Toward XML-Based Knowledge Discovery Systems

Rosa Meo
Università degli Studi di Torino
Dipartimento di Informatica
Corso Svizzera 185 - I-10129 Torino, Italy
e-mail: meo@di.unito.it

Giuseppe Psaila
Università degli Studi di Bergamo
Facoltà di Ingegneria
Viale Marconi 5 - I-24044 Dalmine (BG), Italy
e-mail: psaila@unibg.it

Abstract

Inductive databases are intended to be general purpose databases in which both source data and mined patterns can be represented, retrieved and manipulated; however, the heterogeneity of models for mined patterns makes difficult to realize them. In this paper, we explore the feasibility of using XML as the unifying framework for inductive databases, introducing a suitable data model called XDM (XML for Data Mining). XDM is designed to describe source raw data, heterogeneous mined patterns and data mining statements, so that they can be stored inside a unique XML-based inductive database.

1. Introduction

Data mining applications are typically used in the decision making process; in fact, patterns mined from within the source raw data give an intuitive and synthetic representation of phenomena described by the data. However, the Knowledge Discovery Process (KDD process for short) is a typical iterative process, in which not only the raw data can be mined several times, but also the mined patterns might constitute the starting point for further mining on them.

These are the premises that lead Imielinski and Mannila in [3] to propose the idea of inductive database, a general-purpose database in which both the data and the patterns can be represented, retrieved and manipulated. The goal of inductive databases is to assist the deployment of the KDD process. Inside the context of inductive databases the KDD process is driven by the user as a querying sequence. Each query is an instance of a specialized query language, designed for a specific data mining or data analysis problem. Consequently, an inductive database should integrate several heterogeneous data mining and data analysis tools in the same framework.

However, data mining tools involved in the KDD process deal with very different, heterogeneous and complex data models. For example, classification tools usually adopt a data model that is a classification tree [5], while tools for basket analysis usually represent patterns by means of set enumeration models [2].

In this paper, we propose a semi-structured data model specifically designed for inductive databases and, more generally, for Knowledge Discovery Systems. This model is called XDM (XML for Data Mining). It is based on XML and is devised to cope with several distinctive features at the same time. First, it is semi-structured, in order to be able to represent an apriori infinite set of data models. Second, it is based on two simple and clear concepts, named Data Item and Statement: a data item is a container of data and/or patterns; a statement is a description of a data mining operator application. Third, an inductive database modeled with XDM becomes both a collection of data items and a collection of statements, where the knowledge discovery process is represented as a set of relationships between data items and statements.

XDM provides several interesting features for inductive databases. Source raw data and patterns are represented at the same time in the model. The pattern derivation process is stored in the database: this is determinant for the phase of pattern interpretation and allows pattern reuse. Finally, the framework can be easily extended with new data mining operators, exploiting the notion of XML namespaces.

Thus inductive databases based on XDM really become open systems that are easily customizable according to the kind of patterns in which the user/analyst is interested in.

2. XDM Data Items

We now define the first basic element of the XDM data model, i.e. the XDM Data Item.

A Structural Node is a generic node that composes the
tree structure of an XML document or fragment. It can be either an \texttt{ElementNode} (that defines the structure of a tree fragment) or a \texttt{TextNode} (with a textual content). An \texttt{ElementNode} \texttt{n} has a possibly empty set of attributes, i.e. pairs \((\texttt{Name} : \texttt{String}, \texttt{Value} : \texttt{String})\), denoted as \texttt{n.Attributes}.

\textbf{Definition 1:} An \textit{XDM Data Item} is a tree fragment defined in the following way.

- The root of a tree fragment is a \texttt{ElementNode}, here denoted as \texttt{r}, having the following properties: the element tag name, \texttt{r.Name} = "\texttt{DATA-ITEM}"; the element content \texttt{r.Content} is a sequence of \texttt{StructuredNodes}; the prefix associated to the name space, \texttt{r.Prefix} = "\texttt{XDM}"; \texttt{r.NameSpace}, specifying the name space URI, is the standard XDM name space. Hereafter we will refer to the \texttt{ElementNodes} with the notation \texttt{Prefix:Name}.

- An \texttt{ElementNode} has a set of attributes, \texttt{r.Attributes}, denoting the data item information at creation time: Name, Date, Version, Derived (a boolean denoting whether the data item is derived by another data item), and Virtual, (a boolean denoting if the data item is materialized).

\textbf{Example 1:} The start tag

\begin{verbatim}
<XDM:DATA-ITEM Name="Input" Version="1"
 Derived="NO" Date="..." Virtual="NO"
 xmlns:XDM="http://.../NS/XDM">
\end{verbatim}

defines the attributes for the XDM data item named \texttt{Input}. This XDM data item is not derived and not virtual (it is materialized): it might contain the first version of input data to analyze.

In the above tag, notice the namespace definition \texttt{xmlns:XDM="http://.../NS/XDM"}, which says that all element nodes prefixed as \texttt{XDM} belong to the namespace identified by the specified URI.

\textbf{Definition 2:} The \texttt{XDM:DERIVATION} node is an element node defined on the standard XDM name space, here denoted as \texttt{d}. \texttt{d.Content} is empty and \texttt{d.Attributes} contains only one mandatory attribute \texttt{select}, which contains an XPath expression (see [1]) that refers to the XDM \textit{statement} that generated the data item (see Section 3 and Figure 1 for a detailed discussion on derivation).

\textbf{Definition 3:} The \texttt{XDM:CONTENT} node is an element node defined on the standard XDM name space, here denoted as \texttt{c}. The \texttt{XDM:CONTENT} node has no attributes, and only one child \texttt{ElementNode \texttt{n} in c.Content}.

\textbf{Definition 4:} The \texttt{XDM:UPDATE-OF-ITEM} node is an element node defined on the standard XDM name space, here denoted as \texttt{u}. \texttt{u.Content} is empty, and \texttt{u.Attributes} contains only one mandatory attribute \texttt{select}, which contains an XPath expression identifying the sequence of structural nodes that matches the expression, according to XPath semantics (see [1]).

The \textit{Content} of an \textit{ElementNode} must match the following regular expressions: If Derived="NO" \texttt{r.Content} = "\texttt{XDM:UPDATE-OF-ITEM?, XDM:CONTENT}". If Derived = "YES" \texttt{r.Content} = "\texttt{XDM:DERIVATION, XDM:UPDATE-OF-ITEM?}". Where \texttt{DERIVATION}, \texttt{UPDATE-OF-ITEM} and \texttt{CONTENT} are defined in the following.

\textbf{Example 2:} The following XDM code shows the first version of a derived XDM data item, named \texttt{Rules}, containing the association rules extracted from the source data given in input. These data items are shown in the left hand side of Figure 1 that shows a sample KDD process.

\begin{verbatim}
<XDM:DATA-ITEM Name="Rules" Version="1"
 Derived="YES" Date="..." Virtual="NO"
 xmlns:XDM="http://.../NS/XDM">
<XDM:CONTENT>
<AR:ASSOCIATION-RULE-SET
 xmlns:AR="http://...NS/DATA/AssRules">
<AR:ASSOCIATION-RULE-SET
</AR:ASSOCIATION-RULE-SET>
</XDM:CONTENT>
</XDM:DATA-ITEM>
\end{verbatim}

Notice that Derived is "YES" and Virtual is "NO". Therefore this data item will contain the actual data that is obtained by means of the derivation. The \texttt{XDM:DERIVATION} node specifies, with the XPath expression in the \texttt{select} attribute, the statement whose execution generated the item (statements will be described in Section 3). In particular, this item is generated by a statement based on the \texttt{MR:MINE-RULE} operator (see Section 3) which extract association rules from a data item. The generated set of association rules is included in the \texttt{XDM:CONTENT} section (inside \texttt{AR:ASSOCIATION-RULE-SET} tag).

\textbf{Example 3:} The following XDM code fragment shows the initial part of the second version of the XDM data item

\begin{verbatim}

\end{verbatim}

\footnote{We use the XML notation for regular expression operators. In particular, \texttt{"/"} denotes the sequence operator, while \texttt{"?"} denotes the optionality operator.}
named Rules. This is obtained by extracting rules from within an updated version of the source data item (see data items shown in the right hand side of Figure 1). The element node XDM:UPDATE-OF-ITEM shows that this current version is the update of the previous one. It makes use of an XPath expression in the select attribute.

```
<XDM:DATA-ITEM Name="Rules" Version="2" Derived="YES" Date="..." Virtual="NO" xmlns:XDM="http://.../NS/XDM">
  <XDM:DERIVATION select="//XDM:STATEMENT[@ID='00131']"/>
  <XDM:UPDATE-OF-ITEM select="//XDM:DATA-ITEM[@Name='Rules'][@Version='1']"/>
...
</XDM:DATA-ITEM>
```

Notice that the UPDATE-OF-ITEM relationship is extremely useful, to keep trace of changing analysis results. In fact, if the data from which the analysis moves on changes, a cascade effect on derived data items generates a new set of data items which updates the previous ones; for example, changing the data from which association rules are extracted, a new data item with a new set of association rules is generated that updates the old association rule set. The presence of the UPDATE-OF-ITEM relationship may allow to investigate how association rules change in time, simply following the UPDATE-OF-ITEM chain.

3. XDM Statements

The XDM model is devised to capture the KDD process and therefore it provides also the concept of statement. This one specifies the application of an operator (for data manipulation and analysis tasks) whose execution causes the generation of a new, derived data item.

**Definition 5:** An XDM statement $s$ is specified by an XML fragment, whose structure is the following.

- The root of the fragment is an element node XDM:STATEMENT denoted as $s$. $s$ has the attribute ID, which is the statement identifier (with $ID(s)$ we refer to its value).
- The content of each node XDM:STATEMENT has one single child node, describing the application of the operator.
- The operator application must specify (by means of an XPath expression) a non empty set of source XDM data items, which constitute the input for the operator application.
- The operator application must specify one single output XDM data item, which constitutes the output of the operator application. This occurs again by an XPath expression.

**Example 4:** The following example shows the main features of the statement MR:MINE-RULE (see [4] for a complete description on the operator).

```
<XDM:STATEMENT ID="00128" xmlns:MR="http://.../NS/MINE-RULE">
  <MR:MINE-RULE>
    <XDM:SOURCE-ITEM select="http://XDM-DATA-ITEM[@Name='Item'][@Version='1']"/>
    <MR:OUTPUT>
      <XDM:OUTPUT-ITEM Name="Rules" Virtual="NO" Root="MR:RULE-SET" NS="http://.../NS/DATA/Rules"/>
    </MR:OUTPUT>
  </MR:MINE-RULE>
</XDM:STATEMENT>
```

Notice that the operator and its specific element nodes are defined in the namespace prefix MR:, corresponding to the URI "http://.../NS/DATA/Rules", because they are outside the standard XDM namespace. However, XDM:SOURCE-ITEM and XDM:OUTPUT-ITEM nodes, belonging to the XDM namespace appear in the content of the MR:MINE-RULE operator, as specified by the definition of statement. SOURCE-ITEM specifies the input of the operator providing Name and Version number of the XDM data item while OUTPUT-ITEM identifies the output node by means of the Name and the Root node of its tree fragment. Furthermore it specifies the attributes of the output node.

XDM statements consist of the application of one operator. Operators are divided into two categories: Data Manipulation Operators and Data Mining Operators.

Data manipulation operators are useful for managing data items. Given the extensibility features of the XDM framework several (and the most common) operators can be defined on XDM data items, such as LOAD that insert a new data item into the database, DELETE, etc. For the sake of space we do not report them here, although the reader can easily guess that such operators allow the user to generate new data items by modifying or restructuring other data items.

4. XDM Database State

**Definition 6:** The state of an XDM database is represented as a pair

$$\langle D1 : Set Of(DataItem), ST : Set Of(Statement) \rangle,$$
where $DI$ is a set of XDM data items (see Definition 1), and $ST$ is a set of XDM statements (see Definition 5). The following constraints hold.

- **Data Item Identity.** Given a data item $d$ and its mandatory attributes Name, and Version, the pair (Name, Version) uniquely identifies the data item $d$ in the database state.

- **Statement Identity.** Given a statement $s$ and its mandatory attribute ID, its value uniquely identifies the statement $s$ in the database state.

- **Relationship between statements and source data items.** Consider an XDM statement $s$. The XPath expression specified by the select attribute of each XDM:SOURCE-ITEM appearing in $s$ must find one and only one XDM data item.

- **Relationship between derived data items and statements.** Consider a derived XDM data item $d$ ($Derived(d) = "YES"$). The XPath expression specified by the select attribute of the XDM:DERIVATION element must find one and only one XDM data item.

- **Update-of-item Relationship.** Consider an XDM data item $d$ such that the XDM:UPDATE-OF-ITEM $u$ appears in its content ($u.Name = "UPDATE-OF-ITEM", u.Prefix = "XDM" and $u \in d.Content$). The value $s$ of the select attribute must match one and only one XDM data item.

**Observations.** The XDM database is then both a data item base and a statement base. When a new statement is executed, the new database state is obtained from the former one by adding both the executed statement and the new data item.

This structure represents the two-fold nature of the knowledge discovery process: data and patterns are not meaningful if considered in isolation; in contrast, patterns are significant if the overall process is described, because the meaning of data items is clarified by the clauses specified in the data mining operators that generate data items.

### 5. Conclusions

In this paper we presented a new, XML-based data model, named XDM. It is designed to be adopted inside the framework of inductive databases.

XDM allows the management of semi-structured and complex patterns thanks to the semi-structured nature of the data that can be represented by XML.

In XDM the pattern definition is represented together with data. This allows the reuse of patterns by the inductive DBMS. In particular, XDM explicitly represents the statements that were executed in the derivation process of the pattern. The flexibility of the XDM representation allows extensibility to new pattern models and new mining operators: this makes the framework suitable to build an open system, easily customized by the analyst.

One drawback of using XML in data mining, however, could be the large volumes reached by the source data represented in XML (due to the addition of mark-up tags and attributes).

### References


