transactions) in bio-sequence analysis.

3. Super-pattern Constraint: A super-pattern constraint, to find patterns that contain a particular set of patterns as sub-patterns. For example, an analyst might want to find sequential patterns that first buy a PC and then buy a digital camera.

4. Aggregate Constraint: An aggregate constraint is the constraint on an aggregate of items in a pattern, where the aggregate function can be sum, avg, max, min, standard deviation, etc. For example, a marketing analyst may want sequential patterns where the average price of all the items in each pattern is over $100.

5. Recency Constraint: Recency constraint is specified by giving a recency minimum support (r_min_supp), which is the number of days away from the starting date of the sequence database. For example, if our sequence database is from 27/12/2007 to 31/12/2008 and we set r_min_supp = 200 then the recency constraint ensures that the last transaction of the discovered pattern must occur after 27/12/2007+200 days. In other words, suppose the discovered pattern is < (a), (bc)> , which means “after buying item a, the customer returns to buy item b and item c”. Then, the transaction in the sequence that buys item b and item c must satisfy recency constraint.

6. Monetary Constraint: Monetary constraint is specified by giving monetary minimum support (m_min_supp). It ensures that the total value of the discovered pattern must be greater than m_min_supp. Suppose the pattern is <(a), (bc)> . Then we can say that a sequence satisfies this pattern with respect to the monetary constraint, if we can find an occurrence of pattern <(a), (bc)> in this data sequence whose total value must be greater than m_min_supp.

7. Frequency Constraint: Frequency constraint is specified by giving frequency minimum support (f_min_supp). The frequency of a pattern is the percentage of sequences in database that satisfy the recency constraint and monetary constraint. And a pattern could be output as an RFM-pattern if its frequency is greater than f_min_supp. By setting these three constraints properly, we can discover an RFM-pattern like this, “30% customers who recently bought a computer would return to buy the scanner and microphone, and the total value will exceed Rs. 50,000”.

5. Proposed Algorithm

Algorithm: Mining sequential patterns with multiple constraints using PrefixSpan approach.

Input: A sequence database SDB, support threshold min_sup, and set of constraints C_set;

Output: The complete set of sequential patterns satisfying C_set;

Method:
call prefix_growth(<>, SDB, C_set);

Function prefix_growth(a, SDB[a])
// a: prefix; SDB[a]: the a-projected database
1. Let l be the length of a. Scan SDB[a] once, find length- (l + 1) frequent prefix in SDB[a], and remove infrequent items and useless sequences;
2. For each length- (l + 1) frequent prefix a’ potentially satisfying the constraint C_set do
   (a) if a’ satisfies C_set’, then output a’ as a pattern;
   (b) form SDB[a’];
   (c) call prefix_growth(a’, SDB[a’])

For example, If we represent a set of learning logs data of undergraduate students enrolled in a psychology course who used nStudy to study the textbook. As students select and use tools in nStudy, the system logs data to an XML file, consisting of a series of low-level events spaced along a time dimension, such as what tag is applied, at what time the tag is applied, and how long it takes to complete a “tagging” action.

The first round pattern mining returned 1806 patterns before imposing any constraints. After adding a length constraint of 10 (the maximum length of all patterns), we can get a fuller picture of students’ study patterns. If the first 15 min was known as a practicing session, a time constraint of 15 min can be applied to remove the “noise” actions within that period. If we are interested
in students who made notes, we can add an action constraint of “note-making” to retain patterns that include this action. If we request a super-pattern constraint of “Creation > Update > Review”, we can detect the antecedents or subsequent actions of this pattern. The filtering system can also incorporate context and distance constraints.

6. Implementation

The proposed technique has been implemented in JAVA using Eclipse IDE Indigo Release 1 version. There are total 13 java files in the original algorithm. Then it has been tested using synthetic datasets which contains sequence datasets.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Number of Customers</td>
</tr>
<tr>
<td>S</td>
<td>Average number of transactions per sequence</td>
</tr>
<tr>
<td>T</td>
<td>Average number of items per transaction</td>
</tr>
<tr>
<td>N</td>
<td>Number of distinct items</td>
</tr>
</tbody>
</table>

Table 1 The Parameters of Synthetic Dataset

For the purpose of implementing the Constrained Prefixspan algorithm for finding sequential patterns, the dataset generated by illimine synthetic data generator is used. The dataset is provided in text format. The parameters of synthetic dataset are shown in table given above.
7. Results

All the simulations were performed on a 2.70 GHz Pentium(R) 4 PC dual core CPU E5400 with 1 GB of main memory, running Windows 7. All the algorithms were implemented in JAVA using Eclipse IDE Indigo Release 1 version.
8. Conclusion

In this paper, we have presented the mining of sequential patterns using PrefixSpan algorithm to deal with multiple constraints. Thus applying multiple constraints on sequential patterns can effectively prune search space and thus efficiency of mining task increases extensively also effect of generated patterns are considerable that is only interesting patterns to user is generated.

The tests with large dataset showed that algorithm is scalable and the runtimes scaled with the size of the dataset. Since algorithms for frequent pattern enumeration and mining (breadth-first search) are not main memory based due to projected database construction, datasets which are larger than the available physical memory did not suffer due to operating system paging.

This research can be further preceded in several ways. Further we may consider about adding other useful constraints to the C-PrefixSpan, for example the quantitative constraint that the quantity of each item in a sequence must be no less than a given threshold or the number of repetitions of item in a sequence must be no longer than a given threshold. Sequential pattern-based clustering is also possible using C-PrefixSpan, where once a set of frequent patterns are found, we can organize objects into some clusters according to the patterns they share. Also researches could extend it by adding fuzzy constraints, so that the boundary is no longer fixed but flexible. As a future extension, the C-PrefixSpan algorithm can also be implemented on multiprocessor system.

References


[4] Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 10, October 2004.


