Intensional Query Answering to XQuery Expressions

Simone Gasparini, Elisa Quintarelli

Intensional Query Answering to XQuery expressions

Simone Gasparini and Elisa Quintarelli

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci, 32 — 20133 Milano (Italy)
{gasparini,quintarelli}@elet.polimi.it

Abstract. XML is a representation of data which may require huge amounts of storage space and query processing time. Summarized representations of XML data provide succinct information which can be directly queried, either when fast yet approximate answers are sufficient, or when the actual dataset is not available. In this work we show which kinds of XQuery expressions admit a partial answer by using association rules extracted from XML datasets. Such partial information provide intensional answers to queries formulated as XQuery expressions.

1 Introduction

The eXtensible Markup Language (XML) [12] was initially proposed as a standard way to design markup languages to represent, exchange and publish information on the Web, but its usage has recently spread to many other application fields.

XML is a rather verbose representation of data, which may require huge amounts of storage space and query processing time. In [2] several summarized representations of XML data are proposed to provide succinct information and be directly queried. In particular, the notion of *patterns* is introduced as abstract representations of the constraints that hold on the data and for (possibly partially) answering queries, either when fast (but approximate) answers are required, or when the actual dataset is not available or it is currently unreachable.

In this work we show which kinds of queries admit a partial answer by means of association rules extracted from XML datasets by using data mining techniques.

In particular, once a XML dataset has been analyzed by a miner tool and a set of association rules has been extracted, we investigate how to transform an XQuery expression to be applied to the original XML dataset, in order to apply it to the set of rules previously extracted. In this way we provide an approximate intensional answer.

An intensional answer to a query substitutes the actual data answering the query (the extensional answer) with a set of properties (in our work, with a set of association rules) characterizing them [11]. Thus, intensional answers are in general more synthetic than the extensional ones, but usually approximate.

In order to achieve our goal, an intuitive and effective language is needed to query the extracted knowledge.

We focus on XQuery, the standard XML query language introduced by the W3C [13]. In particular, we propose the fragment of XQuery expressions that can be used

to retrieve useful information from the extracted sets of association rules. Such useful information can provide intensional answers to queries formulated as XQuery expressions. In [7] we have focused also on a graph-based language, and in particular on XQBE [3], because the user could (visually) express a query without taking care about the details of the language really used to query the document. In this way, the overall querying process appears completely transparent to the user: if the actual dataset is not available, the intensional (approximate) answer will be automatically provided by querying the rule set.

The paper is organized as follows. Section 2 summarizes the different types of patterns proposed in [2] and briefly describes how to represent them in a graph-based formalism. In Section 3 we propose some examples which show the set of queries we can manage with our approach and how to transform XQuery expressions in order to retrieve intensional information about XML documents; the formalization of the transformation process is in [7]. Previous work is discussed in Section 4, while conclusions and possible lines for future work are presented in Section 5.

2 Patterns for XML Documents

The summarized representations introduced in [2] are based on the extraction of association rules from XML datasets. Association rules describe the co-occurrence of data items in a large amount of collected data [1] and are usually represented as implications in the form $X \Rightarrow Y$, where X and Y are two arbitrary sets of data items, such that $X \cap Y = \emptyset$. In the XML context, a data item is a pair *(data-element,value)*, e.g. (`Conference`,`Pods`). The quality of an association rule is usually measured by means of *support* and *confidence*. Support corresponds to the frequency of the set $X \cup Y$ in the dataset, while confidence corresponds to the conditional probability of finding Y, having found X and is given by $sup(X \cup Y)/sup(X)$.

In [2] patterns are classified in two orthogonal ways. The first classification refers to the precision with which the pattern represents the dataset: a) an *exact* pattern expresses a property which holds on *any* instance of the dataset. Thus exact patterns represent *constraints* (e.g. functional dependencies between schema elements by means of schema patterns). For example, the name and the edition of a conference identify the location where the conference has taken place. b) A *probabilistic* pattern holds only on a given (large) fraction of the instances in the dataset. It is a weak constraint on the dataset, characterized by a quality index describing its reliability. For example, with a confidence of 0.9 the name of a conference identifies its main topics.

The second classification dimension corresponds to the different summarization levels of the represented information. *Instance* patterns are expressed on the instances of the dataset. In this paper they are used to summarize the content of a XML dataset by means of the most relevant (frequent) association rules holding on the dataset. As proposed in [2], association rules are extracted by using mining algorithms; in particular, to define the concept of transaction a *transaction root* (i.e. an appropriate element of the considered XML document) is selected. A transaction is then defined as a collection of pairs *(element tag,content)* or *(attribute name,value)*, where *element tag* (or *attribute name*) is the name of an element (or attribute) rooted in the transaction root

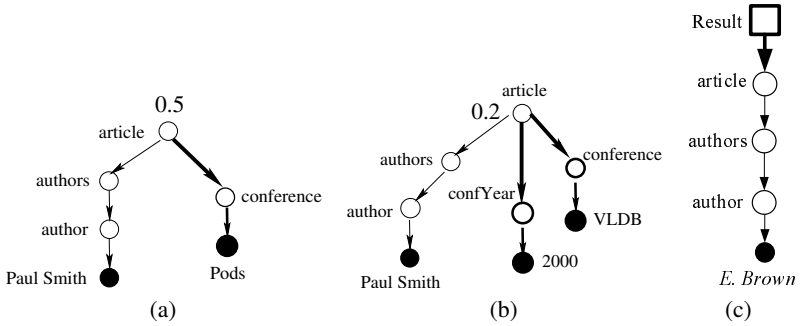

Fig. 1. Three instance patterns

and is defined as a complete sub-path from the root to the element. The quality index of the pattern is the confidence of the rule.

We use previously extracted association rules to derive an approximate answer to an XQuery expression, without requiring to actually access the dataset to compute the answer. The answer may contain a subset or a superset of the required information, depending on the form of the query and of the considered instance patterns.

In this work we focus our attention on (probabilistic) instance patterns. An example of instance pattern is the following: with a confidence of 0.8 the author *Paul Smith* has a publication to at least an edition of the conference *ICDT*.

In [2] a tree-based representation of patterns, which is formalized by means of the language GSL, is proposed as well.

For example, the instance pattern (a) of Figure 1 represents the association rule stating that with a confidence of 0.5 Paul Smith had a publication to any edition of the Pods conference. In the graphical version of patterns we represent nodes with circles (black filled circles represent the content of leaf elements or the value of attribute) and indicate the confidence of the instance pattern on the root of the graph. Thin lines are used to represent the body of the association rule, whereas thick lines represent the head of a rule. A more complex instance pattern expressing an association rule with more than one path in the thick part of the tree (i.e., in the association rule head), is depicted in Figure 1.(b). The rule states that with a confidence of 0.2 Paul Smith had a publication to the conference VLDB 2000. Note that here the confidence is associated to the conjunction of the two conditions in the head of the instance pattern.

3 Experimental setup

In [7] the set of queries which can be considered to obtain approximate answers by using instance patterns is introduced. For the sake of space, in this work we present our idea only by examples.

For our first experiments we have used a dataset based on a slight variation of the SIGMOD Record XML Document [10]. The document reports information about Conference Proceedings; Listing 1.1 reports a XML fragment of the document itself.

```
<articles>
   <article year"2001">
      <volume>30</volume>
      <number>2</number>
      <month>June</month>
      <conference>ACM SIGMOD International Conference on Management of Data</conference>
      <date>May 21 - 24, 2001</date>
      <location>Santa Barbara, California, USA</location>
      <title articleCode="302001">Securing XML Documents ...</title>
      <authors>
         <author authorPosition="01">E. Brown</author>
         <author authorPosition="02">L. Baines</author>
      </authors>
      <indexTerms>
         <term>XML</term>
         <term>Security</term>
         <term>XQuery</term>
         <term>Theory</term>
      <indexTerms>
   </article>
   ...
</articles>
```

Listing 1.1. A portion of the sample document inspired to the SIGMOD Record [10]

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT RuleSet (AssociationRule+)>
<!ELEMENT AssociationRule (RuleBody, RuleHead)>
<!ATTLIST AssociationRule
   support CDATA #REQUIRED
   confidence CDATA #REQUIRED>
<!ELEMENT RuleBody (item+)>
<!ELEMENT RuleHead (item+)>
<!ELEMENT item (ItemName, ItemValue)>
<!ELEMENT ItemName (#PCDATA)>
<!ELEMENT ItemValue (#PCDATA)>
```

Listing 1.2. The DTD of the document reporting the extracted association rules set

Starting from this dataset, we perform a mining process to extract association rules. Most of the proposed algorithms for mining association rules [1], [8] consider a collection of transactions, each containing a set of items. In our examples, we have associated each transaction to an `article`, thus, we have extracted association rules describing information about the elements which characterize articles (e.g. `author`, `title`, etc.).

In order to retrieve intensional answers from the set of extracted association rules, we store them in a XML document. Listing 1.2 reports the relevant Document Type Definition (DTD) we use to represent the rule set. We partially take inspiration from the PMML (Predictive Model Markup Language) standard model proposed by the Data Mining Group [5], which describes statistical and data mining models. Our model, however, is simpler and easier to query, i.e. it requires a less complex XQuery expression to formulate a query. For example, a portion of the valid XML document representing some association rules, which have been extracted from the dataset based on the SIGMOD Record XML Document, is shown in Listing 1.3.

```
<ruleSet>
   <AssociationRule support="0.2" confidence="0.8">
      <RuleBody>
         <item><ItemName>author</ItemName><ItemValue>E. Brown</ItemValue></item>
      </RuleBody>
      <RuleHead>
         <item><ItemName>term</ItemName><ItemValue>XML</ItemValue></item>
      </RuleHead>
   </AssociationRule>
   ...
</ruleSet>
```

Listing 1.3. A sample fragment of the XML document of the rule set extracted from the sample document of Listing 1.1

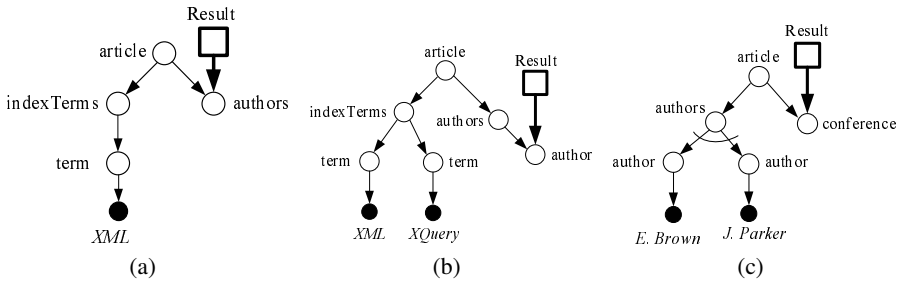

Fig. 2. (a) GSL visual representation of query $Q2$; (b) GSL visual representation of query $Q4$; (c) GSL visual representation of query $Q5$;

In the following we analyze the kinds of queries described in [7] that admit a partial answer by using association rules extracted from XML datasets.

3.1 Queries with conditions on content nodes

Let us consider the first kind of query with conditions on a content node; i.e. queries imposing a restriction on the value of an attribute or on the content of a leaf element of an XML dataset. An example is the query $Q1$ "*List all the information about the articles published by E. Brown*", which contains a condition on the name of an author. The GSL representation is depicted in Figure 1.(c). An XQuery expression for this query is:

```
<result> {
 for $article in doc("document.xml")//article
 where $article/authors/author/text() = "E. Brown"
 return $article }
</result>
```

The above expression can be run on any XQuery engine to get the extensional answer to $Q1$. In order to get an intensional answer from the extracted rule set, we have to transform the original XQuery expression.

For the query $Q1$, according to the DTD for the extracted association rules described before (Listing 1.2), the XQuery expression should be modified as follows:

```
<result> {
 for $article in doc("RuleSet.xml")//AssociationRule
 where $article[RuleBody[item[ItemName="author" and ItemValue="E. Brown"]]]
 return $article }
</result>
```

This expression returns all the association rules that have an item named `author` in the body of the rule, whose value is `E. Brown`. The changes required (underlined in the listing) to query the rule set affect the name of the document to examine (from now on we suppose that the rule set is stored in a XML file named `RuleSet.xml`), the item in the `for` clause, which has to be `AssociationRule`, and the expression in the `where` clause, modified to select only those rules having an item in the body with the same name of the `author` element.

Similarly, to get the association rules that satisfy the condition in the head of the rule, `RuleBody` has to be replaced with `RuleHead` in the `where` clause. A more generic query that looks for interesting information about `E.Brown` both in the body and in the head can be expressed as follows:

```
<result> {
 for $article in doc("RuleSet.xml")//AssociationRule
 where $article//item[ItemName = "author" and ItemValue = "E. Brown"]]]
 return $article }
</result>
```

By using the XPath expression `//item`, the `where` clause selects all the association rules which have an item named `author` in the body or in the head of the rule and whose value is `E. Brown`.

Let us consider now the query that retrieves information about a node which is not a direct ancestor of the constrained content node. An example of this kind of query is the query $Q2$: "*List all the authors who wrote articles about XML*". Figure 2.(a) shows how it can be graphically represented in GSL. Like $Q1$, the source part matches all the `author` having the term `XML` among the index terms of their published articles; the query result will contain all these authors. The XQuery expression is:

```
<result> {
 for $article in doc("document.xml")//article
 where $article/indexTerms/term/text() = "XML"
 return $article/authors/author }
</result>
```

In order to inquire the extracted rule set, the above XQuery expression needs the following transformations:

```
<result> {
 for $article in doc("RuleSet.xml")//AssociationRule
 where $article/RuleBody/item[ItemName = "term"]/ItemValue= "XML"
 return $article[RuleHead/item[ItemName = "author"]] }
</result>
```

The expression returns all the association rules that have an item `term` with required value in the body and has an item `author` in the head, such as the rule relating the author `E. Brown` and the index term `XML` in the rule set listed in Listing1.3. The changes that are required are similar to the ones for query $Q1$, but in this case also a filter on the returned `$author` variable is introduced in order to select only those association rules which contain the item `author` in the head of the rule. In a similar way, by swapping

RuleHead and RuleBody in these expressions, the rules with term in the head and author in the body can be obtained.

The GSL language used to represent XML patterns and the extraction process of association rules make no distinction between elements and attributes of XML dataset since the main aim is to find relationships among elementary values of XML documents. However this choice can be easily tackled transparently to the user, as the following example demonstrates. Let consider the query $Q3$ with condition on the value of an attribute: "*List all the conference held in 1996*". An XQuery expression of $Q3$ is:

```
<result> {
 for $article in doc("document.xml")//article
 where $article/@year = 1996
 return $article/conference }
</result>
```

Due to the mining process, we loose the distinction between elements and attribute and so we have to transform the above expression simply by considering in the same way attributes and elements. Thus the XQuery expression to query the rule set becomes:

```
<result> {
 for $article in doc("RuleSet.xml")//AssociationRule
 where $article/RuleBody/item[ItemName = "year"]/ItemValue= "1996"
 return $article[RuleHead/item[ItemName = "conference"]]}
</result>
```

The expression returns all the rules relating year in the body and conference in the head.

3.2 Queries with AND-conditions on content nodes

Let us now focus on query with AND-conditions on the content nodes. As an example, consider query $Q4$: "*List all the authors who have published articles about XML and XQuery*" (see Figure 2.(b) for the visual representation). A related XQuery expression is:

```
<result> {
 for $article in doc("document.xml")//article
 where $article/indexTerms/term/text() = "XML"
 and $article/indexTerms/term/text() = "XQuery"
 return $article/authors/author }
</result>
```

In order to provide intensional answer to $Q4$ by querying the extracted rule set, the expression has to be modified in the following way:

```
<result> {
 for $article in doc("RuleSet.xml")//AssociationRule
 where $article/RuleBody/item[ItemName = "term"]/ItemValue= "XML"
 and $article/RuleBody/item[ItemName = "term"]/ItemValue= "XQuery"
 return $article[RuleHead/item[ItemName = "author"]]}
</result>
```

This XQuery expression returns all the association rules satisfying both the conditions on the term in the body of the rule and having an item author in the head.

3.3 Queries with OR-conditions on content nodes

Another type of query to consider is the one with two or more OR-conditions. An example of this kind of query is *Q*5: "*List all the conference attended by J. Parker or E. Brown*" (see Figure 2.(c) for the graphical representation. The arc between the two edges represents a disjunctive condition). The equivalent XQuery expression is:

```
<result> {
 for $article in doc("document.xml")//article
 where $article/authors/author = "J. Parker"
 or $article/authors/author = "E. Brown"
 return $article/conference }
</result>
```

In order to query the rule set, we carry out the following adjustments:

```
<result> {
 for $article in doc("RuleSet.xml")//AssociationRule
 where $article/RuleBody/item[ItemName = "author"]/ItemValue= "J. Parker"
 or $article/RuleBody/item[ItemName = "author"]/ItemValue= "E. Brown"
 return $article[RuleHead/item[ItemName = "conference"]]}
</result>
```

Applying this XQuery expression to the rule set, we obtain all the association rules having an item `conference` in the head and satisfying the OR-conditions about both `author` elements in the body of the rule.

3.4 Queries with Element values

Finally let us consider the query that lists all the different values of a content node. An example is the query *Q*6 "*List all the authors who wrote an article, sorting in a lexicographic order*". A relevant XQuery expression can be:

```
<result> {
 for $author in doc("document.xml")//author
 order by $author/text() ascending
 return <author> {$author/text()} </author>}
</result>
```

The above expression should be modified to query the rule set as it follows:

```
<result> {
 for $author in doc("RuleSet.xml")//item[ItemName="author"]/ItemValue
 order by $author/text() ascending
 return <author> {$author/text()} </author>}
</result>
```

This XQuery expression selects all the association rules having an item `author` in the body or in the head of the rule and returns the content of the retrieved `author` elements, lexicographically sorted. This type of query requires a slight adjustment in the expression of the `for` clause to filter the appropriate item in the rules.

4 Related Works

The problem of providing intensional answers by means of integrity constraints has been initially addressed in [11] in the relational databases context. In this work we extend the approach to graph-based probabilistic patterns and XML documents. We starts

from the results published in [2], where several summarized representations of XML data are proposed to provide succinct information: the notion of patterns is introduced as abstract representations of the constraints that hold on the data and a graph-based representation of patterns is proposed as well. In particular, instance patterns are represented by the most frequent association rules and are effectively extracted from XML documents by using data-mining algorithms (e.g. by using Apriori). Some preliminary ideas on the possibility to use such instance patterns to provide intensional answers to user's queries are sketched. In this work we extend the result of [2] in order to apply the proposed approach to the XQuery language and we identify some classes of queries that admit an approximate answer by using previously extracted association rules.

Another two works which present a framework to discover association rules in large amounts of XML data are [4, 6].

In [4] the authors introduce a proposal to enrich XQuery with data mining and knowledge discovery capabilities by introducing association rules for native XML documents and a specific operator for describing them. They formalize the syntax and an intuitive semantics for the operator and propose some examples of complex association rules. No algorithm for mining such complex rules is proposed, thus, we have decided to start from the results in [2] and use, as a first step, very simple association rules in order to partially answer to XQuery expressions.

In [6] a template model to specify XML-enabled associations to be effectively mined is presented. In our opinion our work differs from [4, 6] because we do not focus on the problems of representing and extracting complex association rules from XML documents, instead we work on the possibility to use and manipulate the extracted knowledge (at the moment we mine instance patterns by using the Apriori algorithm) in order to give partial and approximate answers to XQuery expressions. As a future work we may consider the method described in [6] to improve the performance of the mining process and to be able to consider also more complex association rules on XML document.

5 Conclusion and Future Work

In this work we have shown how to use association rules to provide intensional answers and obtain approximate information about XML documents. In particular, we have explained which kinds of XQuery expressions admit an approximate answer and how to transform them in order to query previously mined association rules.

We have built a first prototype environment implemented in Java and is mainly composed of two components. The first one visualizes the DTD of a XML document in a graph-based representation. The user chooses the items to include in the process of extraction of association rules by indicating also where to apply stemming, stopwords, and discretization procedures. The native XML document is then transformed in order to be processed by the Apriori algorithm. The second component stores the output of the miner into a MySQL database. To conclude, a graphical interface gives to the user the possibility to query the extracted knowledge by using some classes of queries and more in particular by providing values for few parameters that are then used to automatically compose SQL queries to be applied to the MySQL database of rules. We are now

adapting the tool in order to transform the output of the miner into a XML document. Another component under development uses the Saxon [9] engine to apply an XQuery expression either to the original XML document or to the extracted set of rules.

As an ongoing work we are formalizing the degree of approximation (and the time performance) of our approach in answering queries and studying how to combine constraints and association rules to improve the precision of intensional answers. We are also considering extensions of the XQuery fragment proposed in this work that admit partial answers.

Acknowledgment

We like to thank Letizia Tanca for the very useful comments on this work and Elena Baralis and Paolo Garza for the precious collaboration in the setting foundations of this research.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.
2. E. Baralis, P. Garza, E. Quintarelli, and L. Tanca. Answering queries on XML data by means of associations rules. In *Current Trends in Database Technology*, volume 3268. Springer-Verlag, 2004.
3. D. Braga and A. Campi. A graphical environment to query XML data with XQuery. In *Proc. of the Fourth International Conference on Web Information Systems Engineering (WISE'03)*, pages 31–40. IEEE Computer Society, 2003.
4. D. Braga, A. Campi, M. Klemettinen, and P.L. Lanzi. Mining association rules from XML data. In *Proc. of the 2003 ACM Symposium on Applied Computing*, volume 2454, pages 21–30. Lecture Notes in Computer Science, 2002.
5. Data Mining Group. PMML 2.1 – DTD of association rules model. `http://www.dmg.org`.
6. L. Feng and T. Dillon. Mining XML-Enabled Association Rules with Templates. In *Proc. of the Third International Workshop on Knowledge Discovery in Inductive Databases*, volume 3377, pages 66–88. Lecture Notes in Computer Science, 2004.
7. S. Gasparini and E. Quintarelli. Intensional Query Answering to XQuery expressions. Technical Report 2005, Politecnico di Milano. `http://www.elet.polimi.it/upload/quintare/Papers/GQ05-Report.pdf`.
8. Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *2000 ACM SIGMOD Int. Conference on Management of Data*, pages 1–12. ACM Press, 2000.
9. M. Kay. Saxon – the XSLT and XQuery processor. `http://saxon.sourceforge.net/`, 2004.
10. P. Merialdo. SIGMOD RECORD in XML. `http://www.acm.org/sigmod/record/xml`, 2003.
11. A. Motro. Using integrity constraints to provide intensional answers to relational queries. In *Proceedings of the 15th International Conference on Very Large Data Bases*, pages 237–246. Morgan Kaufmann Publishers Inc., 1989.
12. World Wide Web Consortium. Extensible Markup Language (XML) 1.0, 1998. `http://www.w3C.org/TR/REC-xml/`.
13. World Wide Web Consortium. XQuery: An XML Query Language, 2002. `http://www.w3C.org/TR/REC-xml/`.