Mining Association Rules from XML Data

Daniele Braga[1], Alessandro Campi[1], Mika Klemettinen[2], and PierLuca Lanzi[3]

[1] Politecnico di Milano, Dipartimento di Elettronica e Informazione,
P.za L. da Vinci 32,I-20133 Milano, Italy
{braga,campi}@elet.polimi.it
[2] Nokia Research Center, P.O.Box 407, FIN-00045 Nokia Group, Finland
mika.klemettinen@nokia.com
[3] Artificial Intelligence and Robotic Laboratory
Politecnico di Milano, Dipartimento di Elettronica e Informazione
pierluca.lanzi@polimi.it

Abstract. The eXtensible Markup Language (XML) rapidly emerged as a standard for representing and exchanging information. The fast-growing amount of available XML data sets a pressing need for languages and tools to manage collections of XML documents, as well as to *mine interesting information* out of them. Although the data mining community has not yet rushed into the use of XML, there have been some proposals to exploit XML. However, in practice these proposals mainly rely on more or less traditional relational databases with an XML interface. In this paper, we introduce association rules from native XML documents and discuss the new challenges and opportunities that this topic sets to the data mining community. More specifically, we introduce an *extension of XQuery* for mining association rules. This extension is used throughout the paper to better define association rule mining within XML and to emphasize its implications in the XML context.

1 Introduction

Knowledge discovery in databases (KDD) deals with the extraction of *interesting* knowledge from large amounts of data usually stored in databases or data warehouses. This knowledge can be represented in many different ways such as clusters, decision trees, decision rules, etc. Among the others, association rules [4], proved effective in discovering interesting relations in massive amounts of data.

During the recent years, we have seen the dramatic development of the eXtensible Markup Language (XML) [19] as a standard for representing and exchanging information. Accordingly, there is a pressing need for languages and tools to manage collections of XML documents and to *extract interesting knowledge* from XML documents. At the moment, the use of XML within the data mining community is still quite limited. There are some proposals to exploit XML within the knowledge discovery tasks, but most of them still rely on the traditional relational framework with an XML interface. However, the pressure for data mining tools for native XML data is rapidly increasing.

In this paper, we introduce association rules from *native* XML documents. This extension arises 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 in the specific XML context.

2 Association Rules

Association rules (ARs) were first introduced in [3] to analyze customer habits in retail databases; up-to-date definitions can be found in [8]. An association rule is an implication of the form $X \Rightarrow Y$, where the rule *body* X and *head* Y are subsets of the set $\mathcal{I}$ of *items* ($\mathcal{I} = \{I_1, \ldots, I_n\}$) within a set of *transactions* $\mathcal{D}$. A rule $X \Rightarrow Y$ states that the transactions T ($T \in \mathcal{D}$) that contain the items in X ($X \subset T$) are *likely* to contain also the items in Y ($Y \subset T$). ARs are usually characterized by two statistical measures: *support*, which measures the percentage of transactions that contain the items in both X and Y, and *confidence*, which measures the percentage of transactions that contain the items in X and also contain the items in Y. More formally, given the function $\mathit{freq}(X, \mathcal{D})$, denoting the percentage of transactions in $\mathcal{D}$ which contains X, we define:

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

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

Suppose there is an AR "*bread,butter* $\Rightarrow$ *milk*" with confidence 0.9 and support 0.05. The rule states that customers who buy *bread* and *butter*, also buy *milk* in 90% of the cases and that this holds in 5% of the transactions. The problem of mining ARs from a set of transactions $\mathcal{D}$ consists of generating all the ARs that have support and confidence greater than two user-defined thresholds: minimum support (*minsupp*) and minimum confidence (*minconf*).

To help the specification of complex AR mining tasks, a number of query languages have been proposed (see [8] for a review). In particular, [13] introduced the `MINE RULE` operator, an extension of SQL specifically designed for modeling the problem of mining ARs from relational databases. The `MINE RULE` operator captures the high-level semantics of AR mining tasks and allows the specification of complex mining tasks with an intuitive SQL-like syntax. For instance, consider the table in Figure 1, contains the purchase records from a clothing store. The transaction column (`tr`) contains an identifier of the transaction; the other columns correspond to the `customer` identifier, the type of purchased `item`, the `date` of the purchase, the unitary `price`, and the purchase quantity (`qty`). Suppose that we want to mine ARs with a minimum support of 0.1, a minimum confidence of 0.2, at most four items in the body, and exactly one item in the head. Using `MINE RULE`, this task is formalized by the following statement:

tr	customer	item	date	price	qty
1	$cust_1$	ski_pants	2001/12/17	140	1
1	$cust_1$	hiking_boots	2001/12/17	180	1
2	$cust_2$	col_shirts	2001/12/18	25	2
2	$cust_2$	brown_boots	2001/12/18	150	1
2	$cust_2$	jackets	2001/12/18	300	1
3	$cust_1$	jackets	2001/12/18	300	1
4	$cust_2$	col_shirts	2001/12/19	25	3
4	$cust_2$	jackets	2001/12/19	300	2

(a) `Purchase` table

HEAD	BODY	SUPPORT	CONFIDENCE
{ski_pants}	{hiking_boots}	0.25	1.00
{hiking_boots}	{ski_pants}	0.25	1.00
{col_shirts}	{brown_boots}	0.25	0.50
{col_shirts}	{jackets}	0.50	1.00
{brown_boots}	{brown_boots}	0.25	1.00
{brown_boots}	{jackets}	0.25	1.00
{jackets}	{col_shirts}	0.50	0.66
{jackets}	{brown_boots}	0.25	0.33
{col_shirts,brown_boots}	{jackets}	0.25	1.00
{col_shirts,jackets}	{brown_boots}	0.25	0.50
{brown_boots,jackets}	{col_shirts}	0.25	1.00

(b) `SimpleAssociation` table

Fig. 1. The `Purchase` table for a store and `SimpleAssociation` table.

```
MINE RULE SimpleAssociations AS
SELECT DISTINCT 1..4 item AS BODY, 1..1 item AS HEAD, SUPPORT, CONFIDENCE
FROM Purchase
GROUP BY tr
EXTRACTING RULES WITH SUPPORT: 0.1, CONFIDENCE: 0.2
```

The statement produces the table `SimpleAssociations` (see Figure 1) where each tuple corresponds to a discovered AR. The `SELECT` clause defines the structure of output table which collects the extracted rules: the rule `BODY` is defined as a set of at most four `item`s, the `HEAD` as a set with one single `item`, and the `DISTINCT` keyword specifies that no replications are allowed in the rule head and body. The resulting table contains the `CONFIDENCE` and `SUPPORT` values for each AR. The `GROUP BY` clause specifies that the set of transactions ($\mathcal{D}$) is derived from the original table `Purchase` by grouping it with respect to the transaction identifier attribute (`tr`). Finally, the `EXTRACTING RULES WITH` clause specifies the minimum support and confidence.

`MINE RULE` is particularly effective when the mining task is complex. Suppose that we want to find interesting intra-customer relations which associate past customer purchases with future ones. Note that in this case we are looking for *generalized* ARs, since there are no constraints on the number of items in the

head and in the body ("`1..n`" indicates that both the rule head and body contain an arbitrary number of items). This task is formalized by the following statement:

```
MINE RULE OrderedSets AS
SELECT DISTINCT 1..n item AS BODY, 1..n item AS HEAD, SUPPORT, CONFIDENCE
FROM Purchase
WHERE date > 2001/12/17
GROUP BY customer
CLUSTER BY date HAVING BODY.date < HEAD.date
EXTRACTING RULES WITH SUPPORT: 0.1, CONFIDENCE: 0.2
```

The `GROUP BY` clause indicates that ARs will be extracted w.r.t. customers and not the transaction identifiers, as in the previous case. In addition, the `CLUSTER BY` clause specifies that the customer purchases should be further grouped by `date` so that all the items in the rule body will have the same date and all the items in the rule head will have same date. The `HAVING` clause specifies a temporal constraint on the output rules: the items in the rule body should have a purchase date previous to those in the rule head.

3 From Relational Data to XML Data

Any AR mining task can be brought down to four main steps: (i) *context identification*, which defines the *structure* of the set $\mathcal{D}$; (ii) *context selection*, which defines constraints on set δ of *transactions in* $\mathcal{D}$ that are relevant to the problem; (iii) *rule selection*, which defines a set of constraints on the *generated ARs*; (iv) *result construction*, which specifies a *format* to represent the generated ARs. For instance, if we consider the second `MINE RULE` example, (i) the structure of the set of transactions $\mathcal{D}$ is derived from the table `Purchase`, first grouped by the attribute `customer` and then by the attribute `date` through the `CLUSTER BY` clause; (ii) only the purchases occurred after `2001/12/17` are considered for the mining purposes; (iii) the items in the rule head must be purchased after those in the rule body; (iv) the `SELECT` clause declares the columns of the output table which represents the result. Note that, steps (ii), (iii), and (iv) are smoothly mapped into XML. In fact, both *context selection* and *rule selection* phases involve the selection of parts of XML documents which define the source data (i.e., $\mathcal{D}$) and the output ARs. These operations are easily expressed in terms of XQuery statements [21], while the *result construction* phase is naturally expressed using the XML node construction statements. In contrast, the *context identification* in XML documents is where the differences between the "flat" relational data model and the hierarchical XML data model fully emerge.

4 Association Rules from XML Data

We now introduce association rules from *native* XML data. The problem we address can be stated as follows. Given a *generic* XML document, we want to extract a set of ARs (still represented in XML) that describe interesting relations among parts, i.e., fragments, of the source XML document. To accomplish

```
<Department>
  <People>
    <Employee>
      <PersonalInfo> <Name>...</Name> ... </PersonalInfo>
      <Education>
        <Higher>
          <MsC year="1996"> yes </MsC>
          <PhD year="1999"> yes </PhD>
        </Higher> ...
      </Education>
      <Publications>
        <Book year="2001" name="XML Query Languages">
          <Author>...</Author>
          <Publisher>...</Publisher>
          <Keyword>XML</Keyword>...<Keyword>XQuery</Keyword>
        </Book> ...
        <Journal year="2000" month="4" vol="4" name="DMKD"
                 publisher="Kluwer">
          <Author>...</Author>
          <Keyword>RDF</Keyword>...<Keyword>XML</Keyword>
        </Journal>
      </Publications>
      <Awards>
        <Award year="2001" Society="IEEE">This award ...</Award>
      </Awards>
    </Employee> ...
  </People> ...
</Department>
```

Fig. 2. An XML document `http://www.cs.atlantis.com/staff.xml` with various information about the personnel of a university department

this, we follow the approach of [13], and define an extension of the XML query language, XQuery [21], which captures the high-level semantics of the problem and allows the specification of complex mining tasks within an intuitive syntax, inspired to the XML world. We will refer to a sample XML document, depicted in Figure 2, which describes the personnel of a university department.

4.1 Simple Mining

As a first basic example, we wish to discover the associations among the research topics that have been investigated in recent years. For this purpose we extract rules which associate the keywords of the publications since 1990. More specifically, we are interested in ARs of the form "$XML \Rightarrow XQuery$ with support 0.1 and confidence 0.4" which state that researchers who published about XML also published something related to XQuery in 40% of the cases and that this rule applies to 10% of the personnel. We can formulate this as follows.

```
IN document("http://www.cs.atlantis.com/staff.xml")
MINE RULE
FOR ROOT IN /Department//Employee
LET BODY := ROOT/Publications//Keyword,
    HEAD := ROOT/Publications//Keyword
WHERE ROOT//@year > 1990
EXTRACTING RULES WITH SUPPORT = 0.1 AND CONFIDENCE = 0.4
RETURN <AssocRule support = { SUPPORT } confidence = { CONFIDENCE }
                  antecedent = { BODY } consequent = { HEAD } />
```

The `IN` clause specifies the source data. In this case we address *one* XML document (in Figure 2), but in general any XQuery statement to construct the data source might be used instead.
The `FOR` clause implements the *context identification* step. A special variable, `ROOT`, defines through an XPath [20] expression the set of XML fragments constituting the *context* in which the ARs will be extracted (this set corresponds to the set of transactions $\mathcal{D}$ in the relational framework). The `ROOT` variable is fundamental to the process since support and confidence values are computed w.r.t. the fragments in `ROOT`. In the above statement, the XPath expression `/Department//Employee/Publications` specifies that we are interested in rules that are extracted from the publications of *each* employee. Then two special variables, `BODY` and `HEAD`, identify the XML fragments which should appear in the rule body and head respectively. Note that both `BODY` and `HEAD` *have* to be defined w.r.t. the *context* (i.e., to the `ROOT` variable), since confidence and support values have meaning only w.r.t. the *context*. More specifically, both `BODY` and `HEAD` are defined as `Keyword` elements appearing in the `Publications` element of each employee. Since the `Keyword` element appears both in journals and in books, `Keyword` is specified as a subelement of `Publications` placed to an arbitrary level of depth (with the "`//`" XPath step).
The `WHERE` clause filters out some of the source fragments, thus expressing the conditions for the *context selection*. Our example requires that among all the subelements of `Publications` only those published after 1990 are considered.
The `EXTRACTING` clause specifies the minimum support and confidence values.
The final `RETURN` clause imposes the structure of the generated rules. An AR is defined by an `AssocRule` tag, with four attributes, `support` and `confidence` which specify the support and confidence values, `antecedent` and `consequent`, which specifies the items in the head and body respectively. A sketch of a possible result produced by the above statement follows.

```
<AssocRule support="0.25" confidence="0.45"
           antecedent="XML" consequent="XQuery" />
...
<AssocRule support="0.3" ...  consequent="RDF" />
```

4.2 Advanced Mining

We now move to a slightly more complex AR mining problem. Suppose we wish to find out any interesting dependency between the fact that employees have

published books with particular publishers along their career, and the fact that they have later been awarded by a particular society. In addition, we require that the employee is actually an author, and not just an editor of the book, and that the employee has a PhD degree. This problem can be mapped into an AR mining task. Rules have a set of `Publisher` XML elements in the body and a set of `Society` elements in the head, and these elements must be descendants of *the same* `Employee` node. This task can be formulated as follows:

```
IN document("http://www.cs.atlantis.com/staff.xml")
MINE RULE
FOR ROOT IN /Department/Employee
LET BODY := ROOT/Publications/Book/Publisher,
    HEAD := ROOT//Award/@Society
WHERE count(ROOT//Education//PhD) > 0 AND
      contains(BODY//Author,ROOT/PersonalInfo/Name) AND
      BODY/../@year < HEAD/../@year
EXTRACTING RULES WITH count(BODY) = 1 AND count(HEAD) = 1 AND
                      SUPPORT = 0.1 AND CONFIDENCE = 0.2
```

The mining *context* (`ROOT`) is defined with the XPath expression `/Department/People/Employee` which retrieves the set of XML fragments which represent each employee. Then `BODY` is defined as the set of `Publisher` elements within the books published by an employee (referenced by `ROOT`), while `HEAD` is defined as the set of `Society` attributes of every award received by the *same* employee (still referenced by `ROOT`).
In the `WHERE` clause, "`count(ROOT//Education//PhD) > 0`" requires that only the employees with a PhD degree are considered; the clause "`contains(BODY//Author,ROOT/Personal/Name)`" requires that the employee's name appears among the authors of the book (`contains` is one of the core XQuery functions [21]); "`BODY/../@year < HEAD/../@year`" requires that the year of publication of the books precedes the year of the award in the rule head. In the final `EXTRACTING` clause, the "`count(...)=1`" clauses require that both the rule body and head contain exactly one XML fragment.
The `RETURN` clause is optional. If not included, the output is an XML document which contains a set of ARs expressed according to the Predictive Model Markup Language (PMML) standard [2] (a sketch of the produced result follows).

```
<PMML>
 <AssociationModel>
  <AssocItem id="307" value="IEEE" />
   ...
  <AssocItemset id="1" support="..." numberOfItems="1">
   <AssocItemRef itemRef="307" />
  </AssocItemset>
   ...
  <AssocRule support="0.2" confidence="1.0"
             antecedent="2" consequent="1" />
 </AssociationModel>
</PMML>
```

ARs are described by `AssocRule` elements connecting two sets of items, identified by two `AssocItemSet` elements. Each item set is in turn defined as a collection of items (`AssocItem` elements), identified by `id`s. It is easy to note that the PMML representation for ARs tends to be quite lengthy as the complexity of the mining problem increases. For this reason, we included the `RETURN` clause to allow the specification of (possibly) more compact formats for the output result.

5 Related Work

Association rules (ARs) have been first introduced in [4] upon retail databases. The famous Apriori algorithm for extracting ARs was independently published in [5] and in [12]. Although the first algorithms assumed that transactions were represented as sets of binary attributes, along the years many algorithms for the extraction of ARs from multivariate data have been proposed. This includes the use of quantitative attributes (e.g., [18,10]) as well as the integration of concept hierarchies to support the mining of ARs from different levels of abstractions (e.g., [7,17]). Association rules and *episode rules* (a modification of the concept of ARs introduced in [11]) have been applied to the mining of plain text (e.g., [6,15]), or to semi-structured data [16]. However, as far as we know, ARs within the context of *native* XML documents have never been proposed. To assist association rule mining, a number of query languages have been proposed. The `MINE RULE` operator, used in this paper, was first introduced in [13] and extended in [14]. Separately, [9] introduced MSQL, an SQL extension which implements various operations over association rules (e.g., generation, selection, and pruning). [8] introduced DMSQL, a general purpose query language to specify data mining tasks.

Data Mining and 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* XML documents. The Predictive Model Markup Language (PMML) [2] is an XML standard which can be used to represent most of the types of patterns which can be extracted from the data, e.g., association rules, decision trees, decision rules, etc. PMML does not represent the data themselves but it only allows the description of the data.

6 Conclusions

We have introduced association rules (ARs) from native XML data, starting from their formal definition and the consolidated `MINE RULE` operator [13], a declarative operator that formulates ARs mining tasks on relational data by means of statements that recall SQL. In the same spirit, we defined ARs on native XML data using an XQuery-like operator which allowed us to demonstrate some semantic features of the AR mining from XML data.

Many issues remain open, and there are a number of future research directions; an interesting direction regards the possibility to simultaneously mine both

the structure and the contents of XML documents. In fact, XML sets a smooth distinction between the data and their structure.

Another relevant open issue concerns the specification of the *context* for a giving mining problem. Dealing with many generic XML documents, their physical structure might limit the feasible contexts which can be used to extract association rules. This depends on the fact that we use plain XPath expressions to identify the mining context. Our choice is strongly motivated by the intuitiveness and diffusion of the XPath formalism, but if the information is not stored according to the correct mining criteria that suit the user needs, *context identification* becomes a hard task. In order to meet the user needs, a set of transformations might be required, which should simplify the specification of the right context for a specific mining tasks, even if the document schema does not match the given task. In fact, a more general question is how to consider and connect all available metadata, not only the XML DTD or schema information as above, for a specific mining task in a specific domain. As more and more data will be encoded in XML, also an increasing amount of background knowledge will be available in, e.g., RDF metadata or DAML+OIL format ontologies. This additional metadata provides enhanced possibilities to constrain the mining queries both automatically and manually, but it also increases the complexity and thus poses new requirements upon the data mining query language.

Finally, association rules from XML data could be extended to the case of *episode rules*. These are a modification of association rules that take into special account the temporal relations among the different times at which the investigated items are generated. A suitable extension of our proposal could address this specific domain, that proved to be interesting in some telecommunications fields (to mine the log files of nodes in networks for interesting time series that precede several kinds of network faults).

Acknowledgements

The authors have been supported by the *consortium on discovering knowledge with Inductive Queries* (*cInQ*) [1], a project funded by the Future and Emerging Technologies arm of the IST Programme (Contr. no. IST-2000-26469). Daniele Braga and Alessandro Campi are supported by Microsoft Research. The authors wish to thank Stefano Ceri for the inspiration and discussions that made this work possible.

References

1. consortium on discovering knowledge with INductive Queries (*cInQ*). `http://www.cinq-project.org`.
2. PMML 2.0 Predictive Model Markup Language. `http://www.dmg.org`, August 2001.
3. Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *SIGMOD93*, pages 207 – 216, Washington, D.C., USA, May 1993.

4. 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 the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.
5. Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee L. P. Chen, editors, *Proc. 11th Int. Conf. Data Engineering, ICDE*, pages 3–14. IEEE Press, 6–10 1995.
6. Helena Ahonen, Oskari Heinonen, Mika Klemettinen, and A. Inkeri Verkamo. Mining in the phrasal frontier. In *Principles of Data Mining and Knowledge Discovery*, pages 343–350, 1997.
7. J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. of 1995 Int'l Conf. on Very Large Data Bases (VLDB'95), Zürich, Switzerland, September 1995*, pages 420–431, 1995.
8. Jiawei Han and Micheline Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufmann, San Francisco (CA).
9. Tomasz Imielinski and Aashu Virmani. MSQL: A query language for database mining, 1999.
10. Brian Lent, Arun N. Swami, and Jennifer Widom. Clustering association rules. In *ICDE*, pages 220–231, 1997.
11. Heikki Mannila, Hannu Toivonen, and et al. Discovering frequent episodes in sequences (extended abstract), August 1995.
12. Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, pages 181–192, Seattle, Washington, 1994. AAAI Press.
13. Rosa Meo, Giuseppe Psaila, and Stefano Ceri. A new sql-like operator for mining association rules. In *VLDB'96, September 3-6, 1996, Mumbai (Bombay), India*, pages 122–133.
14. Rosa Meo, Giuseppe Psaila, and Stefano Ceri. A tightly-coupled architecture for data mining. In *ICDE*, pages 316–323, Orlando, Florida, USA, February 1998.
15. M. Rajman and R. Besanon. Text mining: Natural language techniques and text mining applications, 1997.
16. Lisa Singh, Peter Scheuermann, and Bin Chen. Generating association rules from semi-structured documents using an extended concept hierarchy. In *CIKM*, pages 193–200, 1997.
17. Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *The VLDB Journal*, pages 407–419, 1995.
18. Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, Montreal, Quebec, Canada, 4–6 1996.
19. World Wide Web Consortium. Extensible Markup Language (XML) Version 1.0 W3C Recommendation. `http://www.w3c.org/xml/`, February 1998.
20. World Wide Web Consortium. XML Path Language (XPath) Version 1.0, W3C Recommendation. `http://www.w3c.org/tr/xpath/`, November 1999.
21. World Wide Web Consortium. XQuery 1.0: An XML Query Language W3C Working Draft. `http://www.w3.org/TR/2001/WD-xquery-20010607`, June 2001.