AN ENVIRONMENT FOR REPRESENTATION AND MANAGEMENT OF DATA MINING RESULTS AS XML DOCUMENTS

Thereza P. P. Padilha[1 2], Leandro M. Almeida[1] and João B. M. Alves[2]

Abstract: The employment of the eXtensible Markup Language (XML) has been increasing rapidly in several application domains. It has some powerful properties that make it a great candidate for representation and handling of different kinds of data. In this paper we will explore the use of XML documents for representing and management of data mining (DM) results. For this, an implementation of an environment for discovered knowledge management and some experimental results will be presented.

Keywords: data mining, XML, knowledge management.

INTRODUCTION

In recent years, a change in knowledge discovery has occurred, which can be divided into three generations. First, the change was mainly directed towards the development of more powerful data mining algorithms that discover better patterns, achieving higher accuracy, and scaling better on a large data set. The second generation tackled the knowledge discovery life cycle, which includes human resource identification, problem specification, data prospecting, domain knowledge incorporation, methodology identification, data preprocessing, pattern discovery, and knowledge post-processing. The third and current generation is tackling the interchange of discovered knowledge among compliant applications, which requires the specification of a commonly accepted representation. This endeavor has been approached by XML-based languages [1].

The knowledge discovery in databases (KDD) is a complex, iterative, interdisciplinary multi-step process, comprising in patterns investigating. The obtained knowledge of this process should be possible to manipulate it, share it, consolidate it, report it for decision making support, marketing, etc. In this context, this paper presents an environment that uses an XML-based language to represent and manage data mining results (decision tree, for example), gained in the KDD process. The use of XML as representation language allows to build a flexible and extensible environment for processing of discovered knowledge.

THE KDD PROCESS

The knowledge discovery in databases is not only the application of data mining algorithms against the raw data from database. Its definition can change slightly from one author to another, but the general shape means the same, as can be found in [2, 3, 4]. According to Fayyad et al., the KDD process consists of five steps [2]:

- **selection**: understanding of the application domain, creating a dataset, focusing on a subset of attributes (features) or data samples;
- **preprocessing**: data cleaning, deciding on strategies for handling missing data value and removing inconsistent data or noise if necessary;
- **transformation**: data adapting to a particular format of data mining algorithm;

[1] Lutheran University of Brazil – ULBRA - Email: {thereza, leandro}@ulbra-to.br
[2] Federal University of Santa Catarina – UFSC - Email: {thereza, jbosco}@inf.ufsc.br

- **data mining**: searching for relevant patterns and representing in a particular form;
- **interpretation/evaluation**: visualization of the extracted patterns.

Sometimes the KDD process is considered as consisting of only 2-4 steps as presented in [5]. An important issue of this process is extracted knowledge examination by domain experts. The knowledge should be presented in expressive description languages so that they can be easily understood and interactive. In general, the languages used are: trees, tables, rules, graphs, matrices, etc. Each language takes a peculiar format.

Recently, the XML-based approach emerged to support problems of data mining languages, such as platform independence and extensibility. Some works about this have already been reported in the literature. In [6], an XML-based environment is proposed to support part of the KDD process. More specifically, it designed a query language that allows realizing sophisticated combinations of data mining steps. Data mining tasks and their results are specified by means of XML documents. Another previous work proposed an XML framework for the domain of knowledge discovery in databases [7]. Thus, it shows a prototype of the definition of data interfaces for respective KDD steps using XML. The steps were performed by specialized agents and a system of formal ontology would be created to describe the domain of the KDD process.

Although research on XML-based approach to support languages for data mining is still immature, already there is an initiative for standardization of a language, which several participating companies. Some of these languages will be presented later.

XML

The World Wide Web Consortium (W3C) [8] built a language for semi-strutured data named XML, which is designed to allow marking, transferring and reusing information through a standard method of definition of the documents structure and format. XML data are organized into elements delimited by tags, and elements can be nested. It has been recognized by many researches as a promising solution to their problems. The characteristics of XML-based applications are:

- **platform independence**: XML is a document that can be managed by any application, according to Document Type Definition (DTD) that defines elements of the document and a hierarchical order between them;
- **robustness**: XML documents have to be well-formed;
- **extensibility**: from DTDs, XML serves as a metalanguage for definitions of other languages;
- **human legibility**: XML is not directly meant to be read by humans, but simple text editors or specialized editing applications can be used to view, create, and modify it;
- **information interchange**: integration with several repositories and metadata services increase the information interchange by applications.

There are some XML-based description languages have been proposed for data mining results, such as Predictive Model Markup Language (PMML) [9] and XML Data Mining Specification Language (XDMSL) [11], but these still seem to be supported by few tools and research projects. PMML is an XML-based language that provides a quick and easy a way for companies to define predictive

models and share models between compliant vendor's applications [4]. Initially, in version 1.0, it created a small set of DTDs that specify the entities and variables for documenting decision tree and regression models. Actually, in version 2.1, it provides DTDs for clustering, neural networks, association rules, and others. More discussion and further references about PMML can be found in [11]. XDMSL is an XML-based language for the description of each step in the KDD process [10]. The intended use of XDMSL is to capture the whole evolution process from the original data to the knowledge mined from it. In other words, it is used to describe and store all relevant informations to data mining projects, and enable applications to exchange and share such projects.

THE PROPOSED ENVIRONMENT

The proposed environment offers a support for representation and management of knowledge using XML documents. Our choice of using XML as language for data mining results is guided by its characteristics (cited in early section), easiness of the knowledge reusability, and possibility for building queries. The reusability has a vast importance, mainly when it is situated in business applications allowing to manipulate, search and know the identified patterns from databases that contain customer purchase data, for instance. The building queries allows to obtain interesting patterns through combinations among elements, values, features and, mainly, some data mining results, facilitating decision making support.

This environment has been implemented in Java and supports three data mining models. Figure 1 shows a simplified schema of the environment architecture. Graphical User Interface (GUI), DM models and Java API (XalanJ and Document Object Model (DOM)) are main components of the environment. The GUI component allows users to interact with the environment and explore its recourses. This component, basically, handles extracted knowledge. The building a new query is supported by an editor that guides users in the whole process. Presentation of results, setting parameters and exporting results to XML files are also available recourses. In following, other components (DM models and Java API) will be described in more detail.

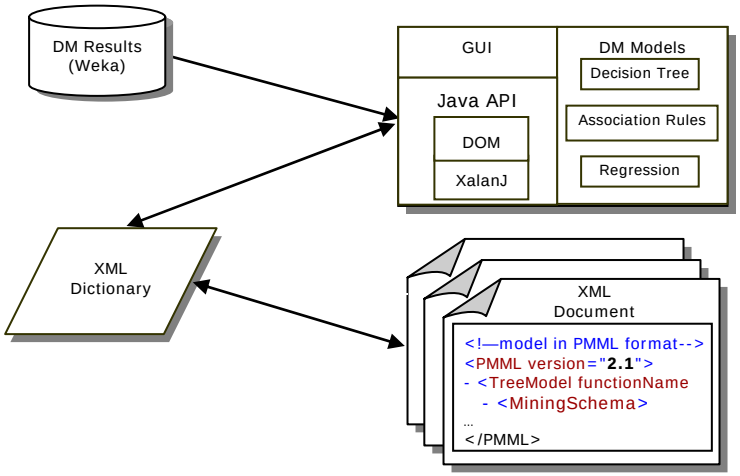

Fig.1.- Environment architecture

The provided data mining results to the environment are based on output format of the weka software. This format consists of identified patterns and several informations such as attributes, number of instances, data source name (relation), learning algorithm and others. Weka is a Java-based collection of machine learning algorithms for data mining tasks [12]. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka has tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU General Public License [12]. The weka was chosen because it is well-known software in Machine Learning community, and, moreover, presents good-quality data mining methods to discover patterns.

DM Models

Every one of the data mining models has a data dictionary, represented by the DataDictionary element, which contains definitions of data fields, specifies their types and ranges of valid values. These definitions are assumed to be independent of specific data sets as used for training or scoring a specific model. Every model contains also one mining schema, represented by the MiningSchema element, which lists fields used in that model. Mining schema is a subset of the fields defined in the data dictionary. While the mining schema contains information that is specific to a certain model, the data dictionary contains data definitions, which do not vary per model. The main purpose of the mining schema is to list the fields that a user has to provide in order to apply the model.

The data mining models used are: a) decision tree (converts weka C4.5, ID3 and Decision Stump decision tree learning model into PMML decision tree learning model), b) association rules (converts weka Apriori Association Rule learning model into PMML association rule model) and c) regression (converts weka M5 numerical prediction learning model into PMML Regression learning model). The decision tree model, for example, consists of the following essential elements:

- **TreeModel**: starts the definition for a tree model;
- **Node**: is an encapsulation for either defining a split or a leaf in a TreeModel. Every Node contains a predicate that identifies a rule for choosing itself or any of its siblings;
- **modelName**: identifies the model with an unique name in PMML file.

The environment receives a data mining result, identifies a mining model through learning algorithm name, analysis according to a data dictionary and provides a correspondent PMML document. One or more mining models can be contained in a PMML document. A PMML document is an XML document with a root element of type PMML.

Java API

The environment uses basically two Java APIs: DOM and XalanJ. For each provided data source, the environment creates an XML dictionary that can aggregate several data mining results. An XML dictionary has pertinent informations of XML documents, such as existing attributes and its values, type of attributes, accuracy level, filename and function name. The main dictionary's functionality is to facilitate the management of stored XML documents.

The knowledge processing from XML documents is made using the XML Path Language (XPath) because it has a similar syntax to navigating through directories. XPath is a language recommended by W3C for addressing parts of an XML document [13]. To support XPath queries is used XalanJ API, developed by Apache Group. XalanJ [14] implements relevant W3C specifications, such as Extensible Stylesheet Language Transformations [15] and XPath [13]. XalanJ basically performs XPath queries and captures query results, transforming them in a new XML document. DOM is a standard recommendation of the W3C for building tree structure in memory for a XML document [16, 17]. In environment, DOM is used for creating, processing, and manipulating XML documents. However, the created tree navigation is made using XPath query because it has flexibility to incorporate logical operators.

The environment supports two kinds of valid and well-formed queries in XML documents: XPath query - the user should know the XPath syntax and documents structure; and SQL query - the user uses SQL syntax and PMML elements. When the user executes a SQL query then it will be transformed to an XPath query. The user can choose a search in specific dictionary that contains links for several other documents, or just one XML document. So, it is possible to find Node elements that contain a specific attribute in XML dictionary, for example. The query output shows the number and description of PMML elements found, i.e., Node. For a visual representation the JTree class is used.

EXPERIMENTAL RESULTS

The aim of this experiment is to show how our environment works for representation and management of data mining results as XML documents. For this, the classic labor data set, available in repository of the weka software, was used to extract weka decision tree learning models (set of data mining results). This data set contains 57 examples and 17 attributes (09 categorical and 08 continuous), grouped in two class.

Figure 2 shows one of these obtained models using the J48 learning algorithm (inductor) and its default parameters. In this model, the generated decision tree has three leaf nodes (each tree level is represented by ' | ' symbol) and its accuracy level is 74%, approximately. Later, we provided to the environment the weka models to transform them in PMML decision tree models and create an XML dictionary, called LaborDictionary.

```
=== Run information ===
Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     labor-neg-data
Instances:    57
Attributes:   17
              wage-increase-first-year
              wage-increase-second-year
              wage-increase-third-year
              statutory-holidays
                           .
                     .
                     .
              vacation
              class
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
------------------
wage-increase-first-year <= 2.5: bad (15.27/2.27)
wage-increase-first-year > 2.5
|   statutory-holidays <= 10: bad (10.77/4.77)
|   statutory-holidays > 10: good (30.96/1.0)

=== Summary ===
Correctly Classified Instances          42        73.6842 %
Incorrectly Classified Instances        15        26.3158 %
```

Fig. 2.- An example of weka decision tree model

The Fig. 3 presents the data mining result shown in Fig. 2 in PMML format. The DataDictionary element summarizes the attributes employed in this model that are: wage-increase-first-year and statutory-holidays. It shows, for each attribute, its name and data type (continuous). The score attribute in a Node element defines the class value. Informations like number of instances, number of attributes and learning algorithm name are not stored because the existent data mining models still not offer elements for this purpose.

An example of a query in the LaborDictionary (LaborDictionary.xml) and its results are shown in Fig. 4. In Query Elements field, up left, the accessible PMML elements and its respective values are presented. In Dictionary Tree field, bottom left, users can see all existing XML dictionaries and the XML documents that are aggregated. The executed query, in Editor field, finds tree nodes (Node elements) that contains class value equal bad.

```
<?xml version="1.0" ?>
- <PMML version="2.1">
    <Header copyright="www.dmg.org" description="Decision Tree of
      labor." />
  - <DataDictionary numberOfFields="2">
      <DataField name="wage-increase-first-year" optype="continuous"
        />
      <DataField name="statutory-holidays" optype="continuous" />
    - <DataField name="class" optype="categorical">
        <Value value="bad" />
        <Value value="good" />
      </DataField>
    </DataDictionary>
  - <TreeModel functionName="classification" modelName="labor">
    - <MiningSchema>
        <MiningField name="wage-increase-first-year" />
        <MiningField name="statutory-holidays" />
        <MiningField name="class" usageType="predicted" />
      </MiningSchema>
    - <Node score="">
        <True />
      - <Node score="bad">
          <SimplePredicate field="wage-increase-first-year"
            operator="lessOrEqual" value="2.5" />
        </Node>
      - <Node score="bad">
        - <CompoundPredicate booleanOperator="and">
            <SimplePredicate field="wage-increase-first-year"
              operator="greaterThan" value="2.5" />
            <SimplePredicate field="statutory-holidays"
              operator="lessOrEqual" value="10" />
          </CompoundPredicate>
        </Node>
      - <Node score="good">
        - <CompoundPredicate booleanOperator="and">
            <SimplePredicate field="wage-increase-first-year"
              operator="greaterThan" value="2.5" />
            <SimplePredicate field="statutory-holidays"
              operator="greaterThan" value="10" />
          </CompoundPredicate>
        </Node>
      </Node>
    </TreeModel>
  </PMML>
```

Fig. 3.- An example of PMML decision tree model

In this case, just ten Node elements were found and listed. In the dotted rectangle, we can observe a Node element that covers instances with wage-increase-first-year <= 2.5. Other informations such as date, file and accuracy are also provided. All the selected Node elements are presented in the same way allowing the user explores the elements for obtain new knowledge for decision making. The user can, for instance, discover a certain pattern in several data mining results, independent of the kind of mining model (tree, association rules and regression).

Although the Fig. 4 has presented a simple query, the employ of syntax similar to SQL facilitates users find patterns because there is strong assimilation with SQL query.

Furthermore, the possibility of reuse of data mining results can provide, for example, a common Node element among several XML documents.

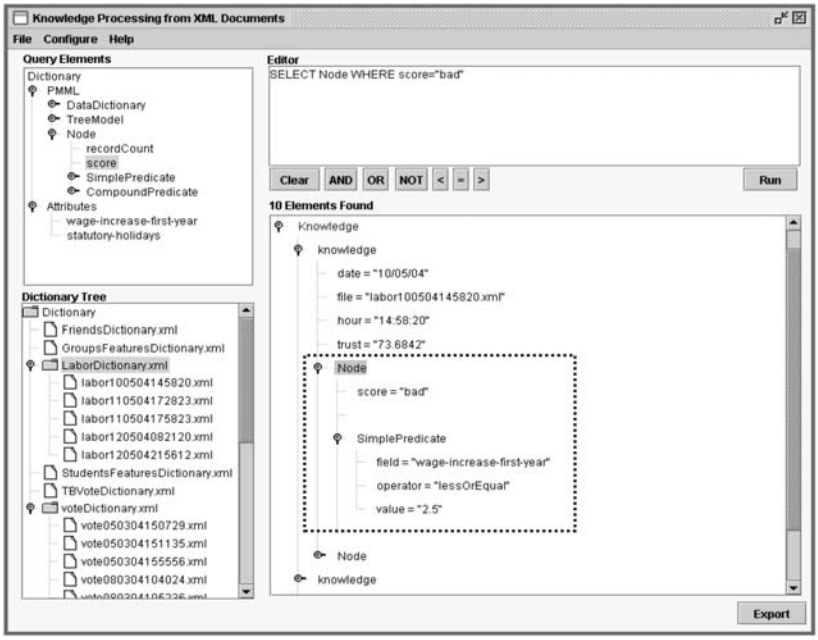

Fig. 4.- An example of query in XML dictionary

CONCLUSIONS AND FURTHER WORK

In general, many communities have used XML to specify the necessary documents format in their applications. In this paper we presented an environment that uses XML-based language for management of obtained data mining results of the weka software. Here, we present just the decision tree data mining model. The main contribution was to show the potential of the XML language for knowledge processing, enhancing the decision making support. We have used XalanJ and XPath to retrieve the contained information in XML documents and JTree class to present visual documents to the user.

The proposed environment is considered open and can be easily extended in multiple directions as the increase of other DM result formats. Some benefits were identified using XML documents for knowledge management, such as: flexibility for accessing Node elements (easy to be parsed), standard model definition supported by PMML and knowledge sharing.

Another advantage of the XML approach is that from XML documents, it is possible and easy to transform one knowledge representation to another, e.g. decision tree to production rules and vice-versa. These documents also can be transformed into other data formats and displayed by different programs using some existing mechanisms such as DOM. So, the users can choice your programs, maximizing the information interchange. In the future, we will intend to add new format of data mining results and explore the integrating into them.

REFERENCES

[1] A. G. Buchner, M. D. Mulvenna, R. Bohm, S. S. Anand; “Data Mining and XML: Current and Future Issues”. In: First International Conference on Web Information Systems Engineering, Hong Kong, 2000.

[2] U. M. Fayyad, G. Piatesky-Shapiro, P. Smyth, R. Uthurusamy: “From Data Mining to Knowledge Discovery in Databases”. American Association for Artificial Intelligence, 1996.

[3] H. Mannila; “Methods and Problems in Data Mining”. In Proceedings of the 7th International Conference on Database Theory (ICDT’97), Delphi, Greece, 1997.

[4] D. Pyle; “Data Preparation for Data Mining”. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1999.

[5] S. O. Rezende; “Sistemas Inteligentes: Fundamentos e Aplicações”. Manole, Barueri, SP, Brazil, 2003.

[6] P. Alcamo, F. Domenichini, F. Turini; “An XML Based Environment in Support of the Overall KDD Process”. In Proceedings of the Fourth International Conference on Flexible Query Answering Systems (FQAS2000), Warsaw, Poland, 2000.

[7] P. Kotàsek, J. Zendulka; “An XML Framework Proposal for Knowledge Discovery in Databases”. In The Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, Lyon, France, 2000.

[8] World Wide Web Consortium Home Page. http://www.w3.org/XML.

[9] Data Mining Group Home Page. http://www.dmg.org/.

[10] P. Kotàsek; “DMSL: The Data Mining Specification Language”. PhD thesis, Brno University of Technology, Faculty of Information Technology, 2003.

[11] Predictive Model Markup Language http://www.dmg.org/pmml-v2-1.html.

[12] I. H. Witten, E. Frank; “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1999.

[13] XML Path Language Home Page. http://www.w3.org/TR/xpath.

[14] Xalan-Java Overview Home Page. http://xml.apache.org/xalan-j/overview.html.

[15] Extensible Stylesheet Language Home Page. http://www.w3.org/Style/XSL/.

[16] Document Object Model Home Page. http://www.w3.org/DOM/.

[17] H. M. Deitel, P. J. Deitel, T. R. Nieto, T. M. Lin, P. Sadhu; “XML: How to Program”. Prentice-Hall, Inc., NJ, USA, 2001.