Data Mining for Xml Query Answering Support

Srinath T K*, Mr. Joby George**

*Department of Computer Science and Engineering, Mar Athanasius College of Engineering, India
**Department of Computer Science and Engineering,Mar Athanasius College of Engineering, India

Abstract- Extracting information from semi structured documents is a very hard task, and is going to become more and more critical as the amount of digital information available on the internet grows. Indeed, documents are often so large that the dataset returned as answer to a query may be too big to convey interpretable knowledge. In this work we describe an approach based on Tree-based Association Rules (TARs) mined rules, which provide approximate, intentional information on both the structure and the contents of XML documents, and can be stored in XML format as well. This mined knowledge is later used to provide: (i) a concise idea – the gist – of both the structure and the content of the XML document, (ii) quick, approximate answers to queries and (iii) output without redundancy, null tags and empty tag. In this work we focus on the second and third feature. A prototype system and experimental results demonstrate the effectiveness of the approach.

Index Terms- XML, approximate query-answering, data mining, intentional information, succinct answers Index Terms- DNA, HCRs, MSA, T-Coffee, Muscle, and Crustal-W

I. INTRODUCTION

In recent years the database research field has concentrated on XML (extensible Markup Language [30]) as a flexible hierarchical model suitable to represent huge amounts of data with no absolute and fixed schema, and a possibly irregular and incomplete structure. There are two main approaches to XML document access: *keyword-based search* and *query-answering*. The first one comes from the tradition of information retrieval [20], where most searches are performed on the textual content of the document; this means that no advantage is derived from the semantics conveyed by the document structure. As for *query-answering*, since query languages for semi structured data rely the on document structure to convey its semantics, in order for query formulation to be effective users need to know this structure in advance, which is often not the case. In fact, it is not mandatory for an XML document to have a defined schema: 50% of the documents on the web do not possess one [5]. When users specify queries without knowing the document structure, they may fail to retrieve information which was there, but under a different structure. This limitation is a crucial problem which did not emerge in the context of relational database management systems. Frequent, dramatic outcomes of this situation are either the *information overload* problem, where too much data are included in the answer because the set of keywords specified for the search captures too many meanings, or the *information deprivation* problem, where either the use of inappropriate keywords, or the wrong formulation of the query, prevent the user from receiving the correct answer. As a consequence, when accessing for the first time a large dataset, gaining some general information about its main structural and semantic characteristics helps investigation on more specific details. This paper addresses the need of getting the gist of the document before querying it, both in terms of content and structure. Discovering recurrent patterns inside XML documents provides high-quality knowledge about the document content: frequent patterns are in fact intentional information about the data contained in the document itself, that is, they specify the document in terms of a set of properties rather than by means of data. As opposed to the detailed and precise information conveyed by the data, this information is partial and often approximate, but synthetic, and concerns both the document structure and its content. In particular, the idea of mining association rules [1] to provide summarized representations of XML documents has been investigated in many proposals either by using languages (e.g. XQuery [29]) 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 intentional knowledge in native XML. Intuitively, a TAR represents intentional knowledge in the form SB ⇒ SH, where SB is the body tree and SH the head tree of the rule and SB is a sub tree of SH. The rule SB ⇒ SH states that, if the tree SB appears in an XML document D, it is likely that the "wider" (or "more detailed"), tree SH also appears in D. of indexes and the design of efficient access methods for frequent queries, and also because frequent patterns allow to discover 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 [9]. 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. Summarizing, TARs are extracted for two main purposes: *1)* to get a concise idea – the *gist* – of both the structure and the content of an XML document, and *2)* to use them for *intentional query answering*, that is, allowing the user to query the extracted TARs rather than the original document. In this paper we concentrate mainly on the second task. We have applied our techniques in the DATABASE OF BT (British Telecom), whose objective is to develop a platform for automated sharing, management, processing, analysis and use of telecom information across UK and Ireland. Frequent patterns, in the form of TARs, provide summaries of these integrated datasets shared by different telecom connections. By querying such summaries, engineers obtain initial knowledge about specific

entities in the vast dataset(s), and are able to devise more specific queries for deeper business intelligence. An important side-effect of using such a technique is that only the most promising specific queries are issued towards the integrated data, dramatically reducing time and cost.

II. GOALS AND CONTRIBUTIONS

This paper provides a 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: a) it works directly on the XML documents, without transforming the data into any intermediate format, b) it looks for general association rules, without the need to impose what should be contained in the antecedent and consequent of the rule, c) it stores association rules in XML format, and d) 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, like in [34]. Accordingly, the paper's contributions are:

• an improved version of the TARs extraction algorithm introduced in [22]. The new version uses the better-performing CMTreeMiner [7] to mine frequent sub trees from XML documents; automatic user-query transformation into "equivalent" queries over the mined intensional knowledge.

• as a formal corroboration of the accuracy of the process, the proof that our intensional-answering process is sound and complete up to a frequency threshold.

III. PROPOSED WORK

Association rules [1] describe the co-occurrence of data items in a large amount of collected data and are represented as implications of the form $X \Rightarrow Y$, where X and Y are two arbitrary sets of data items, such that $X \cap Y = \emptyset$. The quality of an association rule is 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 $supp(X \cup Y)/supp(X)$. 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 tree2 N,E, r, , c where N is the set of nodes, $r \in N$ is the root of the tree, E is the set of edges, : $N \rightarrow L$ is the label function which returns the tag of nodes (with L the domain of all tags) and $c : N \rightarrow C \cup \{\perp\}$ 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 [28], where XML nonterminal tags include only other elements and/or attributes, while the text is confined to terminal elements. We are interested in finding relationships among subtrees 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 figures. Attributes and elements are characterized by empty circles, whereas the textual content of elements, or the value of attributes, is reported under the outgoing edge of the element or attribute. Given two trees T = _NT,ET , rT , _T , cT _ and S = _NS,ES, rS, _S, cS_, S is an induced subtree of T if and only if there exists a mapping $\theta : NS \rightarrow NT$ such that for each node $ni \in NS$, T (ni) = S(nj) and cT (ni) = cS(nj), where $\theta(ni) = nj$, and for each edge $e = (n1, n2) \in ES$, $(\theta(n1), \theta(n2)) \in ET$. Moreover, S is a rooted subtree of T if and only if S is an induced subtree of T and rS = rT . Given a tree T = NT,ET , rT , T , cT , a subtree of T, t = Nt,Et, rt, t, ctand a user-fixed support threshold smin: (i) t is frequent if its support is greater or at least equal to smin; (ii) t is maximal if it is frequent and none of its proper supertrees is frequent; (iii) t is closed if none of its proper supertrees has support greater than that of t. A Tree-based Association Rule (TAR) is a tuple of the form Tr = SB, SH, sTr, cTr , where SB = NB,EB, rB, B, cB and SH = NH,EH, rH, H, cHare trees and sTr and cTr are real numbers in the interval [0,1] representing the support and confidence of the rule respectively (defined below). A TAR describes the co-occurrence of the two trees SB and SH in an XML document. For the sake of readability we shall often use the short notation $SB \Rightarrow SH$; SB is called the body or antecedent of Tr while SH is the head or consequent of the rule. Furthermore, SB is a subtree.

A Tree-based Association Rule (TAR) is a tuple of the form Tr = SB, SH, sTr, cTr , where SB = NB,EB, rB, B, cB and SH = NH,EH, rH, H, cHare trees and sTr and cTr are real numbers in the interval [0,1] representing the support and confidence of the rule respectively (defined below). A TAR describes the co-ccurrence of the two trees SB and SH in an XML document. For the sake of readability we shall often use the short notation $SB \Rightarrow SH$; SB is called the body or antecedent of Tr while SH is the head or consequent of the rule. Furthermore, SB is a subtree of SH with an additional property on the node labels: the set of tags of SB is contained in the set of tags of SH with the addition of the empty label

"

": SB(NSB) ⊆ SB(NSB)∪{

}.

A rooted TAR (RTAR) is a TAR such that SB is a rooted subtree of SH

• an extended TAR (ETAR) is a TAR such that SB is an induced subtree of SH Let count(S,D) denote the number of occurrences of a subtree S in the tree D and cardinality(D) denote the number of nodes of D. We formally define the support of a TAR $SB \Rightarrow SH$ as count(SH,D)/cardinality(D) and its confidence as count(SH,D)/count(SB,D). Notice that TARs, in addition to associations between data values, also provide information about the structure of frequent portions of XML documents; thus they are more expressive than classical association rules which only provide frequent correlations of flat values.

It is worth pointing out that TARs are different from XML association rules as defined in [24], because, given a rule X ⇒ Y where both X and Y are subtrees of an XML document, that paper require that (X⊆ Y)∧(Y⊆ X), i.e. the two trees X and Y have to be disjoint; on the contrary TARs require X to be an induced subtree of Y .

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 and on the data values contained in a document.

According to the definitions above we have: structure-Rooted-TARs (sRTARs), structure-Extended-TARs (sETARs), instance-Rooted-TARs (iRTARs) and instance-Extended-TARs (iETARs).

TAR mining is a process composed of two steps: 1) mining *frequent subtrees*, that is, subtrees *with a support above a userdefined threshold*, from the XML document; 2) computing interesting rules, that is, rules *with a confidence above a userdefined threshold*, from the frequent subtrees.

Algorithm 1 INTERESTING RULES ((D,minisupp miniconf)

- To search frequent subtrees
- 1: // To Search For frequent subtrees
- 2: FS=Find Frequent Subtrees (D, minisupp)
- 3: ruleSet = Ø // rule set is initialised to null
- 4: for all s in Frequent Subtree DO
- 5: // rules computed from s
- 6: tempSet = Algorithm2 (**Compute-Rules** (s, miniconf))
- 7: //For all rules
- 8: ruleset = ruleSet U tempSet
- 9: end for
- Return ruleset.

Algorithm 1 presents our extension to a generic frequentsubtree mining algorithm in order to compute interesting TARs. The inputs of Algorithm 1 are the XML document D, the threshold for the support of the requent subtrees *minsupp*, and the threshold for the confidence of the rules, *minconf*. Algorithm 1 finds frequent subtrees and then hands each of them over to a function that computes all the possible rules. Depending on the number of frequent subtrees and their cardinality, the amount of rules generated by a naive Compute- Rules function may be very high. Given a subtree with *n* nodes, we could generate $2n - 2$ rules, making the algorithm exponential. This explosion occurs in the relational context too, thus, based on similar considerations [1], it is possible to state the following property, that allows us to propose the optimized vesion of Compute-Rules shown in Function 2.

Remark 1. If the confidence of a rule *SB ⇒ SH* is below the established threshold *minconf* then the confidence of every other rule *SBi ⇒ SHi* , such that its body *SBi* is an induced subtree of the body *SB*, is no greater than minconf.

Algorithm 2: Compute-Rules(s, miniconf)

1: ruleSet = Ø; blacklist = Ø // initialise ruleset and blacklist – null
2: for all CS, subtrees of s do
3: if CS is not a subtree of any element in blacklist then
4: conf = sup(s) / sup(CS)
5: if conf >= minconf then
6: newRule = (CS,s,conf,supp(s))
7: if the rule set is Unique AND if the rule set does not carry NULL AND if the rule set does not carry empty tags then step 8
8: ruleSet = ruleSet U {newRule}
9: else
10: blackList = blackListU CS
11: end if
12: end if
13: end for
14: return ruleSet

Here null, empty and redundant tags get blacklisted providing a faster and precise solution space.

In Algorithm 2 TARs are mined exploiting Remark 1 by generating first the rules with the highest number of nodes in the body tree. Consider two rules Tr1 and Tr2 whose body trees contain one and three nodes espectively; suppose both rules have confidence below the fixed threshold. If the algorithm considers rule Tr2 first, all rules whose bodies are induced subtrees of Tr2 will be discarded when Tr2 is eliminated. Therefore, it is more convenient to first generate rule Tr2 and in general, to start the mining process from the rules with a larger body. Using this solution we can lower the complexity of the algorithm, though not enough to make it perform better than exponentially. However, notice that the process of deriving TARs from XML documents is only done periodically. Since intensional knowledge represents frequent information, to update it, it is desirable to perform such process after a big amount of updates have been made on the original document. Therefore, in the case of stable documents (that is, those that are rarely updated) the algorithm has to be applied few times or only once (for documents that do not change).

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. Each rule is saved inside a <rule> element which contains three attributes for the ID, support and confidence of the rule. Follows the list of elements, one for each node in the rule head. We exploit the fact that the body of the rule is a subtree of the head, and use a boolean attribute in each node to indicate if it also belongs to the body. Each blank node is described by an element <blank>. Finally, the rules in the XML file are sorted on the number of nodes of their antecedent.

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 intensional query

answering. In the literature the problem of making XML query-answering faster by means of pathbased indexes has been investigated (see [34] for a survey). 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.

Figure1 : Redundant and null values returned (run on just a single sample XML)

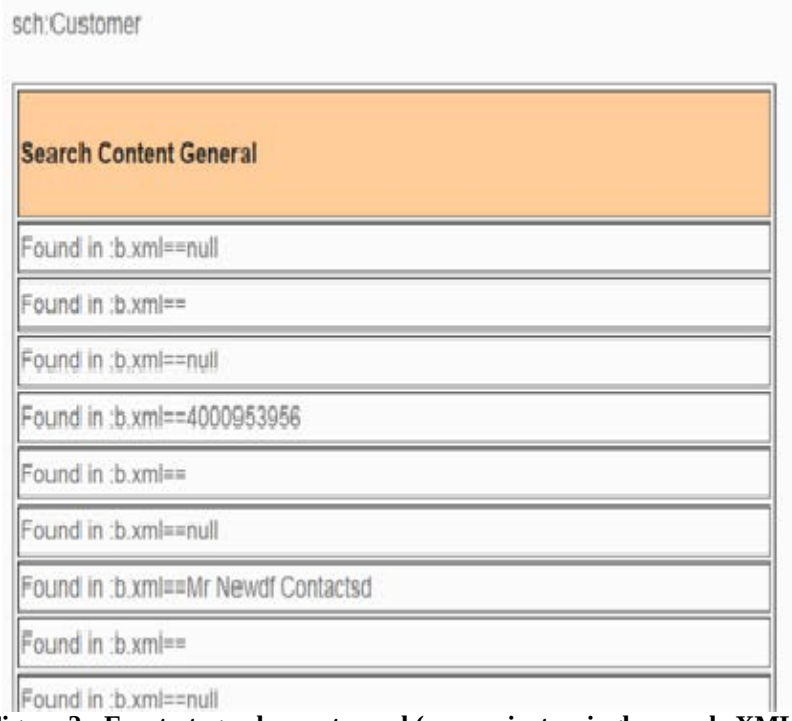

Figure 2 : Empty tag values returned (run on just a single sample XML)

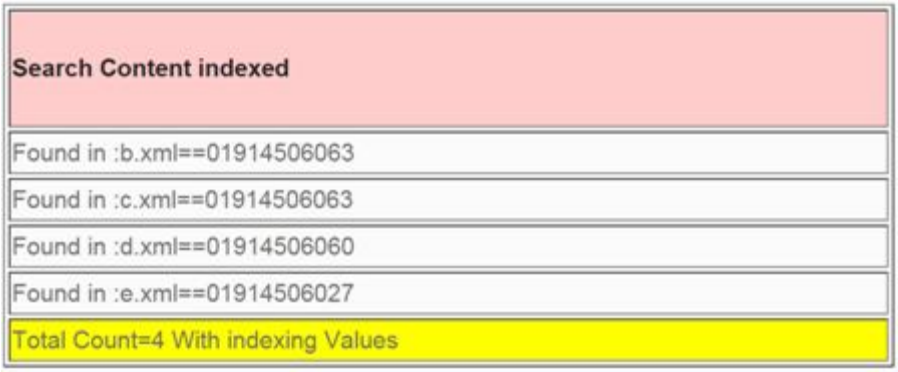

Figure 3 : Empty tag, null values, redundancy removed

IV. Conclusion And Future Work

The main goals we have achieved in this work are: 1) mine all frequent association rules without imposing any a-priori 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. 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 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.

Acknowledgments

Firstly, I thank God Almighty for his grace upon me. I would like to take this opportunity to express my profound gratitude and deep regards to Mr. Joby George, for his exemplary guidance, valuable feedback and constant encouragement throughout the duration of the project. His valuable suggestions were of immense help throughout my project work. I would also thank all my friends, who had motivated me with their knowledge. Finally, I would thank my family, who had supported me spiritually throughout this thesis work.

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 Data Bases, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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 mining association rules from a collection of xml documents. In Proc. of Int. Conf. on Computational Science and Its Applications, pages 936–945, 2005.

[25] A. Termier, M. Rousset, and M. Sebag. Dryade: A new approach for discovering closed frequent trees in heterogeneous tree databases. In Proc. of the 4th IEEE Int. Conf. on Data Mining, pages 543–546, 2004.

[26] A. Termier, M. Rousset, M. Sebag, K. Ohara, T. Washio, and H. Motoda. Dryadeparent, an efficient and robust closed attribute tree mining algorithm. IEEE Transactions on Knowledge and Data Engineering, 20(3): 300–320, 2008.

[27] World Wide Web Consortium. XML Schema, 2001. http://www.w3C.org/TR/xmlschema-1/.

[28] World Wide Web Consortium. XML Information Set, 2001. http://www.w3C.org/xml-infoset/.

[29] World Wide Web Consortium. XQuery 1.0: An XML query language, 2007. http://www.w3C.org/TR/xquery.

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

[31] J. W. W. Wan and G. Dobbie. Extracting association rules from xml documents using xquery. In Proc. of the 5th ACM Int. Workshop on Web Information and Data Management, pages 94–97. ACM Press, 2003.

[32] 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.

[33] K. Wang and H. Liu. Discovering structural association of semistructured data. IEEE Transactions on Knowledge and Data Engineering, 12(3):353–371, 2000.

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

[35] 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.

[36] 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.

[37] 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.

AUTHORS

First Author- Srinath T K- M.Tech Final Year Student, Mar Athanasius College of Engineering, Kerala, India
Srinathtk86@gmail.com@gmail.com
Second Author- Mr. Joby George, Professor, Mar Athanasius College of Engineering, Kerala, India
jobygeo@hotmail.com