ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Mining frequent XML documents using FFP

Amar Nayak
Technocrats Institute of Technology
Bhopal M.P. India
agn_1975@yahoo.com

Abstract: XML has become very popular for representing semi structured data and a standard for data exchange over the web. Mining XML data from the web is becoming increasingly important. To date, the famous Apriori algorithm to mine any XML document for association rules without any pre-processing or post-processing has been implemented using only the XQuery language which is costly. But the algorithm only can mine the set of items that can be written a path expression for. However, the structure of the XML data can be more complex and irregular than that. Consequently, it is difficult to identify the mining context. This paper propose that extracting association rules from XML documents without any preprocessing or postprocessing using XML query language XQuery is possible and analyze the XQuery implementation of the efficient First Frequent method-tree based mining method, First Frequent Pattern-growth, for mining the complete set of frequent patterns by pattern fragment growth. First Frequent Pattern-tree based mining adopts a pattern fragment growth method to avoid the costly generation of a large number of candidate sets and a partition-based, divide-and-conquer method is used. In addition, we suggest features that need to be added into XQuery in order to make the implementation of the First Frequent Pattern growth more efficient.

Keywords: *Data Mining, XML, XPATH, XQuery, SDST*

I. INTRODUCTION

The goal of data mining [4] is to extract or mine" knowledge from large amounts of data. Consequently, data mining consists of more than collecting and managing data, it also includes analysis and prediction. Data mining can be performed on data represented in quantitative, textual, or multimedia forms. Web mining is the use of data mining technologies to automatically interact and discover information from web documents, which can be in structured, unstructured or semi- structured form. The eXtensible Markup Language (XML)[18] has become a standard language for data representation and exchange. XML is a Standard, flexible *syntax* for data exchanging Regular, structured data. Database content of all kinds: Inventory, billing, orders etc. It has small typed values and irregular, unstructured text.

With the continuous growth in XML data sources, the ability to manage collections of XML documents and discover knowledge from them for decision support becomes increasingly important. Mining of XML documents significantly differs from structured data mining and text mining. XML allows the representation of semi-structured and hierarchal data containing not only the values of individual items but also the relationships between data items. Element tags and their nesting therein dictate the structure of an XML document. Due to the inherent flexibility of XML, in both structure and semantics, discovering knowledge from XML data is faced with new challenges as well as benefits. Mining of structure along with content provides new insights and means into the process of knowledge discovery.

XML documents are becoming ubiquitous because of their rich and flexible format that can be used for a variety of applications. Standard methods have been used to classify XML documents, reducing them to their textual parts. XML was designed to transport and store data. XML tags are not predefined. You must define your own tags XML documents form a tree structure that starts at "the root" and branches to "the leaves".

II. DOCUMENT MINING, ASSOCIATION RULE, XML, XPATH, XQUERY

A. Document Mining (DM) or Text Mining:

The process of finding interesting or useful patterns in a corpus of unstructured textual information. Document Mining combines many of the techniques of information extraction, information retrieval, and natural language processing and document summarization with the methods of data mining. The key use for document mining is to elicit previously unknown knowledge locked away in a volume of text. The terms document mining (DM), text mining (TM) and knowledge acquisition from textual databases (KDT) are used interchangeably to represent the same area of document processing. XML data mining: queries. As a matter of consequence, the retrieval precision gets better.

This latter is a very significant mechanism in the context of XML retrieval due to two reasons: (1) a user query can be satisfied by means of different possible answers; closely associated documents tend to be relevant to the same requests, and (2) retrieval systems and web mining tools are generally operational in the same environment. Classifying (and clustering) XML documents can be basically done in three ways: (1) using exclusively the textual contents of documents as usually done in traditional text categorization (and clustering) systems, (2) using exclusively the structure of the XML documents, and (3) using both the contents and the structure in a hybrid manner.

B. Association Rules:

Association rule mining, one of the most important and well researched techniques of data mining. It aims to extract interesting correlations, frequent patterns, associations or

casual structures among sets of items in the transaction databases or other data repositories. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control etc. Association algorithms find correlations between different attributes in a dataset. The most common application of this kind of algorithm is for creating association rules, which can be used in a market basket analysis. An example of an association algorithm is the Association Algorithm. Association rules were first introduced by Agrawal et al to analyze customer buying habits in retail databases. Consider a retail store which sells a great many items. In order to increase the sales and profits, it will make some business decisions based on the knowledge about historical transactions data that gives the buying habits of the customer. The association rule is in the form of customer who bought item M may also bought item N. So the goal of association rules is to extract such rules from the given transaction data history stored in database.

C. XQuery:

XQuery 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 XPath expressions.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.

XPath Standard Functions: XPath includes over 100 built-in functions. There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more. XPath is Used in XSLT. XPath is a major element in the XSLT standard. Without XPath knowledge you will not be able to create XSLT documents.

XQuery and XPointer are both built on XPath expressions. XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<students>
<student>
 <rollno>1001</rollno>
 <name>Rakesh Tiwari</name>
 <branch>IT</branch>
  <sem>V</sem>
</student>
</students>
```

Example of nodes in the XML document

<students> (document node)

<name>Rakesh Tiwari</name> (element node)

Introduction to XSLT: XSLT is the style sheet language for XML files. With XSLT you can transform XML documents into other formats, like XHTML.

D. XML:

The eXtensible Markup Language (XML) has become a standard language for data representation and exchange. XML[9] is a Standard, flexible *syntax* for data exchanging Regular, structured data. Database content of all kinds: Inventory, billing, orders etc. It has small typed values and irregular, unstructured text. It can consists of documents of all kinds: Transcripts, books, legal briefs etc. With the continuous growth in XML data sources, the ability to manage collections of XML documents and discover knowledge from them for decision support becomes increasingly important. Mining of XML documents significantly differs from structured data mining and text mining. XML allows the representation of semi-structured and hierarchal data containing not only the values of individual items but also the relationships between data items. Element tags and their nesting therein dictate the structure of an XML document. XML was designed to transport and store data. XML tags are not predefined. You must define your own tags XML documents[10] form a tree structure that starts at "the root" and branches to "the leaves".

III. RELATED WORK

Algorithms for mining association rules from relational data have been well developed. Several query languages have been proposed, to assist association rule mining. For instance, [12] has proposed an algorithm to construct a frequent tree by finding common subtrees embedded in the heterogeneous XML data. On the other hand, some researchers focus on developing a standard model to represent the knowledge extracted from the data using XML. For example, the Predictive Model Markup Language (PMML) [1] is an XML-based language, which provides a way for applications to define statistical and data mining models and to share models between PMML compliant applications. To date, mining XML documents requires mapping the data to the relational data model and using techniques designed for relational databases to do the mining. For instance, the XMINE operator has been introduced by Braga et al. [9] for extracting association rules from XML documents, where mapping the XML data to a relational structure is required before mining is performed.

A. XML Algebras for Data Mining:

The prerequisite of designing XML mining algebras is that all generalized knowledge must be expressed in XML, i.e. input and output of operators must have the same form. Extension of existing XML algebras by adding some mining operators might work well as a mining algebra. Each

operator is associated with an algorithm for the mining task. However, this method leaves little room for optimization. A further development is to decompose mining tasks into several sub-tasks. These sub-tasks are shared among different mining tasks. All operators should have the following properties:

a. The operation of each operator should be clearly defined and easily implemented. 2. The input and output of each operator should be in the same form.

The ultimate goal of XML algebras is to provide operational procedure to obtain answers of user's queries. XAL is simple and could be implemented easily. Niagara extends the definition of basic unit from one vertex to bag of vertices. TAX is an algebra managing collections of tree directly. These XML algebras are good starting point to design algebras for XML mining. Extension of the existing algebras by adding some operators for mining tasks might work well. The intuitive way to design one operator for each type of mining task is inefficient in execution and leaves little room for optimization. A better solution is to design one operator for each sub-task whose operation is much simpler. How to decompose the mining tasks into sub-tasks is still an open problem, and it is largely dependent on the nature of the mining tasks.

B. Vector Space Model (VSM):

The VSM is a special type of kernel methods for textual data. It transforms a document into a numerical vector, each element of which is an indicator whether or not the document contains a certain word (or phrase). A VSM can be represented either in a boolean vector model, which has only zeros or ones indicating respectively words' absence or presence, or in a term weighting model, which has scalar values that take into account the appearance frequency of a term in a set of documents. A widely used weighting method is 'Term Frequency-Inverse Document Frequency (TF-IDF)' [5], in which the weight of the ith term in the jth document, denoted as wij , is defined by wij = TFij × IDFi = TFij × log(n/DFi), where TFij is the number of occurrences of the ith term within the jth document, andDFi is the number of documents (out of n) in which the term appears.

For two documents v1 and v2, their semantic distance is computed as the cosine of their angle _, i.e., d(v1, v2) = cos _ = VT v1Vv2/kVv1k2kVv2k2. Since this interpretation requires too much computation for massive real-world documents, alternatively it is often to use LSA (or LSI, Latent Semantic Analysis/Indexing)10 [23] and PCA (Principal Components Analysis) [5] of the original VSM for computation efficiency.

C. Kernel-based XML Mining:

The kernel methods for structured data are applicable to XML mining in various ways. The string kernels with minor modifications are used to not only identify and extract useful information from textual contents as used in [6], but also to compare two pieces of context information (i.e., successive element labels in a path). Undoubtedly, the VSM also provides the same stateof-the-art performance in XML content mining as in traditional text mining. The tree kernels are used to measure the structural similarity between XML schema documents and between XML instance documents. By transforming the tree structure into a string (e.g., by a depth-first traversal), the string kernels and the VSM are also used to compute the structural similarity.

IV. FP-GROWTH ALGORITHM

In 2000, Han *et al.* proposed the FP-growth algorithm—the first pattern-growth concept algorithm. FP-growth constructs an FP-tree structure and mines frequent patterns by traversing the constructed FPtree. The FP-tree structure is an extended prefix-tree structure involving crucial condensed information of frequent patterns.

A. FP-tree Structure:

The FP-tree structure has sufficient information to mine complete frequent patterns. It consists of a prefixtree of frequent 1-itemset and a frequent-item header table. Each node in the prefix-tree has three fields: *item-name*, *count*, and *node-link*. *item-name* is the name of the item. *count* is the number of transactions that consist of the frequent 1-items on the path from root to this node. *node-link* is the link to the next same itemname node in the FP-tree. Each entry in the frequent-item header table has two fields: *item-name* and *head of node-link*. *item-name* is the name of the item. *head of node-link* is the link to the first same item-name node in the prefix-tree.

B. Construction of FP-tree:

FP-growth has to scan the *TDB* twice to construct an FP-tree. The first scan of *TDB* retrieves a set of frequent items from the *TDB* . Then, the retrieved frequent items are ordered by descending order of their supports. The ordered list is called an F-list. In the second scan, a tree *T* whose root node *R* labeled with "null" is created. Then, the following steps are applied to every transaction in the *TDB* . Here, let a transaction represent [p\P] where *p* is the first item of the transaction and *P* is the remaining items. In each transaction, infrequent items are discarded. Then, only the frequent items are sorted by the same order of F-list.

Call insert_tree (p\P,R) to construct an FP-tree.

The function insert_tree (p\P,R) appends a transaction [p\P] to the root node *R* of the tree *T* . Pseudo code of the function insert_tree (p\P,R) An example of an FP-tree is shown. This FP-tree is constructed from Table 2 with min-sup=2. In Figure 1, every node is represented by (item-name: count). Links to next same item-name node are represented by dotted arrows.

function insert_tree (*p*|*P*,*R)* {
let *N* be a direct child node of R, such that *N* 's
item-name = *p* 's *item-name*.
if (*R* has a direct child node *N*) {
increment *N* 's *count* by 1. }
else{ create a new node *M* linked under the *R* .
set *M* 's *item-name* equal to *p* . set *M* 's *count* equal to 1. }
call insert_tree (*P* ,N). }

Table 1 Sample data

Transaction ID	Items	Frequent Items
100	A, B, C	A, B, E
200	B, D	B, D
300	B, C	B, C
400	A, B, D	A, B, D
500	B, C	B, C
600	A, B, C, E	A, B, C, E
700	A, B, C	A, B, C

B	7
A	4
C	4
D	2
E	2

C. FP-Growth:

FP-growth mines frequent patterns from an FP-tree. To generate complete frequent patterns, FP-growth traverses all the node-links from "head of node-links" in the FPtree's header table. For example, we describe how to mine all the frequent patterns including item C from the FP-tree shown in Figure 1. For node C , FP-growth mines a frequent pattern (C:4) by traversing C 's node-links through node (C:2) to node (C:2). Then, it extracts C 's prefix paths; <B:7,A:4> and <B:7>. To study which items appear together with C , the transformed path <B:2,A:2> is extracted from <B:7,A:4> because the support value of C is 2. Similarly, we have <B:2>. The set of these paths {(B:2,A:2),(B:2)} is called C 's conditional pattern base. FP-growth then constructs C's conditional FP-tree containing only the paths in C's conditional pattern base as shown in Table 3 . As only B is an item occurring more than *min_sup* appearing in C 's conditional pattern base, C 's conditional FP-tree leads to only one branch (B:7). Hence, only one frequent pattern (BC:4) is mined. The final frequent patterns including item C are (C:4) and (BC:4).

To date, the famous Apriori algorithm to mine any XML document for association rules without any pre-processing or post-processing has been implemented using only the XQuery language which is costly. But the algorithm only can mine the set of items that can be written a path expression for. However, the structure of the XML data can be more complex and irregular than that. Consequently, it is difficult to identify the mining context. This thesis propose that extracting association rules from XML documents without any preprocessing or postprocessing using XML query language XQuery is possible and analyze the XQuery implementation of the efficient First Frequent method-tree based mining method, First Frequent Pattern-growth, for mining the complete set of frequent patterns by pattern fragment growth. First Frequent Pattern-tree based mining adopts a pattern fragment growth method to avoid the costly generation of a large number of candidate sets and a partition-based, divide-and-conquer method is used. Divide-and-conquer method divides the problem into a number of subproblems and the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner and then combine the solutions to the subproblems into the solution for the original problem. In addition, we suggest features that need to be added into XQuery in order to make the implementation of the First Frequent Pattern growth more efficient.

V. CONCLUSION AND FUTURE WORK

This paper propose that extracting association rules from XML documents without any preprocessing or postprocessing using XML query language XQuery is possible and analyze the XQuery implementation of the efficient First Frequent method-tree based mining method, First Frequent Pattern-growth, for mining the complete set of frequent patterns by pattern fragment growth. First Frequent Pattern-tree based mining adopts a pattern fragment growth method to avoid the costly generation of a large number of candidate sets and a partition-based, divide-and-conquer method is used. Divide-and-conquer method divides the problem into a number of subproblems and the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner and then combine the solutions to the subproblems into the solution for the original problem. In addition, we suggest features that need to be added into XQuery in order to make the implementation of the First Frequent Pattern growth more efficient.

In our future work we are planning to use semantic web mining techniques such as RDF (Resource Description Framework) and Ontology for better search and decision. We are also planning to implement other standard data mining algorithms which can be expressed in XQuery to improve the performance of the results.

VI. REFERENCES

[i]. Chit nilar and Khin haymar saw hla, Mining frequent patterns from XML data, IEEE, 2007

[ii]. Jiang-Chun Song, Jun-Yi-Shen, Qing-Bao Song, " A New Document Clustering Algorithm Base On Association Rule ", IEEE, Third International Conference on Machine Learning and Cybernetics, 2004

[iii]. Andre Vizine, Landro N. De Castro, Richord Gudwin, An Evolutionary Algorithm to Optimize Web Document Retrieval", IEEE, KISMAS, 2005

[iv]. Ludovic Denoyer and Patrick Gallinari, Categorization and Clustering of XML Documents, Report on the XML Mining Track at INEX 2007

[v]. Wang Feng Xuwei Li Zhu Hong , An Immune-based Model for Web Data Mining (© 2005 IEEE Xplore

[vi]. André L. Vizine1, Leandro N. de Castro1 & Ricardo R. Gudwin2, An Evolutionary Algorithm to Optimize Web Document Retrieval @ 2005 IEEE.

[vii]. Converting Web Pages into Well-formed XML Documents H. Ouahid, A. Karmouch Q00 1 999 IEEE.

[viii]. Osmar R. Zaïane Principles of Knowledge Discovery in Databases, , 1999

[ix]. Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, and Andrea Pugliese , Fast Detection of XML Structural Similarity;; IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 17, NO. 2, FEBRUARY 2005

[x]. Calin Garboni, Florent Masseglia, and Brigitte Trousse, Sequential Pattern Mining for Structure-Based XML Document Classification, 2006

[xi]. Buhwan Jeong, Daewon Lee, Jaewook Lee, and Hyunbo Cho Towards XML Mining: The Role of Kernel Methods, 2004

[xii]. Jung-Won Lee, Kiho Lee, Won Kim; Preparations for Semantics- Based XML Mining in @2001 IEEE Xplore

[xiii]. Anne-Marie Vercoustre, Mounir Fegas, Saba Gul, and Yves Lechevallier; rXiv:cs/0607012v1 [cs.IR] A Flexible

Structured-based Representation for XML Document Mining 5 Jul 2006

[xiv]. M. Zhang J.T. Yao; ,XML Algebras for Data Mining,2003

[xv]. Qin Bao Song, Nai Qina Li, Jun Yi Shen Li Ming Chen, Web Documents Mining; , Proceeding of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 Novemeber 2002.