International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 6, December 2012 www.ijcsn.org ISSN 2277-5420

Extracting TARs from XML for Efficient Query Answering

[1]Chandra Sekhar.K, [2]Dhanasree

[1] Dept of CSE, JNTU H, DRK Institute of Science and Technology
Hyderabad, Andhra Pradesh, India

[2] Dept of CSE, JNTU H, DRK Institute of Science and Technology
Hyderabad, Andhra Pradesh, India

Abstract

The massive amount of datasets expressed in different formats, such as relational, XML, and RDF, avail-able in several real applications, may cause some difficulties to non-expert users trying to access these datasets without having sufficient knowledge on their content and structure. Moreover, the processes of query composition, especially in the absence of a schema, and interpretation of the obtained answers may be non-trivial. The existing data mining process is often guided by the designer, who determines the portion of a dataset where useful patterns can be extracted based on his/her deep knowledge of the application scenario. In this paper, we propose efficient mining techniques to mine hidden information from huge datasets, and then use it in order to gain useful knowledge which helps inexperienced users to access huge XML datasets. We also describe XML mining tool which implemented using Java encompasses two main features 1) it mines all the frequent association rules from input documents without any a-priori specification of the desired results 2) it provides quick, summarized, thus often approximate answers to user's queries, by using the previously mined knowledge.

Keywords: *XML Association Rules, Keyword search, Approximate answering, XML mining.*

1. Introduction

One of the trickiest problems of finding information in the context of large XML datasets is reaching fast and concise answering capabilities. Inexperienced users need the support of a knowledge discovery system able to search, retrieve information from huge XML datasets. Data mining techniques offer a privileged way to deal with the information overload problem by extracting frequent patterns and providing intensional, often approximate, information both about the content and the structure of a document. An intensional representation of a dataset is a set of patterns (e.g., association rules, clusters, etc.) describing the most relevant properties of the dataset. Intensional information is thus a summarized representation of the original document, which means that less space is required to store it and less time is required to query it.Together with intrinsically unstructured documents, there is a significant portion of XML documents which have only an implicit structure, that is, their structure has not been declared in advance, for example via a DTD or an XML-Schema [1]. Querying such documents is quite difficult for users for two main reasons: 1) they are not able to specify a reasonably probable structure in the query conditions and 2) they are very often confused by the large amount of information available. The extraction of intensional information through the use of data mining techniques has been proposed in the literature, both with respect to the relational model [2] and to the XML format [3]. However, while in the relational context a lot of algorithms [4] [5] and tools [6] have been proposed, the literature about this topic is not as rich in the XML context. Major difficulties consist in the fact that XML is more expressive than the relational format and allows to represent both the structure and content of information in a different (i.e., hierarchical) way. Such novelty has made it difficult to give a generally accepted definition of how an association rule or a cluster should look like in the XML context. The literature presented in this paper addresses the problems of: (1) extracting intensional information from XML datasets without guiding the mining process, (2) representing it by means of appropriate association rules, and (3) allowing users to use such information in the query-answering process. We represent intensional knowledge in native XML as TARs (Tree-based Association Rules).A TAR represents intensional knowledge in the form SB $\Rightarrow$ SH, where SB is the body tree and SH the head tree of the rule and SB is a sub tree of SHA TAR may state that, given an XML document and a node n labeled book, in 75% of the cases,

if n/genere='Computer'
then
n/catalog/book/author='Chandra'.

That is, 75% of Computer books written by Chandra. Notice that this simple rule describes the co-relation between two trees, thus, it contains information both on frequent content values and on the exact structure (i.e., the paths) of these values inside the mined document. Note that it is also possible to mine TARs that describe only structural information (i.e., without PCDATA). Therefore, mined TARs offer a summarized, approximate view of the content as well as the structure of the original XML document. Once TARs have been mined and stored, XML Mining tool accepts user queries, directed to the original document, which are

automatically translated into queries that can be executed over the extracted TARs. The intensional answer provided by XML Mining tool is the set of TARs satisfying the user request.

The intensional information stored in the TARs provides a valid support in several cases:

1) It allows obtaining and storing implicit knowledge of the documents, useful in many respects:

(i) to get a concise idea – the gist – of both the structure and the content of an XML document quickly without knowing its internal structure.

(ii) TARs represent a data guide that helps users to be more effective in query formulation.

(iii) Frequent patterns allow discovering hidden integrity constraints that can be used for semantic optimization (iv) for privacy reasons, a document answer might expose a controlled set of TARs instead of the original document, as a summarized view that masks sensitive details like passwords, back account numbers

2) TARs can be queried to obtain fast, although approximate, answers. This is particularly useful not only when quick answers are needed but also when the original documents are unavailable. In fact, once extracted, TARs can be stored in a (smaller) document and be accessed independently of the dataset they were extracted from.

2. Structure of the paper

The paper is organized as follows. Section I defines tree-based association rules (TARs) and introduces their usage, while Section III presents proposed framework. Section IV presents how these rules are extracted from XML documents. Section V describes a prototype that implements our proposal and how they are used to respond to intensional queries. Section VI presents results with different data and VII at last, states the possible follow-ups to this work draws the conclusions.

3. 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).

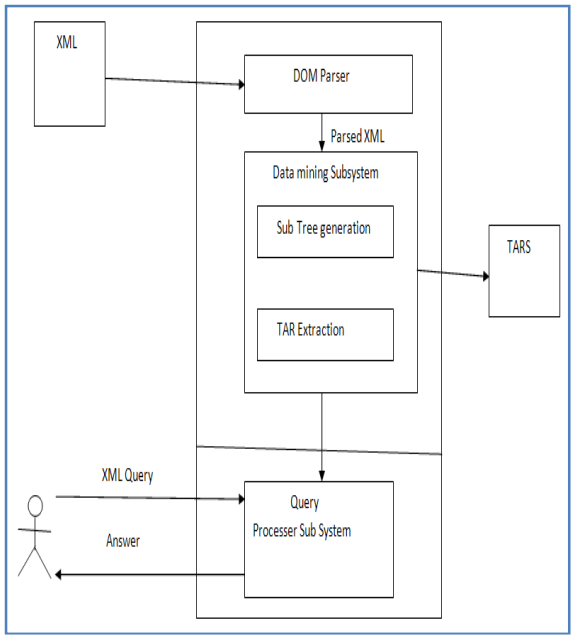

Fig. 1 – Proposed XML query answering support framework

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.

4. 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 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 [7], [8], [9], [10], [11], and [12]. Algorithm 1 finds frequent sub trees and calculates interesting rules.

Algorithm 1 Get-Interesting-Rules $(D, minsupp, minconf)$

```
1:  // frequent subtrees
2:  F_S = FindFrequentSubtrees (D, minsupp)
3:  ruleSet = ∅
4:  for all s ∈ F_S do
5:    // rules computed from s
6:    tempSet = Compute-Rules(s, minconf)
7:    // all rules
8:    ruleSet = ruleSet ∪ tempSet
9:  end for
10: return ruleSet
```

Function 2 Compute-Rules $(s, minconf)$

```
1:  ruleSet = ∅; blackList = ∅
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 = supp(s) / supp(c_s)
5:      if conf ≥ minconf then
6:        newRule = ⟨c_s, s, conf, supp(s)⟩
7:        ruleSet = ruleSet ∪ {newRule}
8:      else
9:        blackList = blackList ∪ 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.

5. Experiments and Results

5.1 Environment

The environment used to develop the prototype application includes JSE (Java Standard Edition) 6.0, JDeveloper 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 external libraries stax (Streaming API for XML) and saxon (XSLT and XQuery processor) used for XML processing and XQuery functions processing. The library log4j used for logging the execution times and messages. The java library JSysmon used for accessing system monitoring information like CPU or Memory Usage. The CMTreeMiner execution binary file is used to generate the frequent sub trees. The main application GUI is as shown in fig.

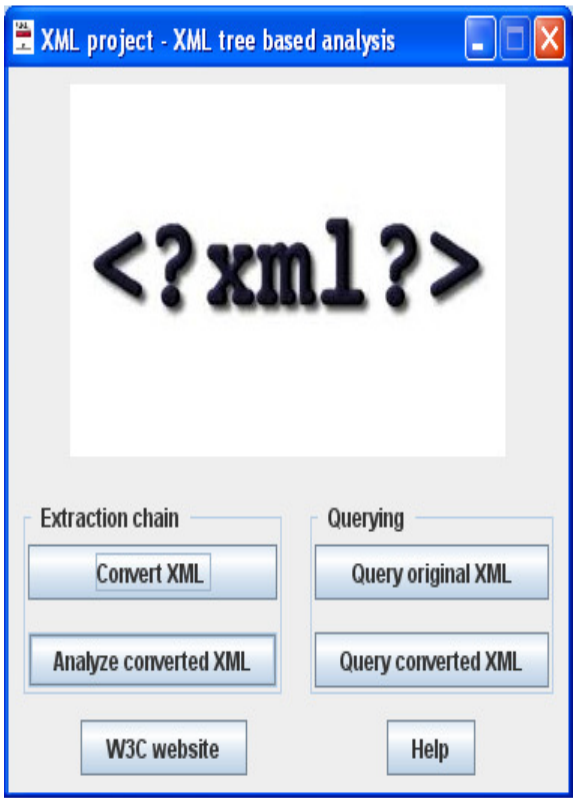

Fig. 2 – The GUI of the prototype application

As can be seen the GUI has provision to choose an XML file and convert it. It also allows querying original XML; analyze converted XML, and querying on converted XML. The XML analysis window is shown in fig. 3.

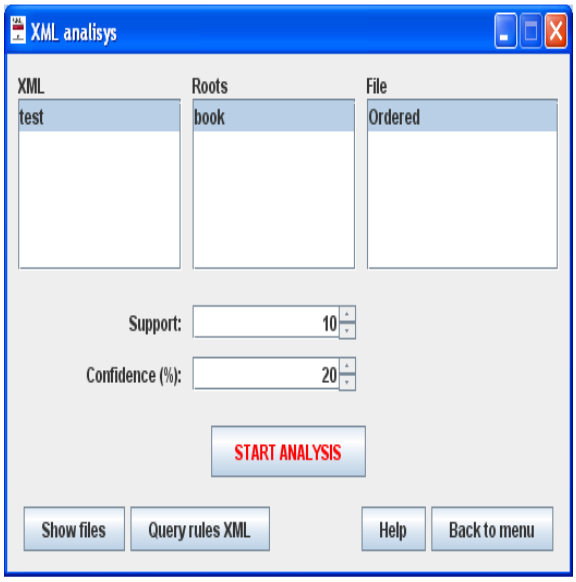

Fig. 3 – XML Analysis

As can be seen the fig. 3 (a) shows interface for making XML analysis. The process starts when user clicks "start

analysis" button. The analysis is based on the input XML file, the content of the file. The given support and confidence are considered while making the analysis. Fig. 4 shows query rules XML.

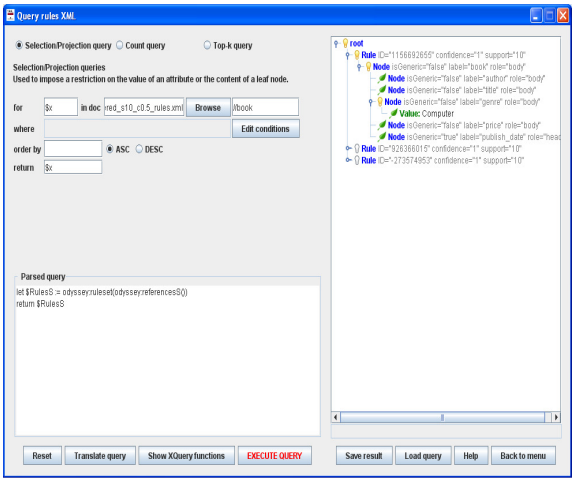

Fig. 4 Querying rules XML

As can be seen in fig. 4, the rules XML file is shown and it can be queried with various types of queries such as selection projection query, count query and Top – k query. It also supports various other clauses such as where, order by and returns besides specifying sort order.

6. 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.

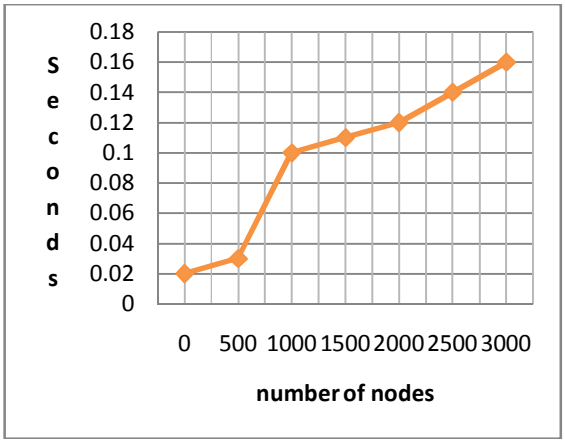

Fig. 4 – Extraction time with respect to number of nodes

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.

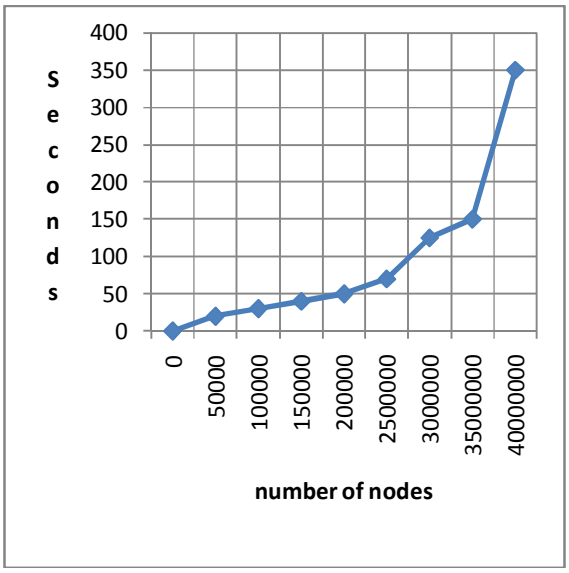

Fig. 5 – Extraction time with respect to number of nodes in XMark generated XML documents

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.

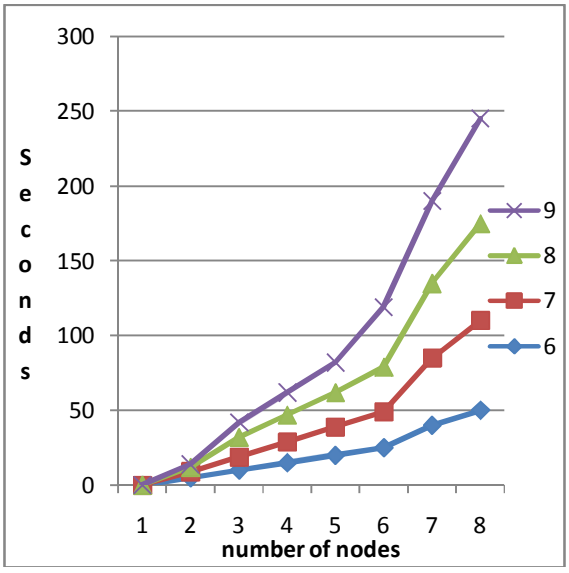

Fig. 6 – Extraction time with respect to number of nodes in document with fixed depth

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.

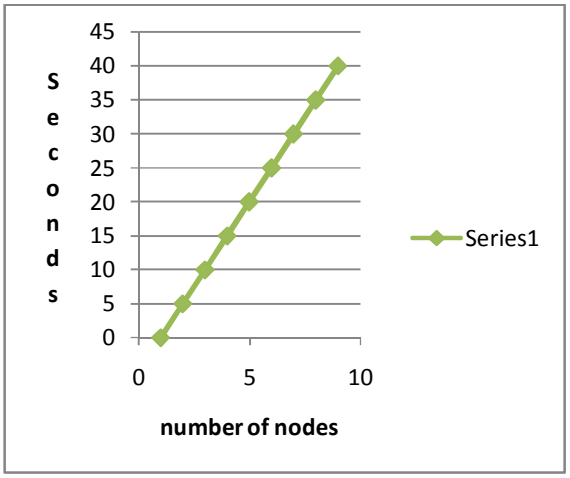

Fig. 7 – Extraction time growth using CMTreeMiner with respect to number of nodes

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.

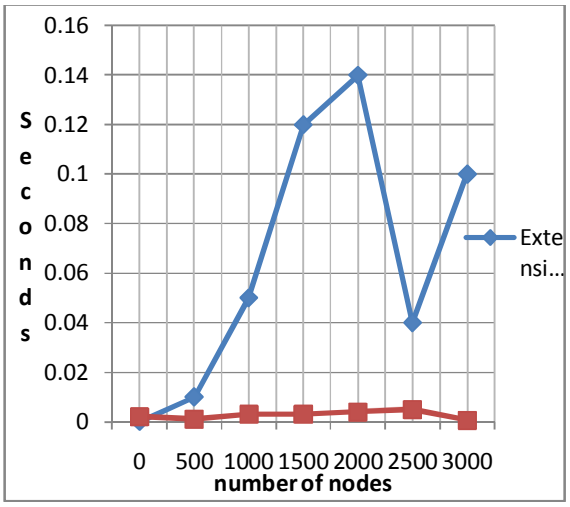

Fig. 8 – Extensional and intentional time answering with respect to real XML documents

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

7. 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] S. Gasparini and E. Quintarelli. Intensional query answering to xquery expressions. In *Proc. of the 16th Int. Conf. on Database and Expert Systems Applications*, pages 544–553, 2005.

[2] B. Goethals and M. J. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *SIGKDD Explorations*, 6(1):109– 117, 2004.

[3] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proc. of the 23rd Int. Conf. on Very Large Data Bases*, pages 436–445, 1997.

[4] R. Goldman and J. Widom. Approximate DataGuides. In *Proc. Of the Workshop on Query Processing for Semistructured Data and Non- Standard Data Formats*, pages 436–445, 1999.

[5] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321– 354, 2003.

[6] A. Jim´enez, F. Berzal, and J. C. Cubero. Mining induced and embedded subtrees in ordered, unordered, and partially-ordered trees. In *Proc. Of the 17th Int. Symposium on Methodologies for Intelligent Systems*, pages 111–120, 2008.

[7] Yogesh R.Rochlani , Prof. A.R. Itkikar, "Integrating Heterogeneous Data Sources Using XML Mediator", ijcsn, vol 1, issue 3, 2012.

[8] T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering frequent substructures in large unordered trees. In *Technical Report DOI-TR 216, Department of Informatics, Kyushu University. http://www.i.kyushuu. ac.jp/doitr/trcs216.pdf*, 2003.

[9] 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.

[10] 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.

[11] 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.

[12] 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

[13] 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.