A Survey on Tree based Association Rules (TARs) from XML Documents

 # Naresh Vurukonda[1], Asst.Professor,CSE department, E mail: naresh.vurukonda@gmail.com
#G. Ranadheer[2] Reddy, HOD, CSE department , E mail: ranadheer0905@gmail.com
#B Mounika[3], G.Yogyatha[4], N.Srujana[5], P. Krishna Priya[6] , B.Tech(CSE)
 # Sumathi Reddy Institute of Technology for women, Warangal, A.P, INDIA

Abstract*—*The increasing amount of XML datasets available to users increases the necessity of investigating techniques to extract knowledge from these data. Data mining is widely applied in the database research area in order to extract frequent correlations of values from both structured and semi-structured datasets. In this work we describe an approach to mine Tree-based association rules from XML documents. Such rules provide information on both the structure and the content of XML documents; moreover, they can be stored in XML format to be queried later on. The mined knowledge is approximate, intentional knowledge used to provide: (i) Quick, approximate answers to queries and (ii) Information about structural regularities that can be used as data guides for document querying. A prototype of the proposed system is also briefly described.

1. Introduction

Association rules describe the co-occurrence of data items in a large amount of collected data and are represented as implications of the form X ⇒Y , where X and Y are two arbitrary sets of data items, such that X ∩ Y = ∅. The quality of an association rule is measured by means of support and confidence. Support corresponds to the frequency of the set X ∪ Y in the dataset, while confidence corresponds to the conditional probability of finding Y , having found X and is given by supp(X ∪ Y)/supp (X). Here we extend the notion of association rule introduced in the context of relational databases to adapt it to the hierarchical nature of XML documents.

1.1. Fundamental concepts

Given an XML document, we extract two types of TARs: • A TAR is a structure TAR (sTAR) iff, for each node
N contained in SH , cH (n)= ⊥ , that is, no data value is present in sTARs, i.e. they provide information only on the structure of the document.
• A TAR,SB ⇒SH ,is an instance TAR (iTAR) iff SH contains at least one node n such that cH (n) = ⊥ , that is, iTARs provide information both on the structure andon the data values contained in a document. Since TARs provide an approximate view of both the content and the structure of an XML document, (1) sTARscan be used as an approximate Data Guide of the original document, to help users formulate queries; (2)iTARscan be used to provide intentional, approximate answers to user queries. By observing s TARs users can guess the structure of an XML document, and thus use this approximate schema to formulate a query when no DTD or schema is available: as Data Guides, sTARs represent a concise structural summary of XML documents. Differently from Data Guides, sTARs do not show all possible paths in the XML document but only the frequent paths. In particular, for each fragment, its support determines how frequent the substructure is. This means that sTARs provide a simple path index which supports path matching and can be used for the optimization of the query process. An index for an XML dataset is a pre-defined structure whose performances maximized when the query matches exactly the designed structure. Therefore, the goal, when designing an index, is to make it as similar as possible to the most frequent queries.By contrast, iTARs give an idea about the type of content of the different nodes.

2. Tar Extraction

TAR mining is a process composed of two steps: 1) mining frequent sub trees , that is, sub trees with a support
above a user-defined threshold, from the XML document; 2)computing interesting rules, that is, rules with a confidence above a user-defined threshold , from the frequent sub trees.Once the mining process has finished and frequent TARs have been extracted, they are stored in XML format. This decision has been taken to allow the use of the same language (XQuery in our case) for querying both the original dataset and the mined rules. One of the (obvious) reasons for using TARs instead of the original document is that processing iTARs for query answering is faster than processing the document. To take full advantage of this, we introduce indexes on TARs to further speed up the access to mined trees – and in general of intentional query answering. In the literature the problem of making XML query-answering faster by means of path-based indexes has been investigated. In general, path indexes are proposed to quickly answer queries that follow some frequent path template, and are built by indexing only those paths having highly frequent queries. We start from a different perspective: we want to provide a quick, and often approximate, answer also to casual queries.

3. Intentional procedures

iTARs provide an approximate intentional view of the content of an XML document, which is in general more concise than the extensional one because it describes the datain 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 intentional, 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, intentional answers are in some cases more useful than the Extensional ones. 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. The classes of queries that can be managed with our approach have been informally introduced and fur-the analyzed in the relational database context. They include the main retrieval functionalities of X Query, 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 intentional 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 vari-ables) which need to be replaced in the actual query.

• Class 1: σ /π -queries . Used to impose a simple, or complex (containing AND and OR operators), restriction on the value of an attribute or the content of a leaf node, possibly ordering the result. The query imposes some conditions on a node's content and on the content of its descendants, orders the results according to one of them and returns the node itself.

• Class 2: count-queries . Used to count the number of elements having a specific content. The query creates a set containing the elements which satisfy the conditions and then returns the number of elements in such set.

• Class 3: top-k queries. Used to select the best k answers satisfying a counting and grouping condition. The query counts the occurrences of each distinct value of a variable in a desired set; then

orders the variables with respect to their occurrences and returns the most frequent k .

4. The Tree Ruler prototype

Tree Ruler is a tool that integrates the functionalities proposed in our approach. Given an XML document, it Enables users to extract intentional knowledge and compose tradtional queries as well as queries over the intentional knowledge, receiving both extensional and intentional answers. Users formulate X Queries over the original data, and queries are automatically translated and executed on the intentional knowledge. The answer is given in terms of the set of TARs which reflect the search criteria. Tree Ruler interface offers three tabs:

• get the Gist allows intentional information extraction from an XML document, given the support, confidence and the files where the extracted TARs and their indexare to be stored.

• get the Idea allows to show the intentional information as well as the original document, to give users the possibility to compare the two kinds of information.

• get the Answers allows to query the intentional knowledge and the original XML document. Users have to write an extensional query; when the query belongs to the classes we have analyzed it is translated and applied to the intentional knowledge. Finally, once it is executed, the TARs that reflect the search criteria are shown. Tree Ruler is implemented in C++ using the eXpat library for XML parsing, and wx Widgets tools for the GUI.The tool implements the CMT Tree Miner.

4.1 algorithm for the extraction of frequent sub trees from the XML document.

Four types of experiments are performed : 1) time required for the extraction of the intentional knowledge from an XML database; 2) time needed to answer intentional and extensional queries over an XML file; 3) a use case scenario on the Doc Book XML database, in order to monitor extraction time given a specific support or confidence; 4) a study of the accuracy of intentional answers.Our experiments are performed on real XML datasets and cover all three classes of queries introduced in our proposal.

5. Related Work

The problem of association rule mining was initially proposed in Agrawal (R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases) and successively many implementations of the algorithms, downloadable from B.Goethals and M.J.Zaki. Advances in frequent item set mining , were developed and described in the database literature, Weka 1 being a known framework. More recently the problem has been investigated also in the XML context “Discovering interesting information in xml data with association rules”, “Extracting association rules from xml documents using XQuery” and “A new method for mining association rules from a collection of xml documents”. In “Discovering interesting information in xml data with association rules” the authors use XQuery (http://www.w3C.org/TR/xquery) to extract association rules from simple XML documents.

They propose a set of functions written only in XQuery which implement together the Apriori algorithm. It is show that their approach performs well on simple XML documents; however it is very difficult to apply this proposal to complex XML documents with an irregular structure. This limitation has been overcome in “Extracting association rules from xml documents using XQuery”, where the authors introduce a proposal to enrich XQuery with data mining and knowledge discovery capabilities, by introducing XMINE RULE, a specific operator for mining association rules for native XML documents.

They formalize the syntax and an intuitive semantics for the operator and propose some examples of complex association rules. However, the operator proposed uses the MINE RULE operator, which works on relational data only. This means that, after a step of pruning of unnecessary information, the XML document is translated into the relational format. Moreover, both “Discovering interesting information in xml data with association rules” and “Extracting association rules from xml documents using XQuery” force the designer to specify the structure of the rule to be extracted and then to mine it, if possible. This means that the designer has to specify what should be contained in

the body and head of the rule, i.e. the designer has to know the structure of the XML document in advance, and this is an unreasonable requirement when the document has not an explicit DTD. Another limitation of these approaches is that the extracted rules have a fixed root, thus once the root node of the rules to mine has been fixed, only its descendants are analyzed. Let us consider the dataset in Figure 1 to explain this consideration.

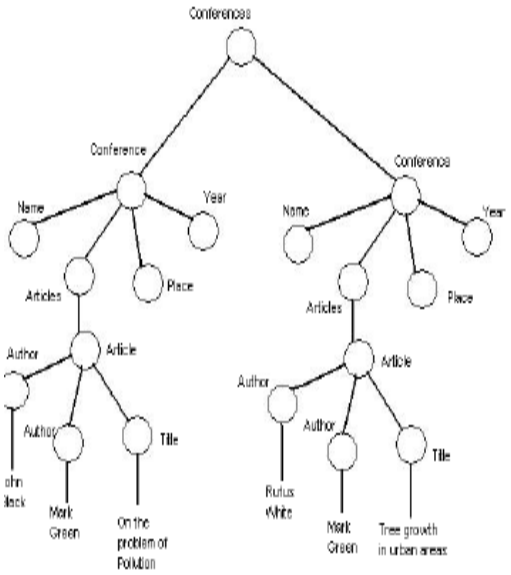

Figure 1. XML sample file: "conferences.xml"

In order to infer the co-author relationship among authors of conferences it is necessary to x the root node of the rules in the article element, the body and head in author.

In such way it is possible to learn that “John Black” and “Mark Green” frequently write papers together. However, it is not possible to mine item sets stating that frequently, during “2008” conferences have been held in “Milan”. Indeed, to mine such property the body of the rules should be fixed in the year element, which is not contained in the sub-tree of the article node, and the head in place. Our idea is to take a more general approach to the problem of extracting association rules from XML documents, i.e. to mine all frequent rules, without having any a-priori knowledge of the XML dataset. A similar idea was presented in “A new method for mining association rules from a collection of xml documents” where the authors introduced HoPS, an algorithm for extracting association rules in a set of XML documents.

Such rules are called XML association rules and are implications of the form X. Y, where X and Y are fragments of an XML document. In particular the two trees X and Y have to be disjunct. The limitation of this proposal is that it does not contemplate the possibility to mine general association rules within a single XML dataset, while achieving this feature is one of our goals. The idea of using association rules as summarized representations of XML documents was also introduced where the XML summary is based on the extraction of association rules both on the structure (schema patterns) and on content values (instance patterns) from XML datasets.

The limitation of such an approach is that the so-called schema patterns, used to describe general properties of the schema applying to all instances, are not mined, but derived as an abstraction of similar instance patterns. In our work, XML association rules are mined starting from frequent sub trees of the tree-based representation of a document. In the database literature it is possible to and many proposals of algorithms to extract frequent structures from tree/graph-based data structures. Just to cite some of them, Tree Miner, Path Join, Close Graph propose algorithms to directly mine frequent item sets not association rules-from XML documents. Tree Miner and Close Graph do not preserve the exact structure of the item sets extracted -only the "descendant-of” (and not the "child-of") relationship between nodes is preserved -whereas Path Join does. In this work we propose an algorithm that extends Path Join to mine generic tree-based association rules directly from XML documents.

6.The Tree ruler 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 intentional knowledge, and allows the user to compose traditional queries as well as queries over the intentional knowledge. Figure 2 shows the

architecture of the tool. In particular, given an XML document, it is possible to extract Tree-based rules and the corresponding index le. 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 reflect the search criteria.

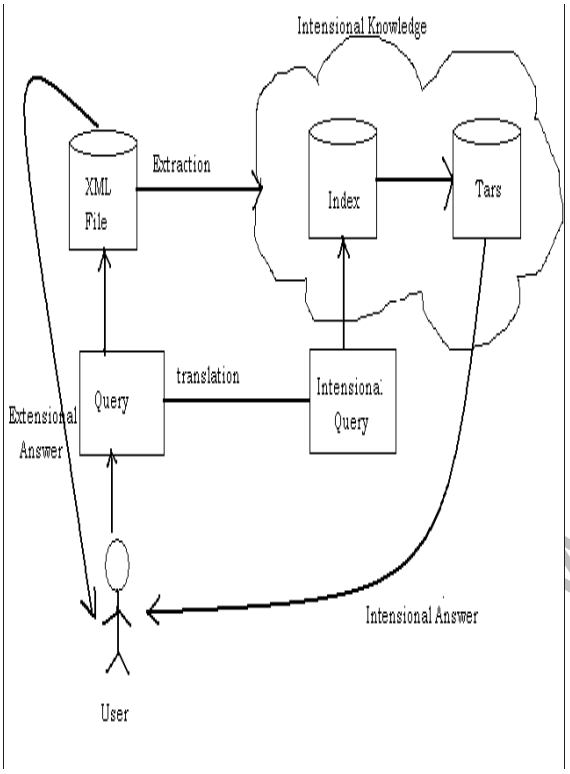

Figure 2 Tree Ruler Architecture

A screenshot of the tool is shown in Figure 3: it is composed by several tabs for performing different tasks. In particular, there are three tabs: • Get the Gist (Figure 3) allows intensional information extraction from an XML document, given the desired support, confidence and the les where the extracted tree-based rules and their index must be stored. • Get the Idea allows the visualization of the intentional 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 3) allows to query the intentional 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 intentional form, shown to the user in the right part of the form.

Finally, once the query is executed, the Tree-based rules that reflect the search criteria are shown in the box at the bottom of the form

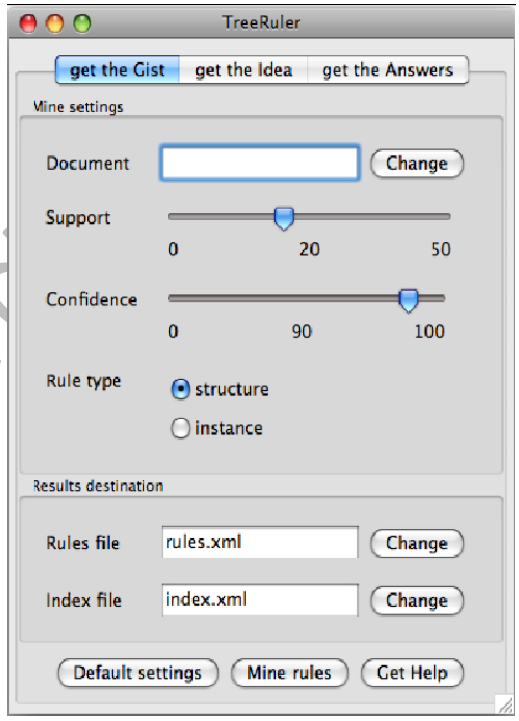

Figure 3 TreeRuler Tool

6.1. The Sedna Tool Prototype

Sedna is a powerful, open source, native XML Database, written from the ground up in C/C++ by Team MODIS. The team has developed and is continuing to develop a XML Database which is starting to seriously boast the functionality and performance of mature relational databases such as My SQL and thus can be taken very seriously by

application developers for production grade applications. Virtually all other XML Databases are written in Java™, also the majority if not all of those provide a network based API which works upon SOAP, XML-RPC, REST or some other bloated protocol. Sedna's network protocol is completely binary based.

7.Features

1.Stands up to immense usage stress, built-in Database Connection Pooling manager. 2.Allows XML documents to be streamed to Sedna directly from http:// and ftp: // locations. 3.Zero dependencies. Other than xmldb.jar interface APIs this package requires nothing other than a JVM. 4.Sedna supports the XUpdate standard for updating data. 5. Sedna can support Binary BLOBS as well as Java™ Object storage. 6.Sedna is hierarchical collections friendly. 7. Extensible, supports custom XML: DB Service plug-in on XML: DB Collections. 8.Makes full use of Sedna's ACID Transactions capability. Manual/Auto Commit/Rollback 100% supported. 9.Very small memory footprint and very fast execution, the server carries the burden where-ever possible.
10.Via this API, Sedna can now execute an XQuery/XPath directly against a resource or a collection.
11.All XML processing is 100% JAXP based.
12.Meets all the requirements for XML: DB API Core Level 1 compliance.

8. References

[1] World Wide Web Consortium. Extensible Markup Language (XML) 1.0, 1998.

[2] http://www.w3C.org/TR/REC-xml/.
[3]World Wide Web Consortium. XML Schema, 2001.

[4]http://www.w3C.org/TR/xmlschema-1/
[5]R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases, pages 487{499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[6]S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava. Minimization of tree pattern queries. In SIGMOD Conference, pages 497{508, 2001.

[7]T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering frequent substructures in large unordered trees, 2003.

[8]E. Baralis, P. Garza, E. Quintarelli, and L. Tanca. Answering xml queries by means of data summaries. ACM Transactions of Information Systems, 25(3):10, 2007.

[9]E. Bertino, B. Catania, and W. Q. Wang. Xjoin index: Indexing xml data for ecient handling of branching path expressions. In International Conference on Data Engineering, page 828, 2004.

[10]D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. Lanzi. Discovering interesting information in xml data with association rules. In SAC '03: Proceedings of the 2003 ACM symposium on Applied computing, pages 450{454, New York, NY, USA, 2003. ACM Press.

[11]Sedna Native XML Database System. www.sedna.org.

[12] 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.

[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] 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.

[15] E. Baralis, P. Garza, E. Quintarelli, and L. Tanca. Answering xml queries by means of data summaries. ACM Transactions on Information Systems, 25(3):10, 2007.

[16] 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.

[17] D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. Lanzi. Discovering interesting information in xml data with association rules. In Proc. Of the ACM Symposium on Applied Computing, pages 450–454, 2003.

[18] Y. Chi, Y. Yang, Y. Xia, and R. R. Muntz. Cmtreeminer: Mining both closed and maximal frequent subtrees. In Proc. of the 8th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pages 63–73, 2004.

[19] C. Combi, B. Oliboni, and R. Rossato. Querying xml documents by using association rules. In Proc. of the 16th Int. Conf. on Database and Expert Systems Applications, pages 1020–1024, 2005.

[20] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In Proc. of the 8th ACM Int.Conf. on Knowledge Discovery and Data Mining , pages 217–228, 2002.

[21] L. Feng, T. S. Dillon, H. Weigand, and E. Chang. An xml-enabled association rule framework. In Proc. of the 14th Int. Conf. on Database and Expert Systems Applications, pages 88–97, 2003.

[22] 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.

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

[24] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semi structured databases. InProc. of the 23rd Int. Conf. on Very Large Data Bases, pages 436–445, 1997.

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

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

[27] A. Jim ´enez, F. Berzal, and J. C. Cubero. Mining induced and embededsubtrees in ordered, unordered, and partially-ordered trees. In Proc. Of the 17th Int. Symposium on Methodologies for Intelligent Systems , pages111–120, 2008.

[28] D. Katsaros, A. Nanopoulos, and Y. Manolopoulos. Fast mining of frequent tree structures by hashing and indexing. Information & Software Technology , 47(2):129–140, 2005.

[18] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. IEEE Transactions on Knowledge and Data Engineering , 16(9):1038–1051, 2004.

[19] H. C. Liu and J. Zeleznikow. Relational computation for mining association rules from xml data. In Proc. of the 14th ACM Conf. on Information and Knowledge Management, pages 253–254, 2005.

[20] Gary Marchionini. Exploratory search: from finding to understanding.Communications of the ACM , 49(4):41–46, 2006.

[21] M. Mazuran, E. Quintarelli, and L. Tanca. Mining tree-based association rules from xml documents. In Technical Report, Politecnico di Milano. http://home.dei.polimi.it/quintare/Papers/MQT09-RR.pdf , 2009.

[22] M. Mazuran, E. Quintarelli, and L. Tanca. Mining tree-based frequent patterns from xml. In Proc. of the 8th Int. Conf. on Flexible Query Answering Systems , pages 287–299, 2009.

[23] S. Nijssen and J.N. Kok. Efficient discovery of frequent unordered trees.In Proc. of the 1st Int. Workshop on Mining Graphs, Trees and Sequences, 2003.[24] J. Paik, H. Y. Youn, and U. M. Kim. A new method for miningassociation rules from a collection of xml documents. In Proc. of Int. Conf. on Computational Science and Its Applications, pages 936–945, 200