International Journal of Engineering Research (ISSN : 2319-6890)
Volume No.2, Issue No.1, pp : 07-12 01 Jan. 2013

A Recent Review on XML data mining and FFP

Amit Kumar Mishra, Hitesh Gupta
Patel College of Science and Technology, Bhopal, M.P. India
amit.sist1@gmail.com, gupta_hitesh@sify.com

Abstract
The goal of data mining is to extract or mine" knowledge from large amounts of data. Emerging technologies of semi-structured data have attracted wide attention of networks, e-commerce, information retrieval and databases.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. However, the structure of the XML data can be more complex and irregular than that. Association Rule Mining plays a key role in the process of mining data for frequent pattern matching. 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. This paper shows a complete review of XML data mining using Fast Frequent Pattern mining in various domains.

Keywords: Data mining,* semi-structured data mining, *Association mining, XML

I. INTRODUCTION

The World Wide Web is one of the fastest growing areas of intelligence gathering. Data mining[1], which is also referred to as knowledge discovery in databases, means a process of nontrivial extraction of implicit, previously unknown and potentially useful information (such as knowledge rules, constraints, regularities) from data in databases. Other terms for data mining are knowledge mining from databases, knowledge extraction, data archaeology, data dredging, data analysis, etc. Mining information and knowledge from large databases has been recognized by many researchers as a key research topic in database systems and machine learning and by many industrial companies as an important area with an opportunity of major revenues. The discovered knowledge can be applied to information management, query processing, decision making [1],

process control, and many other applications. Researchers in many different fields, including database systems, knowledge-base systems, artificial intelligence, machine learning, knowledge acquisition, statistics, and spatial databases have shown great interest in data mining. Nowadays, we have huge amount of data stored in each institute, university, and company. However, data could not tell us anything without processing. We often flooded

by data but lack of the information. Data mining has attracted increasing interests in recent years trying to find the underlying models or patterns of the data, and making use of the found models and patterns [2].

Data mining has been used in a broad range of applications [3]. More and more leading-edge organizations are realizing that data mining provide them the ability to reach their goals in customer relationship management, risk management, fraud and abuse detection, and e-business etc.

Frequent pattern mining [3] and Association rule mining (ARM) [3] plays a very important role in data mining. Semi-structured data refers to set of data in which there is some implicit structure that is generally followed, but no enough of a regular structure to qualify for the kinds of management and automation usually applied to structured data. Examples include the World Wide Web, bioinformatics databases and data ware housing.

Currently many websites are built with HTML tags, one problem associated with retrieval of data from web documents of HTML is that they are not structured in traditional databases because the Web pages created using HTML are semi structured thus making querying more difficult than with well-formed database containing schemas and attributes with defined domains. The concepts of XML have brought convenience for it.

Association rule mining is a process of mining data as a set of rules from a transactional database which support the minimum support and confidence. Association rule of data mining involves picking out the unknown inter-dependence of the data and finding out the rules between those items. Association Rule Mining plays a key role in the process of mining data for frequent pattern matching. A frequent pattern is a pattern (ie. a set of items, substructures, subsequences etc.) that occurs frequently in a dataset. Frequent item sets play an essential role in many data mining tasks that try to find interesting patterns from databases, such as association rules, correlations, sequences, episodes, classifiers, clusters and many more of which the mining of association rules is one of the most popular problems.

We start this review with a discussion on mining in semi-structured data and explore the various challenges and look into the algorithms for mining semi-structured data.

II. BACKGROUND

Association rule mining is a process of mining data as a set of rules from a transactional database[4] which support the minimum support and confidence.

Support: The rule (A=>B) holds in the transaction set D with support[4] *s*, where *s* is the percentage of transactions in D containing A Ū D.

Support(A=>B) = P(A Ū D)

Confidence: The rule (A=>B) has confidence[4] *c* in the transaction set D, where *c* is the percentage of transactions in D containing A that also contains B.

Confidence (A=>B) = P(B | A)

In general, association rule mining can be viewed as a two-step process. (i) Generating all item sets having support factor greater than or equal to, the user specified minimum support. (ii) Generating all rules having the confidence factor greater than or equal to the user specified minimum confidence.

Normally there exist two categories of data mining. (i) Descriptive data mining (ii) Predictive data mining. For carrying out summarizations or generalizations descriptive data mining is used. Whereas for finding out the inference or predictions, Predictive data mining is used. Association rule mining falls under the descriptive category. Association rules aims in extracting important correlation among the data items in the databases.

(A) Semi-Structured Data Mining Techniques

In this section, we briefly describe key semi-structured data mining algorithms that have been developed to handle data which does not have rigid structure.

a) Association Rules Association rule mining is a process of mining data as a set of rules from a transactional database which support the minimum support and confidence. Algorithms for mining association rules from relational data have been well developed. Several query languages have been proposed, to assist association rule mining. Mining semi-structured data, for example XML data has received little attention, as the data mining community has focused on the development of the techniques for extracting common structure from heterogeneous XML data. The straight forward approach for association rule mining from XML data is to map the XML documents to relational data model and to store them in a relational database. This allows us to apply the standard tools that are in use to perform the rule mining from relational databases.

b) Clustering Clustering (clustering) is to group objects of a database into multiple clusters or classes (cluster) so that objects in the same group have a large similarity (similarity) and objects in different groups have a large dissimilarity (dissimilarity). Clustering in data mining is a useful technique for discovering interesting data distributions and patterns in the underlying data. Clustering is also helpful for categorizing www documents, grouping genes and proteins that have similar functions or the detection of seismic faults by grouping the entries in an earthquake catalog.

The algorithms used for frequent pattern mining is divided into two categories: Apriori based algorithms and tree-structure based algorithms. The apriori-based algorithms use a generate-and-test strategy (i.e.) they finds frequent patterns by constructing candidate items and checking their support counts or frequencies against the transactional database.

The tree-structure based algorithms follows a test only approach (i.e.) there is no need to generate candidate items and tests only the support counts or frequencies. Examples are FP-tree and FP-growth, CAN-tree etc.

The eXtensible Markup Language (XML) 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. It can consist 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.

II. LITERATURE SURVEY
III.

(A) FP-growth algorithm

In 2000, Han et al. proposed the FP-growth[5] 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.

FP-tree structure: The FP-tree structure has sufficient information to mine complete frequent patterns. It consists of a prefix tree 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.

Construction of FP-tree: FP-growth has to scan the TDB *(*Transactional Database*)* 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.

Advantages and Disadvantages

This method is advantageous because, it doesn't generate any candidate items. It is disadvantageous because, it suffers from the issues of special and temporal locality issues.

(B) **FP-Tree (Frequent Pattern Tree)**

A tree structure in which all items are arranged in descending order of their frequency or support count. After constructing the tree, the frequent items can be mined using FP-growth.

(a) Creation of FP-Tree

First Iteration: Consider a transactional database which consists of set of transactions with their transaction id and list of items in the transaction. Then scan the entire database. Collect the count of the items present in the database. Then sort the items in decreasing order based on their frequencies (no. of occurrences).

(b)Second Iteration

Now, once again scan the transactional database. The FP-tree is constructed as follows. Start with an empty root node. Add the transactions one after another as prefix subtrees of the root node. Repeat this process until all the transactions have been included in the FP-tree. Then construct a header table which consists of the items, counts and their head-of-node links.

(c)Finding Frequent Patterns from FP-Tree

After the construction of FP-tree, the frequent patterns can be mined using an iterative approach FP-growth. This approach looks up the header table and selects the items that support the minimum support. It removes the infrequent items from the prefix-path of an existing node and the remaining items are considered as the frequent itemsets of the specified item.

Advantages and Disadvantages

This method is advantageous because, it doesn't generate any candidate items. It is disadvantageous because, it suffers from the issues of special and temporal locality issues.

(C) **CAN-Tree (CANonical Tree)**

A tree structure that arranges or orders the nodes of a tree in some canonical order. It follows a tree-based incremental mining approach. Like FP-tree approach, there is no need to rescan the transactional database when it is updated. Because of following the canonical order, frequency changes (if any) due to incremental updates like insertion, deletion and modification of the transactions will not affect the ordering of the nodes in CAN tree[3]. After constructing the CAN tree, we can mine the frequent patterns from the tree.

Finding Frequent Patterns from CAN-Tree*:* After constructing the CAN-tree, we have to mine the frequent patterns by traversing the tree in a upward direction. This can be done similar to FP-growth by constructing a header table and finding only the frequent items.

3.3. Advantages and Disadvantages

This method is advantageous because it supports incremental updates without any major changes in the tree.

(D) **COFI-Tree**

COFI[6] is much faster than FP-Growth and requires significantly less memory. The idea of COFI is to build projections from the FP-tree each corresponding to sub-transactions of items co-occurring with a given frequent item. These trees are built and efficiently mined one at a time making the footprint in memory significantly small.

The COFI algorithm generates candidates using a top-down approach, where its performance shows to be severely affected while mining databases that has potentially long candidate patterns that turns to be not frequent, as COFI needs to generate candidate sub-patterns for all its candidates patterns and also build upon the COFI approach to find the set of frequent patterns but after avoiding generating useless candidates.

Creation of COFI-Tree: The basic idea of our new algorithm is simple and is and is based on the notion of maximal frequent patterns. A frequent item set X is said to be maximal if there is no frequent item set X' such that X€ X'.

Frequent maximal patterns are a relatively small subset of all frequent item sets. In other words, each maximal frequent item set is a superset of some frequent item sets. Let us assume that we have an Oracle that knows all the maximal frequent item sets in a transactional database. Deriving all frequent item sets becomes trivial. All there is to do is counting them, and there is no need to generate candidates that are doomed infrequent. The oracle obviously does not exist, but we propose a pseudo-oracle that discovers this maximal pattern using the COFI-trees[4] and we derive all item sets from them.

CATS[10] algorithms enable frequent pattern mining with different supports without rebuilding the tree structure. The algorithms allow mining with a single pass over the database as well as efficient insertion or deletion of transactions at any time.

CATS Tree and CATS Tree algorithms. Once CATS Tree is built, it can be used for multiple frequent pattern mining with different supports. CATS Tree and CATS Tree algorithms allow single pass frequent pattern mining and transaction stream mining. In addition, transactions can be added to or removed from the tree at any time.

Finding Frequent Patterns from CATS-Tree: After constructing the CATS tree, the frequent patterns can be found by following the frequency of an item by considering its both upward and downward paths.

Advantages and Disadvantages

This method is advantageous because it requires only one scan of the database and the trees are ordered according to their local frequency in the paths.

It is disadvantageous because lot of computation is required to build the tree.

(E) IMPROVED FREQUENT PATTERN TREE (FP -TREE)

It is proposed to recover the weakness of some traditional data mining algorithm. For example: Apriori[7]. It compresses a large data set into a structured and compact data structure, known as FP-Tree. FP –Trees follows the divide and conquers methodology. The root of the FP-tree is labeled as "NULL". A set of item is defined as a child of the root. Traditionally a FP tree consists of three fields- Item name, node link and count. To help the tree traversal, a header table is used. It decomposes the data mining task in to smaller parts.

The main idea is, to construct an efficient FPtree and mining frequent item set algorithm (MFI algorithm). This is because the FP-tree is highly condensed and surely it is smaller than the main database (for example, the transaction database as described as table 2) .It will reduce the expensive database scan in mining method. After forming a FP tree, the frequency of the FP-tree (F) is used as an input for MFI[12] algorithm. It will generate the frequent item sets. These frequent item sets will be used to get strong association rules.

Algorithm 1*:* Construct improve FP-Tree

Input : Transaction database.

Output **:** Improved FP-tree, Frequency of each item in FPtree, and the Stable count.

Step1 : Sort the items in the transaction database in descending order, according to the number of occurrence in transaction.

Step 2 : Create the root of the tree R. Since it is a prefix tree, R= NULL.

Step 3 : For each transaction in transaction database **do**

Step 4 : Let a descending ordered transaction is represented by [p|Q].where p is the first item and Q is rest of the items in transaction.

Step 5 : If p= the most frequent item, do the following step 6 and step7.Otherwise go to step 8.

Step 6 : If the root R has a direct child node M, such that M's item_name = p's item_name, Increase the frequency of the item M, denoted as G.M by 1. Transfer the root from R to p.

Step 7 : For each item in Q do the following steps up to Q is empty.

[a] Create a new node for each new item from root.

Increase the frequency of the new item, G. new item by 1. Transfer the root to new node.

[b] If an item, Qi has no single edge from the current root to its node, Where Qi is an item in Q and already exists in FP-tree. Then Qi is a spare item.

Store Qi with frequency 1 in spare table (Stable)

Step 8 : For each item in transaction [p|Q]

Store all items in Stable with frequency 1.

Step 9 : Calculate the total frequency of each item in stable, called Stable count.

Step 10: Output the FP-tree, frequency of each item in FP-tree and total frequency of Stable count

In above algorithm, maximum attention on the most frequent item, because most of the relations are related with that particular item. So when use the descending ordered transactions, then the most frequent item come first and the tree is formed based on that. The root of the tree is changed with the transactions (step 6 of algorithm 1).As the root is changing with the coming items, so, the correlation is more efficiently visible. Algorithm 1 compares items and classify them (step 5).This classification decides which is spare item (case 1 in definition 1).It enhances the efficiency of the FPtree. After one node is created the other nodes will be created according to the frequency of occurrence of each item (step 7 of algorithm 1). If the same element occurs then the frequency of that item is increased by 1(step 6 of algorithm 1).

Algorithm 2 For Frequent Item set

```
Mining Frequent Item set (FP-tree, S, C, F, R)
for each item i in FP-tree where (i! = R) do
if i.S = i.F then \* i.S is the support of the item
And i.F is frequency of the item in FP tree */
frequency of the frequent item set, K = i.F
Generate item set, $P = (i\ \bar{U}\ \mu)$ with the
frequency value of the tree.
Frequent item set is written in {P: K} format.
else if i.S > i. F then
Frequency of frequent item set K = i.F + C
Generate item set, $P = (i\ \bar{U}\ \alpha)$
else Frequency of the frequent item set K = i.F
Generate item set, $P = (i\ \bar{U}\ \beta)$
: end of for
```

When the database is large, the above algorithm may suffer the problem of memory scarcity. The algorithm will improve if it solves the memory problem without any preprocessing or postprocessing. And another method is to improve Bitwise And Operation on the binary string, or replace it by some more effective techniques.

To date, the famous Apriori algorithm to mine any XML document for association rules without any pre-processing or post-processing has been implemented. 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. First Frequent Pattern-growth[9], 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 can be used.

IV. CONCLUSION

Data Mining is referred to as Knowledge Discovery in Databases. It deals with issues such as representation schemes for the concept or pattern to be discovered, design of appropriate functions and algorithms to find patterns. However data on the web and bioinformatics databases often lack such a regular structure called semi-structured. This survey papers gives a brief survey of XML data mining using association rules and fast frequent pattern in various fields, the modifications made to the association rules according to the applications they were used and its effective results. Thus association rules prove themselves to be the most effective technique for frequent pattern matching over a decade. 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. However, the structure of the XML data can be more complex and irregular than that. Association Rule Mining plays a key role in the process of mining data for frequent pattern matching. 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. This paper shows a review of XML data mining using Fast Frequent Pattern mining in various domains.

References

i. Jaiwei Han and Micheline Kamber, “Data Mining Concepts and Techniques”, Second Edition , Morgan Kaufmann Publishers.

ii. Jayalakshmi.S, Dr k. Nageswara Rao, “Mining Association rules for Large Transactions using New Support and Confidence Measures”, Journal of Theoretical and applied Information Technology, 2005.

iii. Muthaimenul Adnan and Reda Alhajj, “A Bounded and Adaptive Memory-Based Approach to Mine Frequent Patterns From Very Large Databases” IEEE Transactions on Systems Management and Cybernetics- Vol.41,No. 1,February 2011.

iv. W. Cheung and O. R. Zaiane, “Incremental mining of frequent patterns without candidate generation or support constraint,” in Proc. IEEE Int.Conf. Database Eng. Appl., Los Alamitos, CA, 2003, pp. 111–116.

v. C. K.-S. Leung, Q. I. Khan, and T. Hoque, “Cantree: A tree structure for efficient incremental mining of frequent patterns,” in Proc. IEEE Int.Conf. Data Mining, Los Alamitos, CA, 2005, pp. 274–281.

vi. M. El-Hajj and O. R. Zaiane, “COFI approach for mining frequent itemsets revisited,” in Proc. ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery, New York, 2004, pp. 70–75.

vii. Savasere, E. Omiecinski, and S. B. Navathe, “An efficient algorithm for mining association rules in large databases,” in Proc. Int. Conf. Very Large Data Bases, 1995, pp. 432–444.

viii. C.K.-S. Leung. Interactive constrained frequent-pattern mining system.In Proc.IDEAS 2004, pp. 49–58.

ix. J. Han, J. Pei, Y. Yin, and R. Mao . “Mining frequent patterns without candidate generation: a frequent-pattern tree approach”, Data Mining and Knowledge Discovery, 8(1), pp. 53–87, Jan. 2004.

x. Jacky W.W. Wan , Gillian Dobbie, Mining association rules from XML data using XQUERY.

xi. D. Braga, A. Campi, M. Klemettinen and P.L Lanzi. Mining association rules from XML data. In Proceedings of the 4th International conference on data warehousing and knowledge discovery. France 2002.

xii. A.B.M.Rezbaul Islam, Tae-Sun Chung, An Improved Frequent Pattern Tree Based Association Rule Mining Technique, IEEE, 2011 R. Agrawal, T. Imielinski, and A. Swami.. “Mining association rules between sets of items in large databases”. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington, DC, May 26-281993.