Monitoring Change in Mining Results

Steffan Baron[1] and Myra Spiliopoulou[2]

[1] Institut für Wirtschaftsinformatik, Humboldt-Universität zu Berlin
sbaron@wiwi.hu-berlin.de
[2] Institute of Business and Technical Information Systems,
Otto-von-Guericke Universität Magdeburg
myra@iti.cs.uni-magdeburg.de

Abstract. In the last years the datasets available have grown tremendously, and the development of efficient and scalable data mining algorithms has become a major research challenge. However, since the data is more dynamic than static there is also a strong need to update previously discovered rules and patterns. Recently, a couple of studies have emerged dealing with the topic of incremental update of discovered knowledge. These studies mostly concentrate on the question whether new rules emerge or old ones become extinct.
We present a framework that enables the analyst to monitor the changes a rule may undergo when the dataset the rules were discovered from is updated, and to observe emerging trends as data change. We propose a generic rule model that distinguishes between different types of pattern changes, and provide formal definitions for these. We present our approach in a case study on the evolution of web usage patterns. These patterns have been stored in a database and are used to observe the mining sessions as snapshots across the time series of a patterns lifetime.

1 Introduction

While there are many sophisticated techniques for the discovery of knowledge in a large range of domains, methods for knowledge maintenance and updating are emerging at a slower pace. In recent years a considerable number of studies aiming on the update of previously discovered knowledge have emerged. Most of those works focus on the refreshment of the mining results when the underlying dataset is updated. They investigate methods that actualize the mining results on the basis of the data update only, i.e., without scanning the whole dataset again. More recent works also consider the *lifetime* of a rule, i.e., the time interval in which it is sufficiently supported by the data.

In this study, we propose a framework for updating mining results *and* observing their evolution, thus effectively combining both research trends in a general framework. Our work is motivated by the need for sustainable knowledge: beyond updating knowledge, we need a mechanism monitoring its evolution. For example, consider an association rule "Product A ⇒ Product B" appearing in multiple consecutive mining sessions. A decrease in the confidence of this rule would imply that the two products are less strongly correlated than before. To

Y. Kambayashi, W. Winiwarter, and M. Arikawa (Eds.): DaWaK 2001, LNCS 2114, pp. 51–60, 2001.

observe this decrease, we propose the modeling of rules as temporal objects and of their statistics as time series of snapshots corresponding to the mining sessions. Our work differs from other approaches in this domain in a number of aspects. First, rather than only investigating techniques to discover rules that have newly emerged or become extinct after an update, we focus on the changes a specific rule may undergo through several updates to the dataset. We model rules as temporal objects, on which techniques for time series analysis can be applied to discover trends and trend similarities. Second, we propose a generic rule model appropriate for more than one rule type. Third, we consider two aspects of rule evolution, namely changes in the statistics of a rule, as in the above example, and changes in the content of a rule. Based on these types of change, we give formal definitions of possible types of pattern evolution.

The article is organized as follows. In the next section, we give an overview of related work. In Sec. 3, we present our framework, in which rules are modeled as temporal objects. We then distinguish between different types of pattern evolution. In Sec. 4, we demonstrate the advantages of our framework in a case study on monitoring user navigation patterns. Sec. 5 summarizes the study and gives a brief outlook on future work.

2 Related Work

The earliest incremental mining algorithms have been designed for the refreshment of association rules. This mining paradigm is still the focus of the majority of contributions in this domain. The aim is to avoid a complete re-run of the mining algorithm on the whole dataset when new data is added. The most notable works in this area are by Cheung et al. [6, 5]. They deal with the problem of maintaining discovered rules after updates on the original database. An algorithm FUP_2 is proposed which uses information from previous mining sessions to maintain rules after *insertions* and *deletions* in the original dataset. The basic idea is that an itemset X may only remain large after an update to the original database, if it is large in both the original database and in the update. Ayan et al. propose an algorithm *UWEP* which also deals with the incremental update of association rules [1]. Similar to FUP_2 algorithm the aim is the efficiency in performing the update. The main performance advantage of *UWEP* is achieved by pruning away the supersets of a large itemset in DB as soon as they are known to be small in the updated database $DB + \delta$. This methodology yields a much smaller candidate set especially when the set of new transactions does not contain some of the old large itemsets. In [11], an incremental version of the well known *Partition* algorithm for association rules discovery is proposed. Basically, DB is used as one partition, and δ is used as another partition. Then, a slight variation of the original algorithm is used to update the patterns. In this study, efficiency is achieved by minimizing the number of scans over DB and δ. Thomas et al. propose an algorithm which reuses not only the large itemsets from previous mining sessions but also their *negative borders*, the candidate itemsets

which did not have enough support to become large [14]. A similar approach was developed independently by Feldman et al. at about the same time [8].

There have also been studies based on other mining paradigms. Comparable work which emphasizes on dynamic sequential data has been done by Wang et al. [15]. They present a time independent algorithm for the incremental discovery of patterns in large and dynamic sequential data which takes advantage of already discovered patterns and computes the change by accessing only the affected part of the dataset. Ester et al. propose a similar approach for clustering in a data warehouse environment [7]. They present an incremental clustering algorithm, based on DBSCAN, that examines which part of an existing clustering is affected by an update of the database and adjusts the clusters accordingly. For a detailed description of these studies see the general overview of pattern evolution in [2].

Another part of related work focuses more on the similarity between rules and on the temporal aspects of rules. Ganti et al. compute a deviation measure which makes it possible to quantify the difference between two datasets in terms of the model they induce, called `FOCUS` [9]. A new notion of surprising temporal patterns and algorithms to compute these is proposed by Chakrabarti et al. [3]. They argue that once the analyst is already familiar with prevalent patterns in the data, the greatest incremental benefit is likely to be from changes in the relationship between item frequencies over time. [4] concentrate on the identification of interesting temporal features like valid period or periodicity of association rules. The first two approaches take similarity of rules and changes a rule may undergo into account, the latter approach temporal properties of a rule. But these two aspects of a rule belong together, and should not be considered separately. Therefore, we model both aspects as a time series of changes a rule is subjected to in time. In a recent work Ganti et al. introduce a new dimension which takes the temporal aspect of the collected data into account, called *data span dimension* [10]. Here, the analyst can specify an arbitrary time window from which the data is taken for the analysis. Also they describe new model maintenance algorithms with respect to the selection constraints on the data span dimension for frequent itemsets and clustering.

The main difference to our contribution is that they consider a certain time span, but only at a fixed point in time. The analyst can choose to mine all the data available so far, or a block of data that is more suited for the purpose of the analysis. But the analysis itself is limited to that fixed time point treating the analyzed data as a whole, possibly overseeing interesting changes of a rule at the end of the specified time window, for example.

3 A Framework for Monitoring Pattern Evolution

As mentioned above there are two different types of rule change. First, the statistics of a rule may change. For example, an association rule $X \Rightarrow Y$ may hold with less confidence after adding the increment database δ because a smaller percentage of transactions in the updated database $DB + \delta$ that contain X also contain Y. Second, the content of a rule may change. For example, after updating

the database an association rule may become less restrictive by having more items in the consequent than before, or a user navigation pattern may follow a different path, though leading from the same start page to the same end page.

3.1 A Generic Rule Model

We model in a rule both its content and its statistics and, along with the timestamp of the mining session which has produced it and a key which uniquely identifies the rule across consecutive mining sessions. Then, a rule is a temporal object with the following signature:

$$R = (ID, query, timestamp, stats, body, head)$$

Here, ID is the unique identifier of rule R which is built from *body* and *head* of that rule, *query* is the mining query which has produced it at the given *timestamp*, and *body* and *head* describe the antecedent and the consequent of R, respectively. For the statistics of a rule, denoted by *stats*, we need a special treatment because some values refer to the entire rule, whereas other values only refer to a specific item. For example, in a rule $X \Rightarrow Y$ the support of the single item X or Y may also be of interest. In order to overcome this problem, we model the statistics concerning a single item with the item as part of the body or head of a rule directly in the rules content, and denote only statistics which refer to the rule as a whole by *stats*.

Example. In the simplest case of association rule discovery, a query Q specifies a lower boundary on the permitted support and an upper boundary on the rule length. We observe a mining session as the run of a mining query over the dataset, producing a quite large set of rules in the general case. Using the above notation, an association rule $A \Rightarrow B$ with support $s = 10\%$ and confidence $c = 60\%$ produced by query Q at time stamp t_0 would be modeled as $R_0 = (ID_{AB}, Q, t_0, [s = 10\%, c = 60\%], A, B)$ where ID_{AB} denotes an identifier that accompanies the rule across all mining sessions and makes it uniquely identifiable, as long as body and head of the rule does not change. □

3.2 Different Types of Patterns

A rule base does not consist solely of association rules. A sequence miner outputs frequent sequences, while a clustering algorithm computes clusters of similar items. When modeling a rule base, these types of rules should also be taken into account.

Without loss of generality, for frequent sequences we define the whole sequence but the last item to be the body of the rule, and the last item to be the head of the rule. Then, a frequent sequence $ABCDE$ is modeled as $ABCD \Rightarrow E$. The statistics for each item are directly stored in the item, i.e., in the body or head of the rule. The same yields for more complex sequences like web usage patterns. As a first step, the start page and the end page of a navigation can

be modeled as an association rule of the form *startpage* ⇒ *endpage*. However, since this leads to an information loss, the actual navigation pattern is decomposed into paths which can be treated just as sequences. For example, consider a frequent navigation pattern starting in page `S.html` and ending in `E.html` which consists of two paths. On the first path, users accessed two pages `T1.html` and `T2.html` on their way from `S.html` to `E.html`, on the second path, they visited one and the same page `T3.html`. This navigation pattern is modeled as two sequential patterns `S`, `T1`, `T2` ⇒ `E` and `S`, `T3` ⇒ `E`.

For clusters we use a similar approach: each cluster is identified by its centroid and is decomposed into its constituent items by giving the value of each item, along with the centroid of the cluster it belongs to and the distance to the centroid. The advantage of this approach is that it also works for a density based clustering. Then, a cluster is a set of association rules of the form $item, centroid, distance \Rightarrow cluster$. However, as with complex navigation patterns, rather than a single item the body and head of a rule would be a set of items. Therefore, we extend the generic rule model to subsume also more complex types of patterns:

$$R = (ID, query, timestamp, stats[\,], body[\,], head[\,])$$

Then, a simple association rule with a single item in the antecedent and a single item in the consequent is just modeled as $body[0] \Rightarrow head[0]$.

3.3 Monitoring Pattern Change

As described in Sec. 1, we consider two types of changes that may affect a pattern: changes to its statistics and changes to its content. Regarding changes to the statistics of a rule, we give the following definition for the *evolution* of a rule:

Definition 1 *As* evolution *of a rule we denote changes to its statistical measurements in time.*

When monitoring a rules statistics we need to identify the rule non-ambiguously through several mining sessions. For this purpose, a unique key is derived from body and head of the rule. Hence, if the content of the rule changes from one mining session to another, the identifier also changes, and we say the rule has died:

Definition 2 *Changes to the contents of a rule and changes to the statistics of a rule that violate given thresholds lead to its* death.

However, this approach may be considered too restrictive since it circumvents monitoring of changes to the contents of a rule. When monitoring content changes the above definitions are not sufficient:

Definition 3 *As* mutation *of a rule we denote changes to the contents of a rule provided that given thresholds are not violated.*

However, a global identifier is still needed. For example, consider an association rule $AB \Rightarrow CD$ which is generated from the initial dataset. After an update we apply some incremental mining algorithm, and obtain the rule $AB \Rightarrow CE$. Here, the question arises whether there has emerged a new rule or if the old rule has mutated. Only if each rule is uniquely identifiable, we can answer this question. To derive the identifier, the analyst could decide to use only the rules body, with the effect that one identifier may represent more than one rule, i.e., all rules with the same body. The question of how such an identifier can be produced is the subject of future work.

Now we consider the case of a rules death. When it dies, it is still important to know why it disappeared. In general, the analyst specifies minimum thresholds for support and confidence to observe patterns from data. Then, all rules that satisfy the given thresholds are found. After an update, it is possible for a rule to vanish if it does not fit the requirements defined by the analyst. For monitoring these rules three different approaches are reasonable. First, exactly those threshold values given by the analyst are used to discover rules. This would imply that the analyst needs to adjust the threshold values and re-run the mining algorithm to locate those rules that have vanished. Second, internally lower values for support and confidence as those given by the analyst are used to discover additional rules. These rules are then used to locate rules that disappeared through an incremental mining run, as opposed to external rules that are presented to the analyst. Finally, the extreme is not to restrict support and confidence, and produce all possible rules from the dataset. However, again only those rules are presented to the analyst that fulfill the given requirements. On the other hand, it is also valuable to monitor emerging rules. Then, a complete time series for the entire lifespan of a pattern can be derived.

4 Case Study

In order to prove our concepts we investigated the evolution of frequent sequences as produced by the Web Usage Miner (WUM), a tool for analyzing the behavior of a web sites visitors developed at Humboldt-University Berlin [12]. Using WUM, the analyst can specify a template expressed in the MINT query language which describes the navigation patterns he is interested in [13]. Moreover, the number of discovered patterns can be limited by giving constraints on support and confidence for the pages visitors accessed throughout their navigation.[1] The results WUM produces are generalized sequences (*g-sequences*) which are instances of the template given by the analyst, where each g-sequence consists of a non-empty set of navigation patterns [12].

The dataset used for the case study is a common access log file as produced by a web server spanning seven months. From the data three equally spaced samples were drawn, each representing a single month. All page names in the log file were mapped to a concept hierarchy, where each concept represents pages

[1] For a complete example see `http://wum.wiwi.hu-berlin.de`.

with a particular subject on the web server. Then, rather than navigation patterns between single pages navigation patterns within these concepts have been analyzed. For the analysis we used the first and the last approach as described in Sec. 3.3.

4.1 The Constraint-Based Approach

In the first mining run the following MINT query was issued which generated all navigation patterns up to length 5 where the start page has an absolute support of more than 100 visitors and the end page a minimum support of more than 50 visitors within the same navigation pattern, and more than 20% of the users started in page `x` should also visit page `y`.

```
SELECT t
 FROM NODE AS x y,
 TEMPLATE x [0;5] y AS t
 WHERE x.support > 100
 AND y.support > 50
 AND ( y.support / x.support ) > 0.2
```

The results comprise a number of g-sequences that match the given template. A sample of results is shown in Fig. 1. Each g-sequence consists of a pattern identifier which accompanies the pattern uniquely within the mining session, an antecedent (the start page of a navigation) and a consequent (the end page of a navigation). Antecedent and consequent consists of the page identifier, the page name (the concept name in our example), the occurrence (how often this page was accessed), and the absolute number of visitors accessed this page within their navigation. A g-sequence is uniquely identified across all mining sessions by its antecedent and consequent, i.e., the concept name along with its occurrence.

The results of the mining run were imported into a relational DBMS according to our generic rule model. Fig. 2 shows the sample of g-sequences depicted in Fig. 1 as stored in the database. For simplicity, the results were not normalized before importing them in the DBMS. Instead, all attributes of a pattern were stored in one single relation, i.e., in one separate relation for the results from each month, where the key of the relation consists of four attributes, `ant_page`, `ant_occ`, `con_page` and `con_occ`. The values for the confidence attribute were computed during the import.

```
PatternID=1: [3000016;I143;2;132], [3000016;I143;3;93]
PatternID=2: [3000006;I111;1;738], [3000013;I114;1;153]
PatternID=3: [3000002;I13;1;274], [3000004;I135;1;69]
PatternID=4: [3000008;I11;1;320], [3000006;I111;1;143]
PatternID=5: [3000013;I114;1;407], [3000013;I114;2;172]
```

Fig. 1. Sample of g-sequences as produced by WUM.

ant_page	ant_occ	con_page	con_occ	ant_supp	con_supp	conf
I143	2	I143	3	132	93	0.7045
I111	1	I114	1	738	153	0.2073
I13	1	I135	1	274	69	0.2518
I11	1	I111	1	320	143	0.4469
I114	1	I114	2	407	172	0.4226

Fig. 2. Sample of g-sequences as stored in the DBMS.

In order to identify patterns that have emerged or disappeared through the different mining sessions, several SQL queries were issued. We found 8 patterns that disappeared and 3 patterns that emerged of a total of 27 patterns during the second mining session. Through mining the third dataset 5 patterns disappeared and 3 patterns emerged. After that, we investigated the statistics of those patterns. Fig. 3 is a time series of the statistics of a sample pattern which was drawn over the different mining sessions. The first diagram shows a rising support of the antecedent of the pattern. However, since the support of the consequent does not rise equally we encounter a loss of confidence for that pattern.

Summarizing our results, it turned out that the differences which led to changes in the result sets were very small. Therefore, we continued to analyze the changes that might be interesting using the third approach mentioned in Sec. 3.3.

4.2 The Unlimited Approach

This approach is mainly based on the idea by Chakrabarti et al. mentioned in Sec. 2 [3]. We extend this idea by introducing a *difference* parameter for each statistical measurement which can be used to identify interesting changes. However, since also strong changes of patterns that do not have minimum support and confidence might be interesting, we applied this parameter on each pattern discovered from the dataset.

We implemented a simple *monitor* which takes the discovered patterns and the difference parameters as inputs and outputs all those patterns also showing

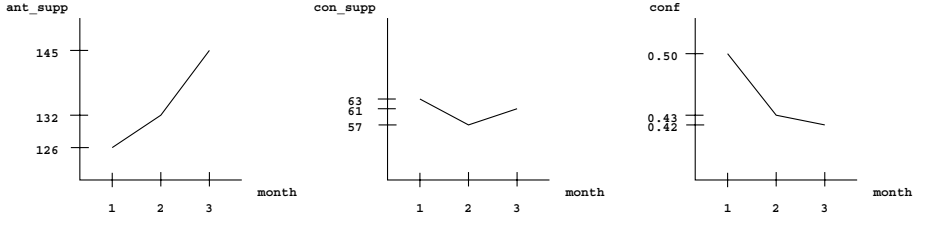

Fig. 3. A time series of a rules statistics.

the specified strength in their change. For this purpose, all patterns have been generated from the datasets. After importing them into the DBMS a series of SQL statements produces a set of patterns that meet the requirements given by the analyst. For example, if a rise of 20% in the support of the antecedent is considered interesting, the following SQL query would generate these patterns:

```
SELECT n.ant_page, n.ant_occ, n.con_page, n.con_occ,
    (n.ant_supp / (f.ant_supp - n.ant_supp)) AS delta
  FROM november n, february f
  WHERE n.ant_page = f.ant_page
  AND   n.ant_occ = f.ant_occ
  AND   n.con_page = f.con_page
  AND   n.con_occ = f.con_occ
  AND   abs(n.ant_supp / (f.ant_supp - n.ant_supp)) > 0.2
```

We computed a large set of patterns that showed interesting changes. However, it turned out that this approach only led to useful results if lower boundaries for support and confidence were also given. Moreover, there is a big overhead when generating all patterns from the datasets.

5 Conclusions and Future Work

In this work we have developed a framework which allows the management and the maintenance of mining results. We model the rules output by consecutive mining sessions as temporal objects, which may change in terms of statistics and in terms of contents. To monitor statistical changes of a rule, we treat rule statistics as time series, on which prediction and similarity searches can be performed using conventional techniques of time series analysis. Changes in a rules contents are harder to monitor. We are confronted with the fundamental question of whether a change in the contents of a rule should be observed as a mutation of the rule or as a new rule that has replaced the old one. As far as not given directly by the data, our framework delegates this decision to the analyst who can explicitly define which changes to a rule contents should be observed as rule "mutations", thus effectively supplying a unique identifier for rules across the mining sessions. In a case study we have shown the applicability of the proposed framework. Moreover, we have implemented a *monitor* which can observe interesting changes in these patterns.

The proposed work is only a first step in the direction of supporting the maintenance and monitoring of non-crisp knowledge that evolves across mining sessions. It should be extended in several ways. Firstly, we intend to combine our framework with a time series analysis tool and establish a complete environment for the monitoring of the statistic evolution of rules. Secondly, we will extend our approach to investigate also changes of a rules contents, especially when a global key is missing. In the case study we have also discovered patterns that disappeared from one mining session to another and re-appeared in the following mining session. To identify those rules without a global identifier is also challenging. Finally, the incorporation of a theorem prover that assesses new rules from

the non-crisp mining results seems an indispensable component of a full-fledged knowledge base.

References

1. N. F. Ayan, A. U. Tansel, and E. Arkun. An Efficient Algorithm To Update Large Itemsets With Early Pruning. In *Proc. of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 287–291, San Diego, CA, USA, August 1999. ACM.
2. S. Baron and M. Spiliopoulou. Monitoring the Results of the KDD Process: An Overview of Pattern Evolution. In J. M. Meij, editor, *Converting Data into Knowledge*, chapter 5. The Netherlands Study Center for Technology Trends, Den Haag, Netherlands, to appear in Sep. 2001.
3. S. Chakrabarti, S. Sarawagi, and B. Dom. Mining Surprising Patterns Using Temporal Description Length. In *VLDB'98*, pages 606–617, New York, USA, Aug. 1998. Morgan Kaufmann.
4. X. Chen and I. Petrounias. Mining Temporal Features in Association Rules. In *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, pages 295–300, Prague, Czech Republic, September 1999. Springer.
5. D. W. Cheung, S. D. Lee, and B. Kao. A General Incremental Technique for Maintaining Discovered Association Rules. In *DASFAA'97*, Melbourne, Australia, Apr. 1997.
6. D. W. Cheung, V. T. Ng, and B. W. Tam. Maintenance of Discovered Knowledge: A Case in Multi-Level Association Rules. In *KDD'96*, 1996.
7. M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental Clustering for Mining in a Data Warehousing Environment. In *VLDB'98*, pages 323–333, New York, USA, August 1998. Morgan Kaufmann.
8. R. Feldman, Y. Aumann, A. Amir, and H. Mannila. Efficient Algorithms for Discovering Frequent Sets in Incremental Databases. In *DMKD'97*, Tucson, USA, Mai 1997.
9. V. Ganti, J. Gehrke, and R. Ramakrishnan. A Framework for Measuring Changes in Data Characteristics. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 126–137, Philadelphia, USA, May 1999. ACM Press.
10. V. Ganti, J. Gehrke, and R. Ramakrishnan. DEMON: Mining and Monitoring Evolving Data. In *Proceedings of the 15th International Conference on Data Engineering*, pages 439–448, San Diego, USA, February 2000. IEEE Computer Society.
11. E. Omiecinski and A. Savasere. Efficient Mining of Association Rules in Large Databases. In *BNCOD'98*, pages 49–63, 1998.
12. M. Spiliopoulou. The Laborious Way from Data Mining to Web Mining. *Int. Journal of Comp. Sys., Sci. & Eng., Special Issue on "Semantics of the Web"*, 14:113–126, Mar. 1999.
13. M. Spiliopoulou and L. C. Faulstich. WUM: A Tool for Web Utilization Analysis. In *EDBT Workshop WebDB'98*, Valencia, Spain, Mar. 1998. Springer.
14. S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases. In *KDD-97*, pages 263–266, Newport Beach, USA, Aug. 1997.
15. K. Wang. Discovering Patterns from Large and Dynamic Sequential Data. *Intelligent Information Systems*, 9:8–33, 1997.