Discovering Interesting Information in XML Data with Association Rules

Daniele Braga*, Alessandro Campi*, Stefano Ceri*,
Mika Klemettinen†, PierLuca Lanzi*
*Politecnico di Milano, P.za L. da Vinci 32, I-20133, Milano, Italy
†Nokia Research Center, FIN-00045 Nokia Group, P.O.Box 407, Finland
{braga,campi,ceri,lanzi}@elet.polimi.it, Mika.Klemettinen@nokia.com

ABSTRACT

Data mining algorithms are designed to extract *interesting* information from large amounts of data. They usually assume that source data are in relational (tabular) form. However, the recent success of XML as a standard to represent semi-structured data and the increasing amount of data available in XML pose new challenges to the data mining community. In this paper we introduce association rules for XML data. To accomplish this, we propose a new operator, based on XPath and inspired by the syntax of XQuery, which allows us to express complex mining tasks, compactly and intuitively. The operator can indifferently (and simultaneously) target both the content and the structure of the data, since the distinction in XML is slight.

1. INTRODUCTION

Knowledge discovery in databases (KDD) deals with the process of extracting *interesting* knowledge from large amounts of data, usually stored in large databases or data warehouses. Knowledge can be represented in many different ways, such as clusters, decision trees, decision rules, etc.; in particular, association rules [2, 3] have proved to be an effective tool to discover interesting relations in massive amounts of data.

During the recent years, we have seen the dramatic development of the eXtensible Markup Language. However, it is easy to foresee that the spreading of XML will cause an increasing interest on this subject, going beyond a mere syntactic adaptation to XML of data mining artifacts and techniques.

We introduce association rules from XML documents. This extension is associated to nontrivial problems, related to the hierarchical nature of the XML data model. Consequently, most of the common and well-known abstractions of the relational framework need to be adapted to and redefined. The proposed extension is formalized by introducing an XML-specific operator for describing association rules, called `XMINE RULE`, which is based on the use of XML query languages.

SAC 2003 Melbourne, Florida, USA

2. ASSOCIATION RULES

Association rules are implications of the form $X \Rightarrow Y$ where the rule *body* X and the rule *head* Y are subsets of the set $\mathcal{I}$ of *items* ($\mathcal{I} = \{I_1, \ldots, I_n\}$) within a set of *transactions* $\mathcal{D}$. The rule $X \Rightarrow Y$ states that the transactions $T \in \mathcal{D}$ which contain the items in X ($X \subset T$) are *likely* to contain also the items in Y ($Y \subset T$). Association rules are characterized by two measures: the *support*, which measures the percentage of transactions that contain both items X and Y; the *confidence*, which measures the percentage of transactions that contain the items Y within the transactions that also contain the items in X. More formally, given the function $freq(X, \mathcal{D})$, denoting the percentage of transactions in $\mathcal{D}$ which contains X, we define:

$$support(X \Rightarrow Y) = freq(X \cup Y, \mathcal{D}) \quad (1)$$

$$confidence(X \Rightarrow Y) = freq(X \cup Y, \mathcal{D}) / freq(X, \mathcal{D}) \quad (2)$$

The problem of mining association rules consists of finding association rules, from a set of transactions $\mathcal{D}$, that have support and confidence greater than the minimum support (*minsupp*) and the minimum confidence (*minconf*).

To define association rules for XML documents, we have to map the basic concepts of association rules in the XML context. In particular, we must define what is a *transaction* T to define *support* and *confidence* within an XML document.

As a working example, we introduce the XML document depicted in Figure 1.

We consider the problem of mining frequent associations among people that appear as coauthors in the publications, i.e. associations of the form:

"$Wilson \Rightarrow Holmes$"

Since any part of an XML document is a tree (XML *fragment*), both $\mathcal{D}$ and $\mathcal{I}$ are sets of trees. In particular, the transactions $T \in \mathcal{D}$ are the XML fragments that define the *context* in which the items $I \in \mathcal{I}$ must be counted. The *items* $I \in \mathcal{I}$ are the fragments that must be counted.

If we consider the problem of mining association rules among authors appearing in the same papers, we have that $\mathcal{D}$ are the set of *fragments* of the publications. The items $I \in \mathcal{I}$ are the set of *fragments* of all the authors who appear in the works published by people within the depart-

```
<DEPARTMENT>
 <PhDCourses>
  <Course teacher="fp1" title="Advanced Data Mining">
   <TimeTable>...</TimeTable>
   <Student ref="ps1" /><Student ref="ps2" />
  </Course>
  <Course teacher="fp3" title="Intricacies of XML parsers">
   <TimeTable>...</TimeTable>
   <Student ref="ps2" /><Student ref="ps3" />
  </Course>
 </PhDCourses>
 <People>
  <PhDStudent id="ps2" advisor="fp3">
   <PersonalInfo email="fp3@cs.atlantis.edu">
    <Name>...</Name>
   </PersonalInfo>
   <Subscription year="2001" />
   <Publications> ... </Publications>
  </PhDStudent>
  <FullProfessor id="fp3">
   <PersonalInfo email="fp3@cs.atlantis.edu">
    <Name> ... </Name>
   </PersonalInfo>
   <Publications>
    <Article title="Golden Data Mines in Atlantis">
     <Author>...</Author>
     <Conference name="VLDB" year="2001" />
    </Article>
    <Article title="P is just like NP - The Final Proof">
     <Author>...</Author>
     <Journal year="2000" month="4" volume="4"
              name="DMKD" publisher="Kluwer" />
    </Article>
    <Book year="2001" title="XML Query Languages">
     <Author>...</Author>
     <Publisher>...</Publisher>
     <Keyword>XML</Keyword>...<Keyword>XQuery</Keyword>
    </Book>
   </Publications>
   <Award year="2001" society="IEEE">This award..</Award>
  </FullProfessor>
 </People>
</DEPARTMENT>
```

Figure 1: `http://www.cs.atlantis.edu/research.xml`, a sample document with various information about the research activities of a university department.

ment. This situation is depicted in Figure 2 where an example of the `research.xml` document is represented. The overall document is represented by the grey triangle. The *white triangles* represent the *fragments* corresponding to the various publications authored by people of the department. The *black triangles* represent the *fragments* corresponding to the authors who appear in various publications. Note that, the *value* of the `<Author>` tag is depicted at the bottom of the black triangle, thus a black triangle labeled `Wilson` is equivalent to the XML fragment `<Author>Wilson</Author>`. While the *value* of a white triangle, corresponding to a `<Book>` tag or to a `<Article>` tags, represents an entire XML fragment which corresponds to a book or to an article, e.g.:

```
<Article title="Classifier Fitness Based on Accuracy">
 <Author>Wilson</Author>
 <Journal year="1995" month="4" volume="4"
          name="Evolutionary Computation"
          publisher="MIT Press" />
</Article>
```

In the example in Figure 2, $\mathcal{D}$ contains the four white triangles and $\mathcal{I}$ contains the the three black triangles corresponding to the authors *Holmes*, *Stolzmann*, and *Wilson*. While an itemset $X \subset \mathcal{I}$ is a set containing black triangles. It is now possible to compute $freq(X, \mathcal{D})$ as the percentage of XML fragments in $\mathcal{D}$ which contains the fragments in X. In particular, $freq(X, \mathcal{D})$ is computed as the percentage of white triangles which contain all the *black triangles* in X. Thus, in the example depicted in Figure 2, $freq(\{$`<Author>Wilson</Author>`,`<Author>Holmes</Author>`$\}, \mathcal{D})$ is 0.5 since the fragments `<Author>Wilson</Author>` and `<Author>Holmes<Author>` appear together in two white triangles out of four, i.e., in 50% of the cases. Likewise, $freq(\{$`<Author>Wilson</Author>`$\}, \mathcal{D})$ is 0.75 since the fragment `<Author>Wilson</Author>` appears in three white triangles out of four, i.e., in 75% of the cases. Therefore we can now compute the *confidence* of the (XML) association rule:

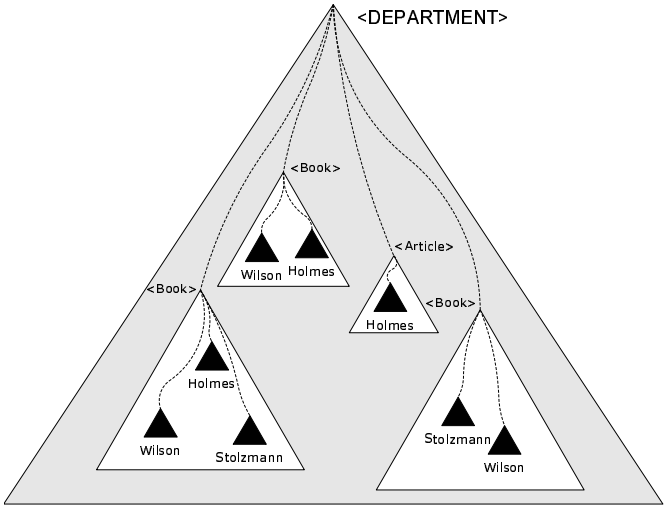

Figure 2: A graphical representation of the document `research.xml` described in Figure 1.

$$\texttt{<Author>Wilson</Author>} \Rightarrow \texttt{<Author>Holmes</Author>}$$

as:

$$\frac{freq(\{\texttt{<Author>Wilson</Author>}, \texttt{<Author>Holmes</Author>}\}, \mathcal{D})}{freq(\texttt{<Author>Wilson</Author>}\}, \mathcal{D})}$$

which returns 0.66, i.e., in 66% of papers published by *Wilson* appears also *Holmes* or, in the XML context, in 66% of the fragments in $\mathcal{D}$ in which appears the fragment `<Author>Wilson</Author>` appears also the fragment `<Author>Holmes</Author>`. While the *support* of the association rule is computed as:

$$freq(\{\texttt{<Author>Wilson</Author>},\texttt{<Author>Holmes</Author>}\}, \mathcal{D})$$

which returns 0.5 since the fragments `<Author>Wilson</Author>` and `<Author>Holmes</Author>` appear together in two white triangles out of four, i.e., in 50% of the cases.

3. AN OPERATOR FOR XML MINING

To specify association rule mining tasks in XML documents we introduce the `XMINE RULE` operator. The operator is based on XPath, inspired to the syntax of XQuery, and to the work of [7] in the context of relational databases.

As a first example, we consider the problem formerly discussed in Section 2 which consists of mining frequent associations among people who appear as coauthors.

```
XMINE RULE
IN document("www.cs.atlantis.edu/research.xml")
FOR ROOT IN //People/*/Publications/*
LET BODY := ROOT/Author, HEAD := ROOT/Author
EXTRACTING RULES WITH
     SUPPORT = 0.1 AND CONFIDENCE = 0.2
RETURN
<RULE support={ SUPPORT }
      confidence={ CONFIDENCE }>
 <BODY> { FOR $item IN BODY
          RETURN <Item> { $Item } <Item> } </BODY>
 <HEAD> { FOR $item IN HEAD
          RETURN <Item> { $Item } <Item> } </HEAD>
</RULE>
```

The XMINE RULE statement begins with the IN clause which specifies the data source, i.e., the document `http://www.cs.atlantis.com/research.xml`.

In the FOR section, the special variable ROOT specifies the fragments which represent the transactions $T \in \mathcal{D}$ by means of one (or more) XPath expressions. In the example in Figure 2, ROOT identifies the white triangles.

The next two special variables, BODY and HEAD, identify the fragments in the source data which should appear respectively in the rule body and in the rule head, i.e., the set of items $I \in \mathcal{I}$. Note that both BODY and HEAD *are* defined with respect to the *context* (i.e., to the ROOT variable) since confidence and support values have meaning only with respect to the *context* defined by the ROOT (i.e., the set of transactions $\mathcal{D}$). More specifically, BODY and HEAD are defined as Author elements appearing in some Publications. Since the Author element appears both in articles and in books, Author is specified as a subelement of Publications/*.

The EXTRACTING RULES WITH clause specifies the minimum support and the minimum confidence values for the output association rules with the clauses, "SUPPORT=0.1" and "CONFIDENCE=0.2". The final RETURN clause specifies the structure of the association rules produced. An association rule is defined by a RULE tag, with two attributes, support and confidence which specify the support and confidence values; and two subelements, <BODY> and <HEAD>, which specify the items in the rule body and in the rule head. Some of the rules produced by the above statement on the data depicted in Figure 1 are the following:

```
<RULE support="0.25" confidence="0.33">
 <BODY><Item><Author>Holmes</Author></Item></BODY>
 <HEAD><Item><Author>Stolzmann</Author></Item></HEAD>
</RULE>
 ...
<RULE support="0.25" confidence="0.50">
 <BODY><Item><Author>Stolzmann</Author></Item>
       <Item><Author>Wilson</Author></Item></BODY>
  <HEAD><Item><Author>Holmes</Author></Item></HEAD>
</RULE>
```

Where, the latter RULE fragment represents the rule with <Author>Stolzmann</Author> and <Author>Wilson</Author> as body, and <Author>Holmes</Author> as head; the rule has support 0.25 and confidence 0.5.

In the the remaining of the paper we will omit the RETURN clause.

3.1 Syntax

The XMINE RULE statement consists of six clauses.

```
XMINE RULE
IN Doc-Name (, Doc-Name)*
FOR ROOT IN XPathExpr (, FOR Variable IN XPathExpr)*
LET BODY := (XPathExpr,)+
     HEAD := XPathExpr (,XPathExpr)*
     (, Variable := XPathExpr)*
[WHERE XQuery-Where-Clause]
[GROUP Variable IN Target BY Criteria ]
EXTRACTING RULES WITH
   SUPPORT = <real> AND CONFIDENCE = <real>
   [AND XQuery-Where-Clause]
[RETURN XQuery-Return-Clause]
```

The IN clause specifies the source documents. The FOR/LET clauses define the *mandatory* ROOT, HEAD and BODY variables. Note that additional *Variable* might be added. All variables are bound to fragment sets in terms of XPath expressions. The optional WHERE clause expresses complex conditions to filter the fragments. The optional GROUP clause allows the restructuring of the source data. The EXTRACTING RULES WITH clause specifies a set of constraints on the generated rules. In particular the mandatory SUPPORT and CONFIDENCE clauses specify the minimum support and the minimum confidence; additional constraints can be expressed with an *XQuery-Where-Clause*. Finally, the optional RETURN clause defines the structure of the output rules. It is expressed according to the syntax of the node constructors of XQuery.

3.2 Semantics

We now present an intuitive semantics of the XMINE RULE operator described in Section 3. Let X_R be the set of XPath expressions specified in the ROOT section; X_B that specified in the BODY section; X_H that in the HEAD section; $X_\mathcal{I}$ be $X_B \cup X_H$. We consider the simple case in which the BODY and the HEAD sections have identical sets of XPath expressions, $X_\mathcal{I} = X_B = X_H$, as in the example in Section 3.

Let *value* be a function that, given an XPath expression p and an XML document d returns the set of XML fragments in d identified by p. Let *satisfy* be a function that, given an XML fragment f and an XQuery where clause w, returns one if f satisfies w, zero otherwise. Let *contains* be a function that, given an XML fragment x and an XML fragment y, returns one if x contains y, zero otherwise. First, we compute the set $F_\mathcal{D}$ as the collection of all the XML fragments defined by the XPath expressions in X_R, more formally:

$$F_\mathcal{D} = \bigcup_{p \in X_R} value(p, d)$$

Likewise, we compute the set $F_\mathcal{I}$ as the collection of XML fragments that are defined by the XPath expressions in $X_\mathcal{I}$.

$$F_\mathcal{I} = \bigcup_{p \in X_\mathcal{I}} value(p, d)$$

We now define $F'_\mathcal{D}$ and $F'_\mathcal{I}$ by filtering the sets $F_\mathcal{D}$ and $F_\mathcal{I}$ through the XQuery where clause w specified in the WHERE section. $F'_\mathcal{D}$ and $F'_\mathcal{I}$ are defined as follows:

$$\begin{aligned} F'_\mathcal{D} &= \{f \mid f \in F_\mathcal{D} \wedge satisfy(f, w)\} \\ F'_\mathcal{I} &= \{f \mid f \in F_\mathcal{I} \wedge satisfy(f, w)\} \end{aligned}$$

From the definitions of $F'_\mathcal{D}$ and $F'_\mathcal{I}$ it is now possible to compute the *frequency* of an itemset X, $X \in F'_\mathcal{I}$, as follows:

$$freq(X, F'_\mathcal{D}) = \frac{\sum_{c \in F'_\mathcal{D}} \prod_{f \in X} contains(c, f)}{|F'_\mathcal{D}|}$$

Given the function *freq* it is then possible to extract frequent (XML) itemsets from an XML document. Therefore it is possible to extract the association rules, as specified in the XMINE RULE statement.

4. EXPRESSIVE POWER THROUGH EXAMPLES

In this section, we progressively demonstrate the various features of the `XMINE RULE` operator by means of examples.

4.1 Collaborations

We begin with a new version of the example introduced in section 3, imposing the condition that at least one of the authors in the body of the generated rules is a member of the department.

```
XMINE RULE
IN document("www.cs.atlantis.edu/research.xml")
FOR ROOT IN //People/*/Publications/*
LET BODY := ROOT/Author, HEAD := ROOT/Author
EXTRACTING RULES WITH
 SUPPORT = 0.1 AND CONFIDENCE = 0.2 AND
 SOME $a IN BODY SATISFIES
  not(empty(//People/*/PersonalInfo/Name[=$a]))
```

We can further develop this example in order to show a filtering condition, restricting the publications to be considered to those published in 2001.

```
XMINE RULE
IN document("www.cs.atlantis.edu/research.xml")
FOR ROOT IN //People/*/Publications/*
LET BODY := ROOT/Author, HEAD := ROOT/Author
WHERE ROOT//@year = 2001
EXTRACTING RULES WITH
 SUPPORT = 0.1 AND CONFIDENCE = 0.2 AND
 SOME $a IN BODY SATISFIES
  not(empty(//People/*/PersonalInfo/Name[=$a]))
```

This clause has obviously a side effect on the generated rules, since all the publications that do not satisfy the condition are pruned. Therefore, the resulting context is a subset of the previous one, and both support and confidence take different values.

4.2 Related topics

The next example shows two rules which differ for the choice of context; the two examples enable a discussion of the meaning of rules. In either case, the mining task aims at discovering frequent associations between research interests, represented by means of keywords.

In this example, the rule searches for associations among keywords related to people of the departments, and therefore extract research fields which are typically addressed by the same people. The head and the body of the generated rules is the `Keyword` element in publications, and the context is the set of all the department members (`PhDStudent` and `FullProfessor` elements), denoted as direct subelements of the `People` element (by means of the `*` XPath step).

```
XMINE RULE
IN document("www.cs.atlantis.edu/research.xml")
FOR ROOT IN //People/*
LET BODY := ROOT/Publications//Keyword,
    HEAD := ROOT/Publications//Keyword
EXTRACTING RULES WITH
    SUPPORT = 0.1 AND CONFIDENCE = 0.2
```

This rule mines the intrinsic "closeness" of research topics — as determined because the topics appear in the same research paper — and not the related interests of the researchers — as determined because two topics are of interest to the same researcher. The second rule is expressed substituting in the previous example the following clauses:

```
FOR ROOT IN /People/*/Publications/*
LET BODY := ROOT/Keyword, HEAD := ROOT/Keyword
```

4.3 Credentials

The next example shows the use of several path expressions. We want to discover the associations between publications and awards, looking for which type of publication is mostly related to which kind of award. Since the publishers are represented as attributes in `Journal` elements and as PCDATA content in Books, while the conference types are stored as `name` attributes, the `BODY` fragment set is the *set union* of the nodes which three different path expressions evaluate to.

```
XMINE RULE
IN document("www.cs.atlantis.edu/research.xml")
FOR ROOT IN /People/*
LET BODY := ROOT/Publications/Book/Publisher,
  ROOT/Publications/Article/Journal/@publisher,
  ROOT/Publications/Article/Conference/@name,
    HEAD := ROOT/Award/@society,
EXTRACTING RULES WITH
    SUPPORT = 0.1 AND CONFIDENCE = 0.2
```

5. IMPLEMENTATION NOTES

We implemented a prototype of the `XMINE RULE` operator as a three-step process. In the initial step, **preprocessing**, the `XMINE RULE` statement is processed in order to generate a representation of the mining problem as a binary relation R. Then in the following step, **mining**, binary association rules are extracted from R. In the final step, **post-processing**, the binary association rules extracted from R are mapped into the XML representation.

Preprocessing. First the sets $F_{\mathcal{D}}$ and $F_{\mathcal{I}}$ (see Section 3.2) are generated. These sets contain the XML fragments addressed by the XPath expressions specified in the clauses `ROOT` ($F_{\mathcal{D}}$), `HEAD`, and `BODY` ($F_{\mathcal{I}}$). Then the sets $F'_{\mathcal{D}}$ and $F'_{\mathcal{I}}$ containing the XML fragments in $F_{\mathcal{D}}$ and $F_{\mathcal{I}}$ which satisfy the `WHERE` clause are generated. This step is implemented on the top of the Xalan interpreter of Xpath. Note that since there is no XQuery interpreter currently available, our initial implementation supports only basic `WHERE` conditions mainly based on XPath expressions such as: `ROOT//@year=2001`. With the sets $F'_{\mathcal{D}}$ and $F'_{\mathcal{I}}$ the relational table R is generated as follows.

- R has as many attributes (i.e., columns) as is the number of XML fragments in $F'_{\mathcal{I}}$, i.e., $R \subset \{0,1\}^{|F_{\mathcal{I}}|}$.
- R has as many tuples (i.e., rows) as the number of fragments in $F'_{\mathcal{D}}$, i.e., $R = \{r_1, \ldots, r_{|F'_{\mathcal{D}}|}\}$;
- for every XML fragment $c_i \in F'_{\mathcal{D}}$ and every XML fragment $f_j \in F'_{\mathcal{I}}$, a tuple $r_i \in R$, $r_i = \langle r_{i,1} \ldots r_{i,|F_{\mathcal{I}}|} \rangle$, is defined as $r_{i,j} = \mathit{contains}(c_i, f_j)$.

From Section 3.2 we recall that the function $\mathit{contains}(c_i, f_j)$ returns one if the fragment c_i contains f_j. Note that the current implementations of XPath do not provide any method to check whether a fragment appear in another fragment. Accordingly, in our prototype the function *contains* is implemented on the top of the XPath interpreter as follows. Given two fragments c_i and f_j, their textual representation is generated exploiting the functionalities of the XPath interpreter. Then it is checked whether the textual representation of f_j actually appear in the textual representation of c_i; if it does, one is returned, otherwise zero. This solution is indeed computationally expensive, but on the other hand, the comparison of XML fragments is one of the major open

issues in the XML community. Thus, it might be possible in the future to reduce the complexity of this phase through some advanced technique introduced in the next versions of XPath and XQuery.

Mining. With the table R and the constraints specified in the `EXTRACTING RULE` clause, binary association rules are extracted by using the Apriori algorithm. Note that among the many constraints which might be specified, our implementation currently considers only constraints concerning the composition of the rules (e.g., the number of items in the rule head or body), the minimum support, and the minimum confidence. In fact, complex constraints (e.g., those described in Section 4) are currently not possible with a tabular representation — they would require mining the XML association rules directly from the DOM (Document Object Model) representation. DOM offers a platform- and language-neutral API (Application Programming Interface) allowing programs to dynamically access and update the content, structure and style of the documents.

Postprocessing. In the final step, the boolean association rules are remapped into the XML representation. Here the information which have been memorized during the preprocessing, such as the correspondence between the XML fragments in $F'_{\mathcal{D}}$ and $F'_{\mathcal{I}}$ and the boolean attributes in R, is exploited in order to print an XML version of the association rules generated through the Apriori algorithm.

Efficiency has not been of our main concern in preparing the initial implementation; however, the first experiments of use are good thanks to the excellent performance of the Xalan XPath interpreter. In addition, our initial experiments suggests that the preprocessing and postprocessing steps are reasonably fast with respect to the central mining phase. These considerations suggest that an efficient execution of `XMINE RULE` statements is feasible with current technology as long as the statement includes only XPath expressions. Our three-step process, which exploits an intermediate relational representation, allowed us to implement a basic prototype smoothly. On the other hand, the lack of a robust XQuery execution environment does not allow us to compute the complex predicates supported by the `XMINE RULE` syntax on a native XML representation.

The initial implementation described above leaves many open issues, which must be addressed to have a fully functional and efficient implementation of the `XMINE RULE` operator. We believe that the most important open issue regards the support evaluation, which should be used to extract association rules from XML documents. There are currently no XQuery interpreters to be used in filtering XML fragments and although XPath processors are quite advanced, they do not yet provide all the required functionalities either. Many of these functionalities should be included in the new XPath 2.0 version, while some operations are still not defined but are still requirements of the W3C consortium. Therefore, many required computation must be coded directly on the raw DOM representation of the documents (e.g., the filtering of `WHERE` clauses) which causes some computational overhead.

6. RELATED WORK

Since the introduction of *association rules* in [2], the area has attracted the research community. First, the Apriori algorithm [4, 5] for extracting association rules was developed. Then, a number of algorithms for the extraction of association rules from multivariate data have been proposed.

Because a large part of the data available in the very next future will be represented in XML, a number of proposals to exploit XML within Data Mining have been proposed. These proposals regard both the use of XML as a format to represent the knowledge extracted from the data and the development of techniques and tools for mining data expressed in XML documents. However, as far as we know, a well-formulated framework for discovering association rules within the context of *native* XML documents has not been introduced so far.

7. CONCLUSIONS

In this paper, we defined association rules from native XML data by means of examples using a new `XMINE RULE` operator based on and inspired by `MINE RULE` [7] and XQuery. We also discussed the syntax and semantics behind the `XMINE RULE` operator, and presented concrete steps for implementation.

Although this research provides a good starting point, there are still many open issues for future research — even within the work presented in this paper. Additionally, association rules from XML data could be extended to the case of episode rules [6].

Acknowledgements

In this work, authors have been supported by the *consortium on discovering knowledge with Inductive Queries* (*cInQ*) [1], project funded by the European Commission under the "Information Society Technologies Programme (1998-2002)" "Future and Emerging Technologies" arm. Daniele Braga and Alessandro Campi are supported by the "Virtual Campus" Microsoft Research Grant.

8. REFERENCES

[1] **c**onsortium on discovering knowledge with **In**ductive **Q**ueries (*cInQ*). `http://www.cinq-project.org`.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216. ACM, 1993.

[3] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, 1996.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*, pages 3–14. IEEE, 1995.

[5] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181 – 192. AAAI Press, 1994.

[6] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovering frequent episodes in sequences. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210 – 215. AAAI Press, 1995.

[7] Rosa Meo, Giuseppe Psaila, and Stefano Ceri. An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195 – 224, 1998.