Supporting Interactive Sequential Pattern Discovery in Databases

Marek Wojciechowski

Poznan University of Technology
Institute of Computing Science
ul. Piotrowo 3a, 60-965 Poznan, Poland
marek@cs.put.poznan.pl

Abstract. One of the most important data mining problems is discovery of sequential patterns. Sequential pattern mining consists in discovering all frequently occurring subsequences in a collection of data sequences. This paper discusses several issues concerning possible extensions to traditional database management systems required to support sequential pattern discovery: a sequential pattern query language for specifying mining tasks and storing discovered patterns in the database, techniques of integrating various pattern constraints that can be specified in mining queries into the mining process in order to improve performance, and a framework for exploiting cached results of previous queries to support iterative and interactive data mining. The paper summarizes the author's recent research on the above topics.

1 Introduction

Data mining is a research area that aims at extracting previously unknown and potentially useful knowledge from large collections of data. Discovered knowledge is represented in the form of patterns, rules, or models depending on a chosen data mining method. The family of data mining problems that particularly attracted the database community is discovery of frequent patterns. It has been observed that frequent pattern mining resembles traditional database querying: a user specifies the source dataset, the class of patterns, and pattern constraints, and then a data mining system chooses the appropriate algorithm and returns the resulting patterns to the user [7]. The above statement initiated the research on data mining query languages, constraint-based pattern mining, and integrating data mining techniques into database management systems.

The most prominent classes of patterns are frequent sets (used to derive association rules) [1] and sequential patterns [2]. Similar techniques are used for both frequent sets and sequential patterns. However, the presence of time dependencies and time constraints makes sequential pattern mining a more challenging and interesting problem. Typically, novel solutions for frequent pattern mining are introduced in the context of frequent sets and then generalized or extended to handle sequential patterns.

Informally, sequential patterns are the most frequently occurring subsequences in sequences of sets of items. The frequency threshold, called support, is provided by a user. Additionally a user may specify time constraints that influence the sequence containment relationship. For discovery of sequential patterns with time constraints [17] proposed the *GSP* algorithm.

It is rather obvious that in order to simplify discovery of patterns that are really interesting to the user, data mining systems should allow users to specify additional constraints on patterns, e.g., referring to their size or contents. Such constraints, for the sake of efficiency, should be then exploited by the system to reduce the overall processing time to the level acceptable for interactive mining. Finally, since the user may need to execute a series of queries, adjusting various constraints, before he or she gets satisfying results, the system should be able to reuse the results of previous queries when possible. The last issue is particularly important in situations where several users execute similar mining queries on the same data.

In this paper we present our solutions supporting interactive discovery of sequential patterns in databases which could be applied to extend the functionality of traditional database management systems. In Section 2 we present basic definitions regarding sequential patterns. Section 3 describes the *MineSQL* data mining query language that can be used to declaratively specify sequential pattern discovery tasks in the form of mining queries. Section 4 discusses the dataset filtering technique to efficiently handle pattern constraints that can be specified in *MineSQL*. In Section 5 we present our framework for reusing results of previous mining queries. Section 6 presents related work. We conclude with a summary and directions for future work in Section 7.

2 Basic Definitions

Let $L = \{l_1, l_2, ..., l_m\}$ be a set of literals called items. An *itemset* is a non-empty set of items. A *sequence* is an ordered list of itemsets and is denoted as $<X_1\ X_2\ ...\ X_n>$, where X_i is an itemset ($X_i \subseteq L$). X_i is called an *element* of the sequence. Let D be a set of variable length sequences (called *data-sequences*), where for each sequence $S = <X_1\ X_2\ ...\ X_n>$, a timestamp is associated with each X_i.

With no time constraints we say that a sequence $X = <X_1\ X_2\ ...\ X_n>$ is *contained* in a sequence $Y = <Y_1\ Y_2\ ...\ Y_m>$ if there exist integers $i_1 < i_2 < ... < i_n$ such that $X_1 \subseteq Y_{i1}$, $X_2 \subseteq Y_{i2}$, ..., $X_n \subseteq Y_{in}$. We call $<Y_{i1}, Y_{i2}, ..., Y_{in}>$ an *occurrence* of X in Y. We consider the following user-specified time constraints while looking for occurrences of a given sequence: minimal and maximal gap allowed between consecutive elements of an occurrence of the sequence (called *min-gap* and *max-gap*), and *window-size*, which represents the maximal time window for several itemsets to form one element of the occurrence.

The *support* for the sequence $X = <X_1\ X_2\ ...\ X_n>$ in the set of data-sequences D is the percentage of data-sequences in D that contain X. A *sequential pattern* is a sequence whose support in D is above the user-specified threshold.

3 MineSQL Data Mining Query Language

Data mining systems to be effective have to support constraint-based mining, in which a user provides a set of constraints to guide the discovery process. One possible solution is to base the user interface on some ad hoc data mining query language, analogous to *SQL* for database systems, possibly designed as *SQL* extension.

In [10] we proposed a query language called *MineSQL* for discovery of various classes of frequent patterns in databases (a preliminary version of *MineSQL* was presented in [12]). *MineSQL* supports sequential pattern discovery and storage by providing data types and grouping functions to represent sets, sequences, and patterns, and the *MINE PATTERN* statement to specify sequential pattern mining tasks. The syntax of the *MINE PATTERN* statement is presented below:

```
MINE PATTERN
[MAXGAP maxgap] [MINGAP mingap] [TOLERANCE window-size]
FOR column
FROM table|(query)
[WHERE pattern_predicate [AND pattern_predicate…]];
```

In the above syntax, *maxgap*, *mingap*, and *window-size* represent values of the corresponding time constraints (defined as in [17]), *column* is the name of the table column or query column of the *SEQUENCE* data type containing data-sequences, *table* is the name of the table containing data-sequences, *query* represents the *SQL* subquery, returning the collection of data-sequences, and *pattern_predicate* represents pattern constraints in the form of a conjunction of basic Boolean pattern predicates concerning pattern structure or support. *MineSQL* defines the set of allowable basic Boolean pattern predicate types (and corresponding language constructs) to specify the support threshold, constraints on the pattern size (number of items in a pattern) and length (number of elements in a pattern), and to specify the requirement that a certain subsequence is or is not present in the pattern. Also constraints on a given pattern element are allowed (referring to its size or the presence of a certain subset).

The discovered patterns can be immediately stored in the database for further analyses by nesting a *MINE PATTERN* statement as a subquery in an *INSERT* statement or by creating a materialized view using a *MINE PATTERN* statement as a subquery in a *CREATE MATERIALIZE VIEW* statement.

Example. The following *MINE PATTERN* statement discovers sequential patterns from the *SHOPPING_HIST* table (we assume that the source data-sequences are stored as *SEQUENCE OF CHAR INDEX BY DATE* values in the *HIST* column). The returned set of patterns should consist of all sequential patterns having support greater than 10%, size less than 5, containing the subsequence "<(A)(B)>".

```
MINE PATTERN
FOR HIST
FROM SHOPPING_HIST
WHERE SUPPORT(PATTERN)>0.1
AND SIZE(PATTERN)<5
AND PATTERN CONTAINS TO_CHAR_PATTERN('<(A)(B)>');
```

4 Constraint-Based Sequential Pattern Mining Using Dataset Filtering Technique

While it is a user's task to specify the class of patterns of interest, the data mining system should be able to take the constraints specified in the query into account, in order to improve the efficiency of mining. As a consequence, there is a need for data mining algorithms and techniques exploiting user-specified constraints to optimize the discovery process. In [11] we considered our framework for handling pattern constraints, called dataset filtering, in the context of sequential pattern discovery.

We assume that sequential pattern discovery tasks are specified by means of the *MineSQL* data mining query language. Thus, pattern constraints have the form of a conjunction of basic Boolean pattern predicates concerning patterns or pattern elements. The dataset filtering technique consists in discarding data-sequences that cannot support any pattern satisfying pattern constraints specified by a user. In other words, dataset filtering optimizes pattern mining queries by pushing pattern constraints down into dataset selection queries.

Firstly, we identified basic Boolean predicates concerning patterns or pattern elements whose presence in pattern constraints of a sequential pattern query leads to the possibility of dataset filtering. Secondly, for each of the applicable basic Boolean pattern predicates, we provided the corresponding predicate concerning data-sequences. The matching pairs of predicates are presented in Table 1 (formal theorems and proofs regarding the mappings can be found in [11]). The meaning of the predicates is explained below the table.

Table 1. Basic Boolean predicates on patterns or pattern elements and corresponding data-sequence predicates

Basic Boolean predicate on a pattern or *n*-th element of a pattern	**Basic Boolean predicate on a data-sequence**
π(**SG**, α, pattern)	σ(**SG**, α, sequence)
π(**LG**, α, pattern)	σ(**CL**, α+1, sequence, *max*, *min*, *win*)
π(**C**, β, pattern)	σ(**C**, β, sequence, $+\infty$, *min*, *win*)
ρ(**SG**, α, pattern$_n$)	σ(**CS**, α+1, sequence, *win*)
ρ(**C**, γ, pattern$_n$)	σ(**C**, $<\gamma>$, sequence, *max*, *min*, *win*)

- π(**SG**, α, pattern) – true if the size of the pattern is greater than α;
- π(**LG**, α, pattern) – true if the length of the pattern is greater than α;
- π(**C**, β, pattern) – true if the β is a subsequence of the pattern;
- ρ(**SG**, α, set) – true if the size of the set is greater than α;
- σ(**CL**, α, sequence, maxgap, mingap, window) - true if there exists a sequence of length α that is contained in the sequence with respect to the max-gap, min-gap, and window-size constraints.
- ρ(**C**, γ, set) – true if γ is a subset of the set;
- σ(**SG**, α, sequence) – true if the size of the data-sequence is greater than α;
- σ(**C**, β, sequence, maxgap, mingap, window) – true if the data-sequence contains the pattern β using given time constraints;

- σ(**CS**, α, sequence, window) - true if there exists a 1-element sequence of size α that is contained in the sequence with respect to the window-size constraint.

Example. Consider the following sequential pattern query expressed in *MineSQL*:

```
MINE PATTERN
MAXGAP 100 MINGAP 7
FOR HIST
FROM SHOPPING_HIST
WHERE SUPPORT(PATTERN)>0.01
AND SIZE(PATTERN)>3
AND PATTERN CONTAINS TO_CHAR_PATTERN('<(A)(B)>');
```

The above query returns sequential patterns having support greater than 1%, having more than three items, containing the sequence <(A)(B)>. The specified values of max-gap and min-gap constraints are 100 and 7 respectively. The time window constraint is not specified explicitly, therefore the default value of 0 is going to be used. There are two basic Boolean pattern predicates supporting dataset filtering in the *WHERE* clause of the above query: π(**SG**, 3, pattern) and π(**C**, <(A)(B)>, pattern). Thus, according to Table 1, the following dataset filtering predicate has to be satisfied by a data-sequence containing a pattern satisfying given pattern constraints: σ(**SG**, 3, sequence) $\wedge$ σ(**C**, <(A)(B)>, sequence, +∞, 7, 0). The dataset filtering predicate says that only data-sequences having more than three items and containing the sequence <(A)(B)> with max-gap of +∞, min-gap of 7, and window-size of 0, have to be considered in the discovery process.

Dataset filtering can be applied to any sequential pattern discovery algorithm since it conceptually leads to an equivalent data mining task on a possibly smaller dataset. However, instead of explicit transformation of the mining task, we proposed integrating dataset filtering within mining algorithms. So far, we focused on implementation details concerning integration of dataset filtering with the *GSP* algorithm. Experiments show that dataset filtering can result in significant performance improvements, especially in case of pattern constrains concerning pattern contents (e.g. the presence of a certain subsequence), which we believe are the most useful. The actual performance gains depend on the selectivity of the derived dataset filtering predicate.

5 Efficient Sequential Pattern Query Processing in the Presence of Materialized Results of Previous Queries

A user interacting with a data mining system has to specify several constraints on patterns to be discovered. However, usually it is not trivial to find a set of constraints leading to the satisfying set of patterns. Thus, users are very likely to execute a series of similar data mining queries before they find what they need. To facilitate such interactive and iterative mining, data mining query optimizers have to be able to significantly reduce processing times in case of queries similar to the queries whose results have been previously discovered. To make such optimizations possible, data mining systems have to support materialization (or caching) of query results.

In [18] we analyzed differences between sequential pattern queries that allow the system to efficiently answer a query using cached (materialized) results of another query. We based our work on [3] where three relationships between mining queries (equivalence, inclusion, and dominance), representing cases when one query can be answered using the results of another query with at most one scan of the source dataset, were identified. In [3] only association rule queries were considered. In [18] we proved which kinds of syntactic differences between sequential pattern queries that can be expressed in *MineSQL* lead to the inclusion and dominance relationships.

Firstly, we formally defined the relationships of *extending pattern constraints* and *extending time constraints* within the constraint model on which *MineSQL* is based. In the next step, we proved that for two sequential pattern queries DMQ_1 and DMQ_2, operating on the same dataset:

1. if DMQ_1 and DMQ_2 have the same time constraints and DMQ_2 extends pattern constraints of DMQ_1, then DMQ_1 includes DMQ_2.
2. if DMQ_1 and DMQ_2 have the same pattern constraints and DMQ_2 extends time constraints of DMQ_1, then DMQ_1 dominates DMQ_2.
3. if DMQ_2 extends pattern constraints of DMQ_1 and DMQ_2 extends time constraints of DMQ_1, then DMQ_1 dominates DMQ_2.

In [18] we also provided sequential pattern query processing algorithms for the above three cases. The general idea is the following: In case of inclusion (1), only filtering of materialized query results is required. In case of dominance (2 or 3), one scan of the source dataset is necessary to reevaluate the supports of patterns (for the third case this scan is preceded by filtering of the patterns). The experiments proved that the proposed solutions typically reduce processing time by several orders of magnitude when materialized results of previous queries are available. In [9] we analyzed situations when more than one of previously materialized queries can be used to answer the current query. We provided cost models for our sequential pattern query processing schemes that can be used to chose the optimal strategy.

Example. Let us consider the following example: we are given materialized results of query *MDMQ* and a new query *DMQ* issued by a user.

DMQ:
```
MINE PATTERN
MAXGAP 10
 FOR HIST
 FROM SHOPPING_HIST
 WHERE SUPPORT(PATTERN)>0.2
AND SIZE(PATTERN)<5;
```

MDMQ:
```
MINE PATTERN
MAXGAP 10
 FOR HIST
 FROM SHOPPING_HIST
 WHERE SUPPORT(PATTERN)>0.1;
```

DMQ and the defining query of *MDMQ* operate on the same dataset, have the same time constraints, and *DMQ* extends pattern constraints of *MDMQ* (*DMQ* has more restrictive support threshold and adds one predicate regarding the size of patterns). Thus, *MDMQ* includes *DMQ*, and therefore *DMQ* can be answered by filtering the results of *MDMQ* using pattern constraints of *DMQ*.

6 Related Work

Many declarative data mining query languages have been proposed so far. In [6] a multi-purpose, *SQL*-like data mining query language, called *DMQL* was introduced. *DMQL* can be used for various mining tasks such as rule discovery, classification, clustering, etc. On the other hand, several single-purpose languages intended only for association rule discovery problem have been proposed, e.g., *MINE RULE* operator [4], and *M-SQL* language [8]. Functionality of our language – *MineSQL* covers only discovering frequent patterns, the family of techniques that really can be regarded as advanced database querying. We do not provide constructs for techniques like clustering or classification, where the results may depend on a chosen data mining algorithm. Nevertheless, *MineSQL* is still a universal language as it considers several classes of patterns.

Constraint-based mining was extensively studied in the context of association rules (e.g., [13]). Only a few works considered constraint-based sequential pattern mining. Some interesting research results were presented in [5], where a family of algorithms, called *SPIRIT*, discovering patterns with regular expression constraints was introduced. *SPIRIT* algorithms can be regarded as extensions of the classic *GSP* algorithm using advance techniques to prune the pattern search space. Recently, a pattern-growth family of algorithms, fundamentally different from *GSP* has been proposed (e.g., *PrefixSpan* [15]) for discovery of sequential patterns. In [16] handling time, regular expression, and various pattern constraints within the pattern-growth framework was discussed. The positive aspects of our dataset filtering method, compared to other solutions, are: intuitive and simple constraint model, possibility of integration with other constraint handling methods, independence of a particular mining scheme, and the ease of use by a query optimizer (query rewriting).

To facilitate interactive and iterative pattern discovery, [14] proposed to materialize patterns discovered with the least restrictive selection criteria, and answer incoming queries by filtering the materialized pattern collection. This approach might lead to collections of frequent patterns even larger than the original database. Moreover, restricting time constraints not only makes some patterns infrequent but also changes the support of patterns that remain frequent. Much more reasonable and flexible solutions supporting interactive and iterative mining were presented in [3] in the context of association rules. The solutions presented there consisted in caching results of mining queries. The most important contribution was the identification of three relationships between results of mining queries, called equivalence, inclusion, and dominance. Syntactic differences between association rule queries leading to the three relationships were analyzed, and efficient query answering algorithms were proposed. As we mentioned earlier, our technique for reusing results of previous sequential pattern queries is the adaptation of the approach from [3].

7 Conclusions and Future Work

We have presented our solutions supporting interactive discovery of sequential patterns in databases which could be applied to extend the functionality of traditional database management systems: *MineSQL* data mining query language, the dataset filtering technique for efficient constraint-based sequential pattern mining, and a framework for exploiting cached results of previous sequential pattern queries.

In the future we plan to further investigate data mining query optimization issues in the context of sequential patterns and other classes of patterns, focusing on cost models and multi-query optimization.

References

1. Agrawal R., Imielinski T., Swami A.: Mining Association Rules Between Sets of Items in Large Databases. Proc. of the 1993 ACM SIGMOD Conf. on Management of Data (1993)
2. Agrawal R., Srikant R.: Mining Sequential Patterns. Proc. of the 11th Int'l Conference on Data Engineering (1995)
3. Baralis E., Psaila G.: Incremental Refinement of Mining Queries. Proceedings of the 1st DaWaK Conference (1999)
4. Ceri S., Meo R., Psaila G.: A New SQL-like Operator for Mining Association Rules. Proc. of the 22nd Int'l Conference on Very Large Data Bases (1996)
5. Garofalakis M., Rastogi R., Shim K.: SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. Proceedings of 25th VLDB Conference (1999)
6. Han J., Fu Y., Wang W., Chiang J., Gong W., Koperski K., Li D., Lu Y., Rajan A., Stefanovic N., Xia B., Zaiane O.R.: DBMiner: A System for Mining Knowledge in Large Relational Databases. Proc. of the 2nd KDD Conference (1996)
7. Imielinski T., Mannila H.: A Database Perspective on Knowledge Discovery. Communications of the ACM, Vol. 39, No. 11 (1996)
8. Imielinski T., Virmani A., Abdulghani A.: Datamine: Application programming interface and query language for data mining. Proc. of the 2nd KDD Conference (1996)
9. Morzy M., Wojciechowski M., Zakrzewicz M.: Cost-Based Sequential Pattern Query Optimization in Presence of Materialized Results of Previous Queries. Proc. of the IIS'2002 Symposium (2002)
10. Morzy T., Wojciechowski M., Zakrzewicz M.: Data Mining Support in Database Management Systems. Proc. of the 2nd DaWaK Conference (2000)
11. Morzy T., Wojciechowski M., Zakrzewicz M.: Efficient Constraint-Based Sequential Pattern Mining Using Dataset Filtering Techniques, Proc. of the 5th International Baltic Conference on Databases and Information Systems (2002)
12. Morzy T., Zakrzewicz M.: SQL-like Language for Database Mining. ADBIS'97 Symposium (1997)
13. Ng R., Lakshmanan L., Han J., Pang A.: Exploratory Mining and Pruning Optimizations of Constrained Association Rules. Proc. of the 1998 ACM SIGMOD Conference (1998)
14. Parthasarathy S., Zaki M.J., Ogihara M., Dwarkadas S.: Incremental and Interactive Sequence Mining. Proceedings of the 1999 ACM CIKM Conference (1999)
15. Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U., Hsu M-C.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. Proceedings of the 17th International Conference on Data Engineering (2001)
16. Pei J., Han J., Wang W.: Mining Sequential Patterns with Constraints in Large Databases. Proc. of the 2002 ACM CIKM Conference (2002)
17. Srikant R., Agrawal R.: Mining Sequential Patterns: Generalizations and Performance Improvements. Proc. of the 5th Int'l Conf. on Extending Database Technology (1996)
18. Wojciechowski M.: Interactive Constraint-Based Sequential Pattern Mining, Proc. of the 5th ADBIS Conference (2001)