# Answering Xml Query Using Tree Based Association Rules

Anam V Bhaskara Reddy
*M.Tech Student, CVSR College of Engineering, Department of Computer Science, A.P. India*

B.Ujwala
*[2] Assistant Professor of Computer Science and Engineering, Anurag group of Institutions, A.P. India*

**Abstract— We describe an approach based on Tree-based Association Rules(TARs) mined rules, which provide approximate, intensional information on both the structure and the contents of XML documents and can be stored in XML format as well. There are two main approaches to XML document access: Keyword-based Search and Query-Answering. The idea of mining association rules to provide summarized representations of XML documents has been investigated in many proposals either by using languages XQuery,JQuery etc., and techniques developed in the XML context or by implementing graph-or-tree-based algorithms. In this paper, we introduce a proposal for mining and storing TARs (Tree-based Association Rules) as a means to represent intensional knowledge in native XML.**

**Index Terms— XML, Keyword search, Intensional answers, Approximate answering, Succinct answers.**

## I. INTRODUCTION

XML has become a popular format for storing and sharing data across heterogeneous platforms. The XML [9] format is neutral, flexible and interoperable. It is widely used in applications as it can allow applications to have communication though they are built in different platforms. The XML documents are plenty in enterprises and the data retrieval can be done in two ways. The first approach is that user gives keywords and the program searches for relevant documents. The second approach is give XML queries that are answered. The first approach is done using conventional information retrieval technique that works on the search process based on the given search word. With respect to query answering [14], it is not easy to process such request. To make this searching easy this paper presents data mining for XML query answering support. XML documents are validated by either DTD or schema [9]. However, schema presence is not mandatory to process XML file.

This paper presents data mining framework for XML query answering support. The XML documents essence is extracted and kept in another XML file in the form of TARs. With the help of this XML query answering becomes easy.

## II. RELATED TOOLS

*XQuery*

XQuery [10] was designed to query XML data. XQuery is to XML what SQL is to database tables. XQuery is designed to query XML data - not just XML files, but anything that can appear as XML, including databases.XQuery is built on XPath expressions. XQuery is supported by all major databases. XQuery is a W3C Recommendation. XQuery is a language for finding and extracting elements and attributes from XML documents. XQuery can be used to

(a) Extract information to use in a Web Service.
(b) Generate summary reports.
(c) Transform XML data to XHTML.
(d) Search Web documents for relevant information.

*XQuery Example*

for $x inoc("student.xml")/college/stud where $x/rollno>30 order by $x/rollno return $x/name XQuery is compatible with several W3C standards, such as XML, Namespaces, XSLT, XPath, and XML Schema.

Introduction to XPath: XPath is used to navigate through elements and attributes in an XML document. XPath is a major element in W3C's XSLT standard - and XQuery and XPointer are both built on XPathexpressions.XPath is a syntax for defining parts of an XML document.

XPath uses path expressions to navigate in XML documents. XPath contains a library of standard functions. XPath is a major element in XSLT. XPath is a W3C recommendation. XPath Path Expressions: XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system.

## III. DATA MODEL

In XAL, each XML document is represented as a rooted directed graph with a partial order relation defined on its edges. Niagara also model the XML document as rooted directed graph with elements and attributes Corresponding to vertices in the graph. TAX treats the XML documents as forest of labeled rooted trees. Each node of the trees has a virtual attribute called *pedigree* which carries the history of "where it came from" as trees are manipulated by operators. Basic unit of information Analogous to relational algebra operators who operate on collections of tuples, an operator of XML algebra performs manipulation on collections of entities. Basic unit of information refers to the individual member of these collections, or counterpart in XML for a relational tuple. In XAL, the basic unit is the vertex which represents either an element or attribute. An operator receives a set of vertices as input and produces a set of vertices as output

## IV. GOAL AND CONTRIBUTION

In this method for deriving intentional knowledge from XML documents in the form of TARs, and then storing these TARs as an alternative, synthetic dataset to be queried for providing quick and summarized answers. Our procedure is characterized by the following key aspects:

- It works directly on the XML documents, without transforming the data into any intermediate format.
- It looks for general association rules, without the need to impose what should be contained in the antecedent and consequent of the rule.
- It stores association rules in XML format.
- It translates the queries on the original dataset into queries on the TARs set.
- The aim of our proposal is to provide a way to use intentional knowledge as a substitute of the original document during querying and not to improve the execution time of the queries over the original XML dataset.

Accordingly, the project contributions are:

- An improved version of the TARs extraction algorithm introduced in which was based on Path Join. The new version uses the better performing CMTreeMiner to mine frequent sub trees from XML documents.
- Approach validation by means of experimental results, considering both the previous and the current algorithm and showing the improvements.
- Automatic user-query transformation into "equivalent" queries over the mined intentional knowledge.
- As a formal corroboration of the accuracy of the process, the proof that our intentional-answering process is sound and complete up to a frequency threshold.

## V. TREE BASED ASSOCIATION RULES

Association rule [1] is an implication of the form X U Y, where the rule body X and head Y are subsets of the set I of items (I= {I1, I2…In}) within a set of transactions D and $X \cap Y = \emptyset$. A rule X => Y states that the transaction T that contain the items in X are likely to contain also the terms in Y. Association rules are characterized by two measures: the support, which measures the percentage of transactions in D that contain both items X and Y (X U Y); the confidence, which measures the percentage of transactions in D containing the items X that also contain the items Y (support (X U Y)/support(X)). In XML context, both D and I are collection of trees. In this work we extend the notion of association rule introduced in the context of relational

databases to adapt it to the hierarchical nature of XML documents.

Following the Infoset conventions, we represent an XML document as a tree (N, E, r, l, c) where N is the set of nodes, r□N is the root of the tree, E is the set of edges, l : N □ L is the label function which returns the tag of nodes (with L is the domain of all tags) and c : N □ C □ { □ } is the content function which returns the content of nodes(with C the domain of all contents).

We consider the element-only Infoset content model where XML non-terminal tags include only other elements and/or attributes, while the text is confined to terminal elements. We are interested in finding relationships among sub trees of XML documents. Thus, since both textual content of leaf elements and values of attributes convey "content", we do not distinguish between them.

As a consequence, for the sake of readability, we do not report the edge label and the node type label in the label in the figures. Attributes and elements are characterized by empty circles, where as the textual content of elements, or the value of attributes, is reported under the outgoing edge of the element or attribute.

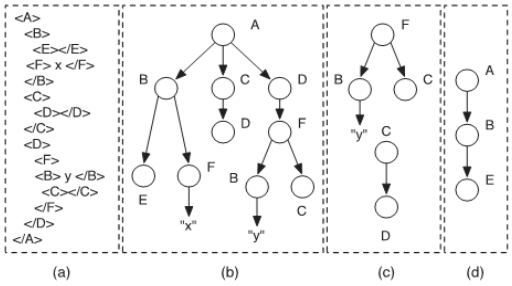



Fig. 1. (a) An example of XML document. (b) Its tree-based representation. (c) Two induced subtrees. (d) A rooted subtree.

## VI. TAR EXTRACTION

Extracting TARs [1] [11] [5] [6] through data mining is a process with two steps. In the first step frequent sub trees that satisfy given support are mined are mined. In the second step interesting rules that have confidence above given threshold are calculated from the frequent sub trees. Finding frequent sub trees is described in. Algorithm 1 finds frequent sub trees and calculates interesting rules.

**Algorithm 1.**Get –Intersting-Rules (D,minsupp minconf)

1: //frequent subtrees
2: $F_S$=FindFrequentSubtrees (D, minsupp)
3: ruleSet = Ø
4: for all s □ $F_S$do
5: // rules computed from s
6: tempSet = Compute-Rules (s, miniconf)
7: //all rules
8: ruleset = ruleSet U tempSet
9: end for
10: return ruleSet.

**Function 1 Compute-Rules(s, miniconf)**

1: ruleSet = Ø; blacklist = Ø
2: for all $C_{S,}$ subtrees of s do
3: if $C_S$is not a subtree of any element in blacklist then
4: conf = sup(s) / sup($C_S$)

5: if conf ≥ minconf then
6: newRule = ($C_S$,s,conf,supp(s))
7: ruleSet = ruleSet U {newRule}
8: else
9: blackList = blackListU $C_S$
10: end if
11: end if
12: end for
13: return ruleSet

Algorithm 1 finds frequent sub trees and then hands each of them over to a function that computes all the possible rules. Depending on the number of frequent sub trees and their cardinality, the amount of rules generated by a naïve Compute Rules function may be very high.

Given a sub tree [15] with n nodes, we could generate $2^n$-2 rules, making the algorithm exponential. This explosion occurs in the relational context too, thus, based on similar considerations, it is possible to state the following property that allows us to propose the optimized version of compute rules shown in function 2.

**Algorithm 2 Create-Index (D)**

1: for all *Di* □ *D* do
2: for all *dj* □ *Di* with *j* □ *{2, 3 . . . n}* do
3: references (root (*d*1)) = references (root (*d*1)) U references (root (*dj)*)
4: sumChildren (*d*1, *dj)*
5: end for
6: end for
7: return D

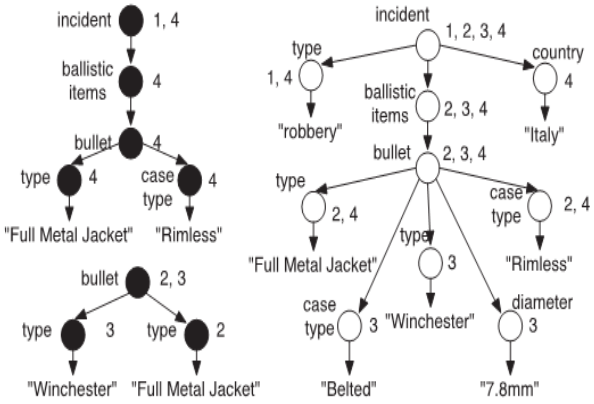
**Function 2 sumChildern(T1,T2)**

1: for all x □ children (root(T2)) do
2: if □ c □ children (root(T1)) | c= x then
3: references (root(c)) = references (root(c)) U references (root(x))
4: c = sumchildern (c,x)
5: else
6: add child (root(T1),x)
7: end if
8: end for
9: return

Before applying the algorithm, two sets A and C are constructed containing respectively the antecedent and consequent trees of all the TARs to be indexed.

## VII. INTENSIONAL ANSWERS

iTARs(intentional Tree-based Association Rules) provide an approximate intensional view of the content of an XML document, which is in general more concise than the extensional one because it describes the data in terms of its properties, and because only the properties that are verified by a high number of items are extracted. A user query over the original dataset can be automatically transformed into a query over the extracted iTARs.
The answer will be intensional, because, rather than providing the set of data satisfying the query, the system will answer with a set of properties that these data "frequently satisfy", along with support and confidence. There are two major advantages: i) querying iTARs requires less time than querying the original XML document; ii) approximate, intensional answers are in some cases more useful than the extensional ones (see the Introduction). Not all queries lend themselves to being transformed into queries on iTARs; we list three classes of queries that can be transformed by preserving the soundness; moreover, we explain how such transformation can be automatically done.




[A] Graphical Representation

```
<index>
 <antecedent>
  <bullet><ref>2</ref><ref>3</ref>
   <type> Winchester
    <ref>3</ref>
   </type></bullet>
 </antecedent>
<consequent>
 <incident>
  <ref>1</ref><ref>2</ref>
  <ref>3</ref><ref>4</ref>
  <type> robbery
   <ref>1</ref><ref>4</ref>
  </type>
 <country> Italy
 <ref>4</ref>
 </country>
  </incident>
</consequent>
</index>
```

[b] XML Document

Fig. 2 Tar index

The classes of queries that can be managed with our approach have been informally introduced in and further analysed in the relational database context in. They includes the main retrieval functionalities of XQuery, i.e. path expressions, FLOWR expressions, and the COUNT aggregate operator. We have not considered operators for adding new elements or attributes to the result of a query, because our purpose is to retrieve slender and approximate descriptions of the data satisfying the query, as opposed to modifying, or adding, new elements to the result. Moreover, since aggregate operators require an exact or approximate numeric value as answer, they do not admit intensional answers in the form of implications, thus queries containing aggregators other than COUNT are excluded. Note however that mined TARs allow us to provide *exact answers* to counting queries. The emphasized objects are meta-expressions (queries or variables) which need to be replaced in the actual query.

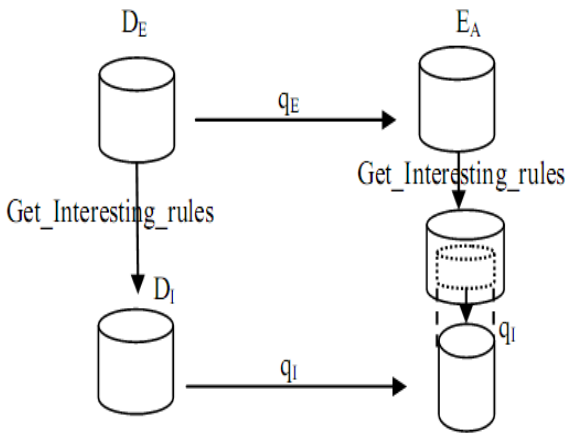



Fig. 3 Intentional query answering

## VIII. EXPERIMENTAL RESULTS

### *A. THE TREERULER PROTOTYPE*

Tree Ruler is a prototype tool that integrates all the functionalities proposed in our approach. Given an XML document, the tool is able to extract intensional knowledge and allows the user to compose traditional queries as well as queries over the intensional knowledge. Figure 1 shows the architecture of the tool. In particular, given an XML document, it is possible to extract Tree-based rules and the corresponding indexed.

The user formulates XQuery expressions on the data and these queries are automatically translated in order to be executed on the intentional knowledge.
The answer is given in terms of the set of Tree-based rules which re etc. the search criteria. It is com posed by several tabs for performing different tasks. In particular, there are three tabs:

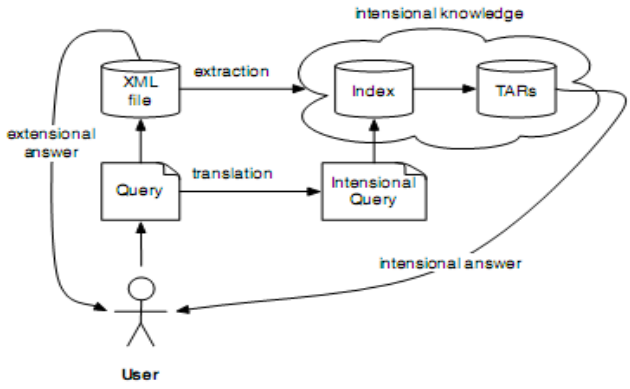



Figure 3: Tree Ruler architecture

- **Get the Gist** (Figure 4) allows in tensional information extraction from an XML document, given the desired support, confidence and the files where the Extracted tree-based rules and their index must be stored.

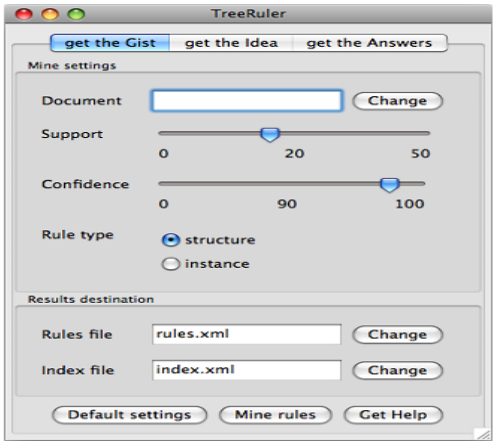



Figure 4: TreeRuler \get the Gist" tab

- **Get the Idea** allows the visualization of the in tensional information as well as the original document, in order to give the user the possibility to compare the two kinds of information.
- **Get the Answers** (Figure 5) allows to query the in-get the Answers (Figure 4) allows to query the in-tensional knowledge and the original XML document. The user has to write an extensional query in the box on the left; when the query belongs to the classes we have analyzed it is translated into the in tensional form, shown to the user in the right part of the form. Finally, once the query is executed, the Tree-based rules that re etc. the search criteria are shown in the box at the bottom of the form. Tensional knowledge and the original XML document. The user has to write an extensional query in the box on the left; when the query belongs to the classes we have analyzed it is translated into the in tensional form, shown to the user in the right part of the form. Finally, once the query is executed, the Tree-based rules thatreflect the search criteria are shown in the box at thebottom of the form.

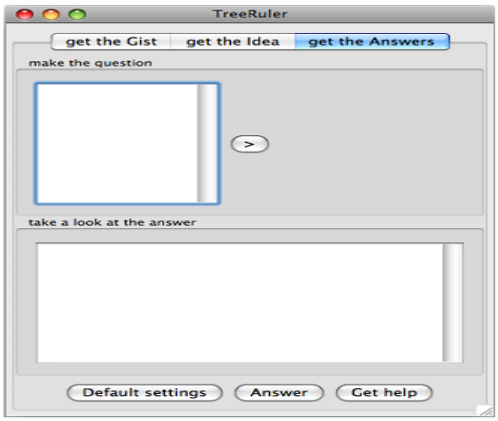



Figure 4: Tree Rulerget the Answers" tab

## IX. CONCLUSION AND FUTER WORK

The main goals we have achieved in this work are:
1) Mine all frequent association rules without imposing any apriority restriction on the structure and the content of the rules.
2) Store mined information in XML format.
3) Use extracted knowledge to gain information about the original datasets.

4) Casual users can search the data by a keyword without any data base knowledge from XML storage media. We have not discussed the updatability of both the document storing TARs and their index.

As an ongoing work, we are studying how to incrementally update mined TARs when the original XML datasets change and how to further optimize our mining efficient algorithm; moreover, for the moment we deal with a (substantial) fragment of XQuery; we would like to find the exact fragment of XQuery, which lends itself to translation into intensional queries.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," Proc. 20th Int'l Conf. Very Large Data Bases, pp. 478-499, 1994.

[2] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S.Arikawa, "Efficient Substructure Discovery from Large Semi-Structured Data," Proc. SIAM Int'l Conf. Data Mining, 2002**.**

[3] D. Barbosa, L. Mignet, and P. Veltri, "Studying the XML Web: Gathering Statistics from an XML Sample,"WorldWideWeb, vol. 8, no. 4, pp. 413-438, 2005.

[4] Y. Chi, Y. Yang, Y. Xia, and R.R. Muntz, "CMTreeMiner: Mining both Closed and Maximal Frequent Subtrees," Proc. Eighth Pacific Asia Conf. Knowledge Discovery and Data Mining, pp. 63-73, 2004.

[5] C. Combi, B. Oliboni, and R. Rossato, "Querying XML Documents by Using Association Rules," Proc. 16th Int'l Conf. Database and Expert Systems Applications, pp. 1020-1024, 2005.

[6] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy Preserving Mining of Association Rules," Proc. Eighth ACM Int'l Conf. Knowledge Discovery and Data Mining, pp. 217-228, 2002.

[7] L. Feng, T.S. Dillon, H. Weigand, and E. Chang, "An XML-Enabled Association Rule Framework," Proc. 14th Int'l Conf.Database and Expert Systems Applications, pp. 88-97, 2003.

[8] World Wide Web Consortium, XQuery 1.0: An XML Query Language, http://www.w3C.org/TR/xquery, 2007.

[9] World Wide Web Consortium, Extensible Markup Language (XML) 1.0, http://www.w3C.org/TR/REC-xml/, 1998.

[10] J.W.W. Wan and G. Dobbie, "Extracting Association Rules fromXML Documents Using XQuery," Proc. Fifth ACM Int'l Workshop Web Information and Data Management, pp. 94-97, 2003.

[11] R.Agrawal and R.Srikant. Fast algorithms for mining association rules in large databases. In Proc. of the 20th Int. Conf. on Very Large Data Bases, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.

[12] D. Barbosa, L. Mignet, and P. Veltri. Studying the xml web: Gathering statistics from an xml sample. World Wide Web, 8(4):413–438, 2005.

[13] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa.Efficient substructure discovery from large semi-structured data. In Proc.of the SIAM Int. Conf. on Data Mining, 2002.

[14] K. Wong, J. X. Yu, and N. Tang. Answering xml queries using pathbased indexes: A survey. World Wide Web, 9(3):277–299, 2006.

[15] Y. Xiao, J. F. Yao, Z. Li, and M. H. Dunham. Efficient data mining for maximal frequent subtrees. In Proc. of the 3rd IEEE Int. Conf. on Data Mining, page 379. IEEE Computer Society, 2003.