Pattern-Miner: Integrated Management and Mining over Data Mining Models

Evangelos E Kotsifakos
Department of Informatics, University of Piraeus
80 Karaoli-Dimitriou St
GR-18534 Piraeus, Greece
+302104142437
ek@unipi.gr

Irene Ntoutsi
Department of Informatics, University of Piraeus
80 Karaoli-Dimitriou St
GR-18534 Piraeus, Greece
+302104142437
ntoutsi@unipi.gr

Yannis Vrahortis
Department of Informatics, University of Piraeus
80 Karaoli-Dimitriou St
GR-18534 Piraeus, Greece
+302104142437

Yannis Theodoridis
Department of Informatics, University of Piraeus
80 Karaoli-Dimitriou St
GR-18534 Piraeus, Greece
+302104142437
ytheod@unipi.gr

ABSTRACT
This demo presents Pattern-Miner, an integrated environment for pattern management and mining that deals with the whole lifecycle of patterns, from their generation (using data mining techniques) to their storage and querying, putting also emphasis on the comparison between patterns and meta-mining operations over the extracted patterns. Pattern comparison (comparing results of the data mining process) and meta-mining are high level pattern operations that can be applied in a variety of applications, from database change management to image comparison and retrieval.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Applications – Data Mining
H.3.0 [Information Storage and Retrieval]: General.

General Terms
Design, Algorithms, Management.

Keywords
Pattern management, pattern bases, pattern comparison, pattern monitoring, meta-mining, data mining, pattern representation.

1. INTRODUCTION
Due to the wide application of Knowledge Discovery in Databases (KDD) and as a result of data flood that appears nowadays, the amount of patterns extracted from heterogeneous data sources (e.g. business, science, telecommunications, Web) is huge and, quite often, non-manageable by humans. Thus, there is a need for efficient pattern management including issues like modeling, storage, retrieval and querying of patterns [6]. Pattern management is not an easy task. Except for the huge amount of the generated patterns, another reason is the large variety of pattern types, as a result of the different application needs that each type tries to accomplish.

Traditionally, research work on Data Mining focuses on efficient mining, putting aside the pattern management problem. Recently however, the need for pattern management has been recognized by both scientific and industrial parts and several approaches have been proposed like PMML standard [3] and PMBS approach [5].

In this paper, we demonstrate Pattern-Miner, an integrated environment that deals with the different aspects of the pattern management problem, namely pattern modeling, storage and retrieval issues, using state-of-the-art approaches. This is in contrast to existing tools that deal with specific aspects of the pattern management problem, mostly representation and storage. Although, pattern representation and storage are very important issues, the amount of patterns generated nowadays and the complexity of the different pattern types (clusters, decision trees, frequent itemsets, etc.) call for more sophisticated operations over the extracted patterns, like pattern comparison and meta-mining. Pattern-Miner offers an integrated environment that provides the capability not only to generate and manage the different types of patterns in a unified way, but also to apply more advanced operations over patterns, such as comparison and meta-mining, without facing interoperability or incompatibility problems if using different applications for each task.

Pattern-Miner follows a modular architecture and integrates the different Data Mining components offering transparency to the end user. Before we proceed with the presentation of Pattern-Miner, we provide some basic notions on patterns and pattern bases, following the PBMS approach [5]. These notions comprise the logical model of our approach and the different retrieval capabilities over patterns are built upon them. The pattern concept is the cornerstone of a pattern-base. A pattern is a compact and rich in semantics representation of raw data. Patterns are stored in a so called pattern base for future analysis. The pattern base model consists of three layers: pattern types, patterns, and pattern classes. A pattern type is a description of the pattern structure, e.g. decision trees, association rules, clusters etc. A pattern type is a quintuple pt = (n, ss, ds, ms, f), where n is the name of the pattern type, ss (structure schema) describes the structure of the pattern type (e.g. the head and the body of an association rule), ds (source schema) describes the dataset from which patterns are extracted, ms (measure schema) defines the quality of the source data representation achieved by patterns (e.g. the support and the confidence in case of an association rule pattern) and f is the formula that describes the relationship between the source data space and the pattern space. A pattern is an instance of the corresponding pattern type and a class is a collection of
Berkeley DBXML stores XML documents into logical groups, an extension of the Berkeley DB with the addition of an XML component focused on meta-clustering \cite{2}, i.e. grouping of them, in order to extract meta-patterns \cite{1}, a generic and flexible framework for the comparison of patterns defined over raw data and over other patterns as well. Comparison utilizes both structure and measure components of patterns. The user defines the patterns as well as the way that they should be compared, i.e. how the different components of PANDA are instantiated. The output is a dissimilarity score accompanied with a justification, a report actually of how the component patterns have been matched. In our experiments and for the needs of some real case studies \cite{9} we enhanced the PANDA framework by adding a couple of new cluster comparison algorithms.

**Meta-mining:** Due to the large amount of extracted patterns, several approaches have lately emerged that apply Data Mining techniques over patterns instead of raw data, in order to extract more compact information. The Meta-mining module takes as input a set of different clustering results extracted from the same dataset (through different clustering algorithms or different parameters) or from different datasets (through from the same generative distribution) and applies Data Mining techniques over them, in order to extract meta-patterns. So far, the meta-mining component focuses on meta-clustering \cite{2}, i.e. grouping of clustering results into groups of similar clusterings. The user has full control of the clustering process by choosing the similarity function and the clustering algorithm.

All Pattern-Miner components are developed in Java.

3. DEMO DESCRIPTION

To make clear the potential use and the value of Pattern-Miner, we consider a supermarket as a simple case study and its manager...
as the end user. Among other pattern types, the manager is interested in discovering the products that customers tend to buy together, i.e. association rules. Except for knowing the product associations at each month, the manager also wants to know how these associations change from month to month: are there any new associations, did some old association disappeared, did some association became stronger (higher confidence) or weaker. Also, he/she wants to discover groups of months with similar associations, so as to decide some strategy for each group instead of each month. This process involves storage of the patterns discovered at each month, querying, comparison and meta-mining operations over them. Existing Data Mining tools do not address all these issues. On the contrary, Pattern-Miner provides the manager with all this information in an easy and transparent way. We describe below how each component works for this supermarket scenario.

Pattern extraction and storage: The user defines the data source, the Data Mining algorithm and its parameters, e.g. in our case the supermarket database, the association rule algorithm and the minimum support and confidence parameters. The extraction takes place in the Data Mining engine and the results are converted into PMML format before being stored in a user-specified container in the XML pattern base (as well as in a file on the hard disk). In Figure 2 the pattern extraction and storage screen is depicted for the case of association rule patterns. Using PMML, the exchange of patterns between different applications is possible without the need for special import-export tools.

Pattern query: The user defines the pattern set to be queried and the query itself, in XQuery language. Pattern-Miner engine creates the connection to the pattern base, executes the query and returns the results to the user (and also saves them to a file). A sample query is shown in Figure 3, described in both natural language and XQuery.

Pattern comparison: The user defines the patterns to be compared as well as the comparison parameters. In our example, the manager asks for the comparison of association rule patterns extracted from the supermarket data of the two previous months, in order to inspect whether and how the buying behavior has been changed. The patterns are retrieved from the Pattern-Base. Then, the manager configures PANDA [1] by choosing the appropriate comparison function from the candidate functions implemented for each pattern type. It should be noticed that in the PANDA framework there are several comparison functions implemented, and the user, depending on the application can decide or test what function better fits his/her application. The results are returned to the manager, who can detect any changes in the sales-patterns and decide whether these changes were expected (based on company’s strategy) or not (indicating some suspicious or non-predictable behavior). Based on the results, the manager can decide future strategies regarding offerings, supply etc.

The manager can also extract clusters of customers based on their buying habits or their demographics. Comparing such clusters of customers can reveal buying patterns over the year, and thus the manager can decide about the supplies. In Figure 4 the clustering comparison tab is shown.

Figure 2: The association-rule extraction screen

![Figure 2: The association-rule extraction screen](image.png)

![Pattern-Miner version 2.0 interface](image.png)

![Figure 3: A sample query in natural language and in XQuery](image.png)

<table>
<thead>
<tr>
<th>Query (natural language):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve the association rules from the supermarket dataset that have a support value greater than 0.2.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query (XQuery):</th>
</tr>
</thead>
<tbody>
<tr>
<td>declare namespace a = &quot;<a href="http://www.dmg.org/PMML-3_1">http://www.dmg.org/PMML-3_1</a>&quot;;</td>
</tr>
<tr>
<td>collection (&quot;AssociationRules.dbxml&quot;)</td>
</tr>
<tr>
<td>[dbxml:metadata (&quot;dbxml:dataFileName&quot;)= &quot;C:\Pattern-Miner\data_files\supermarket.arff&quot;]</td>
</tr>
<tr>
<td>/* PMML */</td>
</tr>
<tr>
<td>a:PMML</td>
</tr>
<tr>
<td>a:AssociationModel</td>
</tr>
<tr>
<td>a:AssociationRule [a:support&gt;0.2]</td>
</tr>
</tbody>
</table>

Figure 3: A sample query in natural language and in XQuery

![Figure 4: The clustering comparison tab](image.png)
4. CONCLUSIONS AND OUTLOOK

Pattern-Miner is an integrated environment for pattern management that supports the whole lifecycle of patterns from their generation to their retrieval, and also offers sophisticated operations over patterns, like comparison and meta-mining. Pattern-Miner follows a modular architecture that employs state-of-the-art approaches at each component. The different building blocks are implemented in JAVA.

Several improvements can be carried out: First, the existing components can be enhanced. For example, the querying component could support more query types, like k-nearest neighbor queries, range queries and also the query processing could be more efficient by employing appropriate index structures.

through appropriate indices and new query types could be supported. Also, the Meta-mining module can be extended so as to support more pattern types, like decision trees, association rules, sequences.

Second, new components can be added, like a visualization module for better interpretation of the results or a pattern monitoring module for monitoring and change detection over patterns extracted from a dynamic population.

Except for the scenario we described, other potential applications include cluster-based image retrieval [9], pattern validation, monitoring/ change detection, comparison of patterns extracted from different sites in a distributed environment setting, etc. In this context, we are planning to incorporate to the PANDA framework, some innovative fuzzy clustering comparison techniques we have recently developed.

5. REFERENCES


