

Data Mining for XML Query-Answering Support

KC. Ravi Kumar¹, E. Krishnaveni Reddy², Ramadevi.G³
^{1, 2, 3}(CSE, Sridevi women's Engineering College, Hyderabad, Andhra Pradesh)

Abstract: XML has become a defacto standard for storing, sharing and exchanging information across heterogeneous platforms. The XML content is growing day by day in rapid pace. Enterprises need to make queries on XML databases frequently. As huge XML data is available, it is challenging task to extract required data from XML database. It is computationally expensive to answer queries without any support. Towards this, in this paper we present a technique known as Tree-based Association Rules (TARs) mined rules that provide required information on structure and content of XML file and the TARs are also stored in XML format. The mined knowledge (TARs) used later for XML query answering support. This enables quick and accurate answering. We also developed a prototype application to demonstrate the efficiency of the proposed system. The empirical results are very positive and query answering is expected to be useful in real time applications.

Index Terms: XML, query answering support, data mining, tree-based association rules

I. Introduction

XML has become a popular format for storing and sharing data across heterogeneous platforms. The XML format is neutral, flexible and interoperable [5]. 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 [4] that works on the search process based on the given search word. With respect to query answering, 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. However, schema presence is not mandatory to process XML file [3].

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. Proposed Framework

The proposed XML query answering support framework is as shown in fig. 1. The purpose of this framework is to perform data mining on XML and obtain intentional knowledge. The intentional knowledge is also in the form of XML. This is nothing but rules with supports and confidence. In other words the result of data mining is TARs (Tree-based Association Rules).

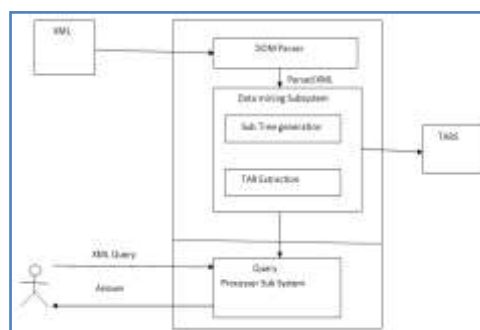


Fig. 1 – Proposed XML query answering support framework

As can be seen in fig. 1, the framework is to have data mining for XML query answering support. When XML file is given as input, DOM parser will parse it for wellformedness and validness. If the given XML document is valid, it is parsed and loaded into a DOM object which can be navigated easily. The parsed XML file is given to data mining sub system which is responsible for sub tree generation and also TAR extraction.

The generated TARs are used by Query Processor Sub System. This module takes XML query from end user and makes use of mined knowledge to answer the query quickly.

Tar Extraction

Extracting TARs through data mining is a process with two steps. In the first step frequent subtrees that satisfy given support are mined. In the second step interesting rules that have confidence above given threshold are calculated from the frequent subtrees. Finding frequent sub trees is described in [1], [2], [6], [7], [8], [9]. Algorithm 1 finds frequent sub trees and calculates interesting rules.

Algorithm 1 Get-Interesting-Rules ($D, \text{minsupp}, \text{minconf}$)

```

1: // frequent subtrees
2:  $F_S = \text{FindFrequentSubtrees}(D, \text{minsupp})$ 
3: ruleSet =  $\emptyset$ 
4: for all  $s \in F_S$  do
5:   // rules computed from s
6:   tempSet = Compute-Rules( $s, \text{minconf}$ )
7:   // all rules
8:   ruleSet = ruleSet  $\cup$  tempSet
9: end for
10: return ruleSet

```

Function 2 Compute-Rules ($s, \text{minconf}$)

```

1: ruleSet =  $\emptyset$ ; blackList =  $\emptyset$ 
2: for all  $c_s$ , subtrees of  $s$  do
3:   if  $c_s$  is not a subtree of any element in blackList then
4:      $\text{conf} = \text{supp}(s) / \text{supp}(c_s)$ 
5:     if  $\text{conf} \geq \text{minconf}$  then
6:       newRule =  $\langle c_s, s, \text{conf}, \text{supp}(s) \rangle$ 
7:       ruleSet = ruleSet  $\cup$  {newRule}
8:     else
9:       blackList = blackList  $\cup$   $c_s$ 
10:    end if
11:  end if
12: end for
13: return ruleSet

```

The rules obtained from algorithm 1 are written to an XML file. Then indexing is made. Afterwards when XML queries are made, the proposed system uses index and TARs and quickly answers the query.

III. Experiments And Results

Environment

The environment used to develop the prototype application includes JSE (Java Standard Edition) 6.0, Net Beans IDE that run in Windows 7 OS. A PC with 2 GB RAM and 2.9x GHz processor is used. The Java SWING API is used to build graphical user interface while IO and JAXP (Java API for XML Parsing) are used for implementing functionality. The main application GUI is as shown in fig.



Fig. 2 – The GUI of the prototype application

As can be seen the GUI has provision to choose an XML file as input. It also allows choosing a file for storing extracted rules. A text area is provided to show the XML file content. View Tree button shows the XML file with graphical tree representation. Generate Large ItemSet button generates frequent sub trees that will be used for further processing. On clicking the GenerateRuleFile button, it extracts TARs from the given XML file and finally extracted rules are saved into the given TAR file. The QueryAnswering button invokes a form where user can enter queries. The query interface is shown in fig. 3.

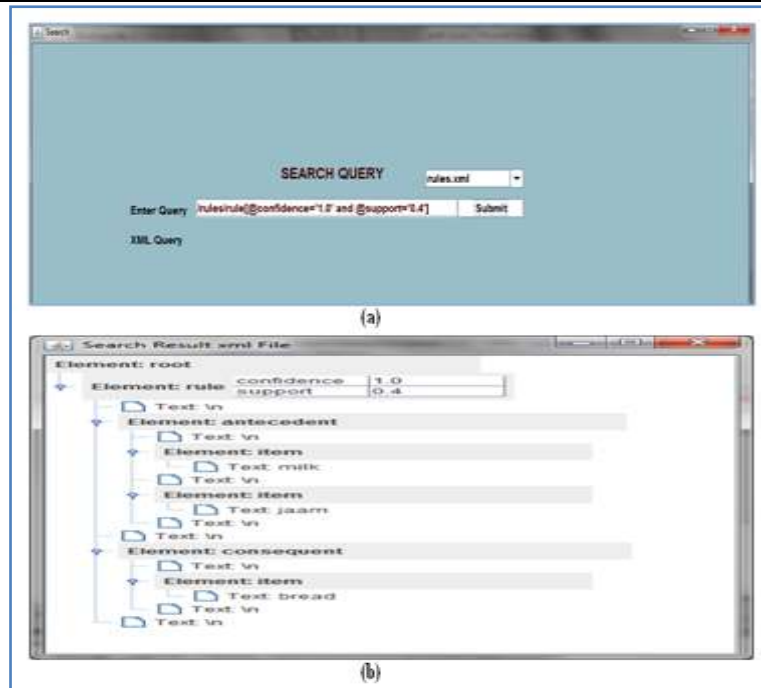


Fig. 3 – Query interface and results

As can be seen the fig. 3 (a) shows interface for making queries. The queries given here are processed faster using rule files extracted from XML files. The result of given query in fig. 3 (a) is shown in fig. 3 (b) with the results that satisfied given support and confidence.

IV. Results

We have performed four types of experiments. They are based on time required to extract intentional knowledge from XML; time required to answer intentional and extensional queries; monitoring extraction time with given support and confidence; and study of accuracy of intentional answers.

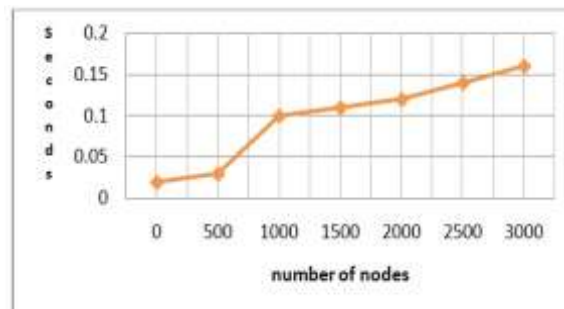


Fig. 4 –Extraction time with respect to number of nodes

As can be seen in fig. 4, the TAR extraction time is more when number of nodes in XML document is more. In other words, the time taken to extract TARs is directly proportional to the number of nodes in given XML document.

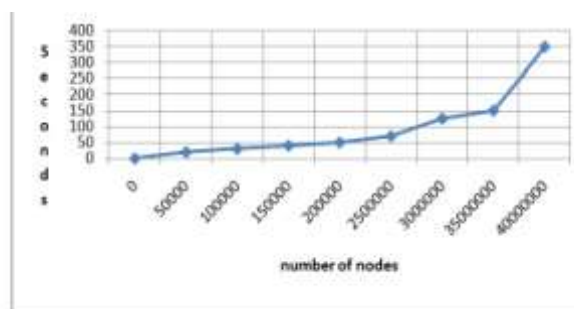


Fig. 5 – Extraction time with respect to number of nodes in XMark generated XML documents

As can be seen in fig. 5, the TAR extraction time is more when number of nodes in XML document is more. In other words, the time taken to extract TARs is directly proportional to the number of nodes in given XML document which is generated using XMark.

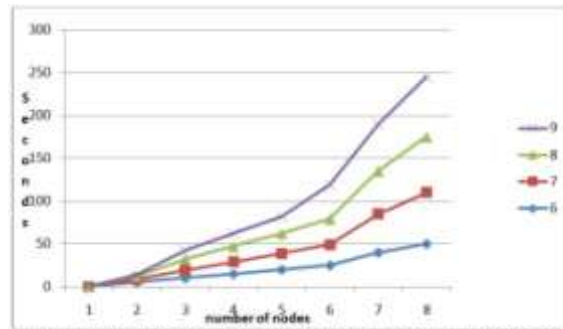


Fig. 6 – Extraction time with respect to number of nodes in document with fixed depth

As can be seen in fig. 6, the TAR extraction time is more when number of nodes in XML document is more. In other words, the time taken to extract TARs is directly proportional to the number of nodes in given XML document with fixed depth.

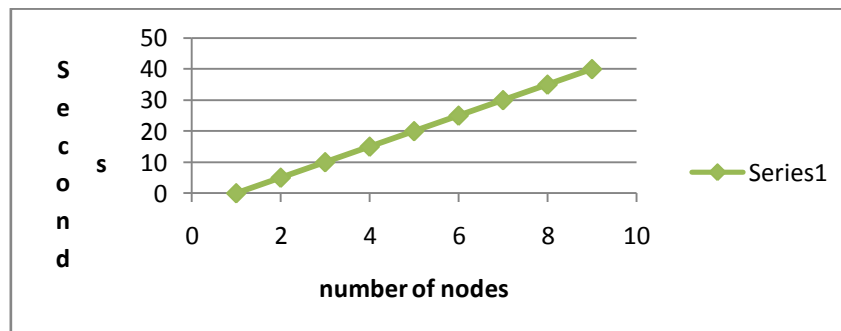


Fig. 7 – Extraction time growth using CMTreeMiner with respect to number of nodes

As can be seen in fig. 7, the TAR extraction time is more when number of nodes in XML document is more. In other words, the time taken to extract TARs is directly proportional to the number of nodes in given XML document when CMTreeMiner is used.

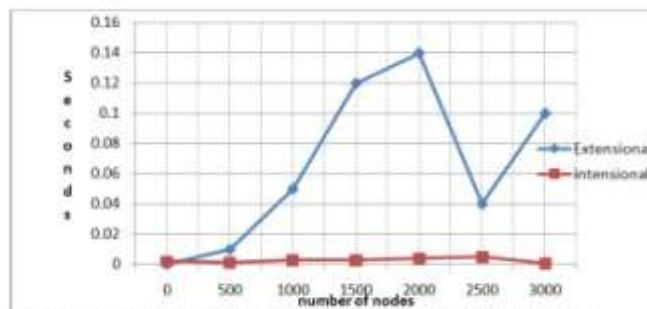


Fig. 8 – Extensional and intentional time answering with respect to real XML documents

As can be seen in fig. 8, the time taken for intentional and extensional query answering are plotted. However, the intentional query answering takes very less time when compared with that of extensional answering.




V. Conclusion

In this paper we presented a framework for extracting TARs from given XML file so as to support XML queries. Towards this end, the aim of this paper is to mine frequent association rules and store the mined content in XML format; use the TARs to support query answering or to gain information from XML databases. A prototype application is built to test the efficiency of the proposed framework. The application takes XML file as input and generates TARs and then finally index file that helps in query processing. The experimental results revealed that the proposed application is useful and can be used in real time applications.

References

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the 20th Int. Conf. on Very Large DataBases*, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.
- [2] T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering frequent substructures in large unordered trees. In *Technical Report DOI-TR216, Department of Informatics, Kyushu University*. <http://www.i.kyushuu.ac.jp/doitr/trcs216.pdf>, 2003.
- [3] 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.
- [4] Gary Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.
- [5] World Wide Web Consortium. Extensible Markup Language (XML) 1.0, 1998. <http://www.w3c.org/TR/REC-xml/>.
- [6] K. Wang and H. Liu. Discovering typical structures of documents: a road map approach. In *Proc. of the 21st Int. Conf. on Research and Development in Information Retrieval*, pages 146–154, 1998.
- [7] 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.
- [8] X. Yan and J. Han. Closegraph: mining closed frequent graph patterns. In *Proc. of the 9th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 286–295. ACM Press, 2003.
- [9] M. J. Zaki. Efficiently mining frequent trees in a forest: algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1021–1035, 2005. Mirjana

About Authors:

	<p>Mr.K.C Ravi Kumar M.Tech CSE from JNTU Hderabad currently he is the head of department for M.Tech CSE programme in Sridevi Women’s Engineering College having 17 years of Academic Experience. He is life member of IEEE & IST areas of research include Data Mining & Data Warehousing Information Retrival Systems Information Security.</p>
	<p>Mrs. E. Krishnaveni Reddy B.Tech(C.S.E), M.Tech(S.E) Asst.Prof</p>
	<p>Ms. Ramadevi. G, B.E in C.S.E from Muffakham Jah College of Engineering and technology M.Tech in C.S.E from Sridevi women’s Engineering college</p>