XML data mining, Part 1: Survey several approaches to XML data mining

Examine and address the challenges of static and dynamic documents

Laura Irina Rusu, PhD MACS CP

May 02, 2012

(First published November 15, 2011)

XML is used for data representation, storage, and exchange in many different arenas. This series explores one facet of XML data analysis: XML data mining. In this first article, get an introduction to some techniques and approaches for mining hidden knowledge from XML documents. Learn about mining data, the hierarchical structure of the information, and the relationships between elements. Subsequent articles will cover mining XML association rules and clustering multi-version XML documents.

13 Dec 2011 - Added sidebars with a link to Part 2 of this article in the Introduction, Summary, and Resources.

02 May 2012 - Added links to Part 3 of this article to sidebars in the Introduction and Summary, and in Resources.

View more content in this series

Introduction

XML has increasingly become the language of choice for data representation, storage, and exchange in many domains—from technology to financial services, medical systems, bioinformatics, aviation, defense, and so on. There has been a rapid increase in the amount of information being represented in XML. Many folks are searching for sensible techniques to solve issues associated with XML document storage and analysis. In this three-part series, explore certain facets of XML data mining.

In this article, learn about the research and techniques available for XML data mining. Explore the challenges in mining association rules from static and dynamic XML documents.

Other articles in this series

- Part 2, Mining XML association rules
- Part 3, Clustering XML documents to improve data mining

In this article, learn about the research and techniques available for XML data mining. Explore the challenges in mining association rules from static and dynamic XML documents.
Subsequent articles in this series will cover mining XML association rules and clustering multi-version XML documents.

**XML mining**

<table>
<thead>
<tr>
<th>Frequently used abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>W3C: World Wide Web Consortium</td>
</tr>
<tr>
<td>WSDL: Web Services Description Language</td>
</tr>
</tbody>
</table>

XML mining includes mining both the structure and the contents from XML documents. *Mining of structure*, which is essentially mining the XML schema, includes intra-structure mining (mining the structure inside an XML document) and inter-structure mining (mining the structures between XML documents). *Mining of content* involves content analysis and structure clarification. Content analysis is concerned with analyzing texts within the XML document. Structural clarification is concerned with determining the similar documents based on their content.

**Static** XML documents do not change their content and structure over time. For example, an XML document containing details of papers presented at a conference is a static document. **Dynamic**, or multi-versioned, XML documents change their structure or content over time. For example, if the content of an online bookshop were represented in XML format, it would change daily based on e-customer behavior.

The main characteristics differentiating static from dynamic XML documents are:

**Representation of temporal coverage**

A static XML document does not contain an element indicating how long the document is valid. Conversely, a dynamic XML document inherently contains at least one element that indicates the temporal coverage of the specific version of the document.

**Persistence of the rendered information**

Once it has been created, the information in a static XML document is always valid. Conversely, a version of a dynamic XML document is valid only for the period specified by the temporal elements. Once a new version emerges, the information contained in the previous version is superseded.

*Figure 1* shows an example of a schema for a static XML document in WSDL format.

**Figure 1. Schema for a static XML document**

![](image)

The content of an XML document instance based on the schema in *Figure 1* will never change because, once published, the papers presented at that particular conference will not be modified.
Figure 2 shows a schema example for a dynamic XML document in WSDL format.

**Figure 2. Schema for a dynamic (multi-versioned) XML document**

![Schema for a dynamic (multi-versioned) XML document](image)

In Figure 2, the `<AsAtDate>` node contains the validity period for the data in each XML document version. Instead of using a single date node to specify a validity time, you might use two date nodes to specify a validity date range (for example, `<from_date>` and `<to_date>`). Generally, changes to XML documents make them dynamic. Given an initial version of an XML document, each set of changes applied to it produces a new XML document. The new XML document will be considered a new version of the old one.

**Mining static XML documents**

This section reviews several approaches for mining association rules from static XML documents and for clustering static XML documents.

**Association rules from static XML documents**

Most of the work with mining association rules from static XML documents uses XML-oriented algorithms based on the Apriori algorithm. There are several non-Apriori-based approaches, however.

### The Apriori algorithm

The concept of association rules was introduced in 1993 and described in more detail in 1994 (see Related topics). For example, an association rule is: "If the user buys product A, he or she also buys product B, and this happens in more than 80% of transactions." The support of the rule is 80%, which means that 80% of transactions contain both items A and B.

The algorithm to find the rules sets the minimum support and confidence required at the beginning of the mining process. Then, all the large $k$-itemsets are determined, starting from $k = 1$ (itemsets with only one item) and looping through the set of transactions to calculate the support and the confidence. If they are not validated against the minimum required, the $k$-itemsets are considered not large and are removed.

This algorithm has become known as the Apriori algorithm and has been applied to many subsequent approaches to discover association rules.

Discovering association rules involves finding the interesting relationships among items that appear together in a given data set. The relationship should occur often enough to be treated as a rule; the rule has to have support and confidence with regard to the data set.

If the concept of association rules is extended to XML documents, you're seeking relationships between parts of the XML documents. A part of an XML document can be either a simple node
or a complex node, which consists of other simple nodes. Any complex node can be seen as a tree within the bigger tree that is the whole XML document. Hence, the task of mining association rules in an XML document entails finding relationships between subtrees (substructures) of XML documents. The list below summarizes three different methods.

- One XML association rule extraction algorithm proposed by Wan and Dobbie (see Related topics) is actually an implementation of the Apriori algorithm using the XQuery language. Using this algorithm, XML data can be mined directly, without the need to preprocess the XML document. Mapping to a different format is not required. However, discovering large itemsets in the mining process is slower when using the XQuery implementation than when using a C++ implementation. In the XQuery implementation, the update capabilities that the algorithm requires are limited.

- A method proposed by Daniele Braga and colleagues (see Related topics) is based on the Apriori algorithm and mines the association rules in three major steps: preprocessing data, extracting association rules, and post-processing association rules. The authors introduce the concepts of:
  - Context of the association rules
  - Context selection
  - Body
  - Head

In a relationship $X \rightarrow Y$, $X$ is the body and $Y$ is the head of the association rule. Body and head are always defined with respect to the context of the rule. The support and the confidence will be calculated and relevant only with respect to the established context. Figure 3 shows an example of an XML document represented as a tree with the identified context, body, and head.

**Figure 3. XML document with context, body, and head**

- A technique proposed by Ling Feng and colleagues (see Related topics), which is not based on the Apriori algorithm, proposes a different mapping of the concepts of transaction and item to the tree-like structure of the XML documents. The goal is to discover association rules from a collection of XML documents rather than from a single document. Hence, each XML
document (tree) corresponds to a database record (transaction), where each XML fragment (subtree) corresponds to an item in the transaction. This work tries to discover association rules among trees in XML documents rather than among simple-structured items. Each tree is named a \textit{tree-structured item} and is a rooted, ordered tree with nodes classified into basic nodes (no edges emanate from them) and complex nodes (internal nodes with one or more edges emanating from them).

The main difference between the Apriori-based and non-Apriori-based approaches is in the way they use the notions of transaction and item. Although the Apriori-based algorithms extract the items to be mined as a list of nodes by querying the XML document for a specific path, the non-Apriori algorithms see each subtree (substructure) in the XML documents as an item.

Various algorithms allow different degrees of complexity (different levels of nesting) in the mined documents. It is also possible that you need to mine only a single XML document or to extract association rules from multiple XML documents.

The Apriori-based XML association rules mining techniques described here work with XML documents of any complexity as long as the structure is known in advance. For XML documents that are complex and irregularly structured, consider using a set of transformations on the XML data to simplify identifying the mining context. In some of the algorithms, you need to define the context, body, and head of the association rules at the beginning of the algorithm. The methods, their strengths, and their weaknesses are summarized in Table 1.

**Table 1. Mining association rules from static XML documents**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wan &amp; Dobbie, 2003; Wan &amp; Dobbie, 2004</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Braga et al., 2002; Braga et al., 2003</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Feng et al., 2003</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Clustering static XML documents**

Static XML documents can be clustered based on the type of similarity—structural, semantic, or combined—each technique employs.

Clustering based on structural similarity focuses on grouping XML documents into clusters based on the similarity between their structures. Structural approaches include:

- The S-GRACE algorithm (see Related topics) clusters XML documents by structure using \textit{structural graphs} (s-graphs). An s-graph is a directed graph that provides a summary of the nodes and edges in the XML documents together with the relationships between parent and
child nodes. S-GRACE applies a hierarchical clustering algorithm on the s-graphs extracted from the given set of XML documents.

- A generic, agglomerative hierarchical clustering algorithm proposed by De Francesca and colleagues (see Related topics) addresses the XML clustering problem in a parametric way. Given a suitable distance measure, the algorithm finds optimal cluster representatives. A **cluster representative** is an XML document that mirrors the structural content of the cluster. Hence, the task is mainly to calculate a cluster representative through tree matching, tree merging, and pruning of the merged tree.

In addition to structural information, each node has a semantic meaning. You can cluster the XML documents based on both structural and semantic similarity.

- Yoon proposed an approach (see Related topics) that breaks down each XML document that needs to be clustered into *ePaths*. ePaths are full paths starting from the root of the XML document where the most nested element can only be a simple element (containing a value and no children elements). A three-dimensional bitmap index called a **BitCube** is then built. The dimensions are the list of documents, the list of ePaths extracted from the documents, and the list of words extracted from the simple content of all ePaths. From this cube, you can then determine which documents have certain words associated with them (among other things) and speed up the retrieval process when the user searches by specific terms.

- A technique offered by Shen and Wang (see Related topics) extracts macro-paths from the XML documents, which may or may not be based on an XML schema, where each macro-path has three elements: a path sequence, an attribute sequence, and a content sequence. The degree of similarity between these macro-paths indicates the degree of similarity between the XML documents analyzed.

Distance-based clustering techniques are an extension of the structural methods because they also consider both structural and semantic features of the given set of XML documents to cluster. Distance-based clustering techniques have a specific characteristic: They use the **tree edit distance** concept.

**Tree edit distance**

Given two XML documents represented as trees, the edit distance between them is the set of edit operations with the lowest total cost that is able to transform one document into the other. The difference between distance-based clustering approaches is in the way each defines the acceptable edit operations for edit tree distance.

- A technique from Theodore Dalamagas and colleagues (see Related topics) views XML documents as ordered trees. Their algorithm reduces nesting and repetition then calculates a tree edit distance between each pair of structural summaries. Clustering is based on the distance between structural summaries.

- A method by Nierman and Jagadish (see Related topics) calculates the distance between two XML documents by using five different operations on an XML tree: *relabel*, *insert*, *delete*, *insertTree*, and *deleteTree*. The last two operations allow the cutting and pasting of whole sections of an XML document. Allowable sequences of edit operations can also be defined...
and used by a dynamic programming algorithm to calculate the distances between pairs of XML documents in the given set.

- A more recent technique from Xing, Xia, and Guo (see Related topics) calculates the similarity between two XML documents by calculating the edit distance between XML documents and their schema. A schema is extracted from each document and the distance between two XML documents is then calculated as the average of the distances between each XML document and the schema extracted from the other XML document in the pair.

Table 2 summarizes the approaches involving clustering static XML documents.

### Table 2. Clustering static XML documents

<table>
<thead>
<tr>
<th>Works</th>
<th>Calculated distance as a set of edit operations</th>
<th>Assesses structural similarity only</th>
<th>Assesses structural and semantic similarity</th>
<th>Assesses semantic similarity only</th>
</tr>
</thead>
<tbody>
<tr>
<td>De Francesca et al., 2003</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Liang et al., 2004</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Yoon et al., 2001</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Shen &amp; Wang, 2003</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Nierman &amp; Jagadish, 2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Dalamagas et al., 2004</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Xing et al., 2007</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

### Mining dynamic XML documents

Mining dynamic XML documents brings new challenges to developers. In real-world applications, the differences between consecutive versions of an XML document fluctuate, so the mining techniques for static XML documents cannot be replicated for dynamic XML documents.

### Discovering association rules from dynamic XML documents

Discovery of association rules from multi-versioned XML documents is still in its infancy. The Weighted-FPgrowth algorithm (see Related topics) addresses the issue. This algorithm extracts the association rules from XML structural deltas, or XSD-ARs, in three steps:

1. Build the structural deltas database.
2. Discover frequent subtree patterns.
3. Extract association rules.

XML structural deltas are structural differences between consecutive versions of a dynamic XML document. The Weighted-FPgrowth algorithm also defines the concepts of degree of change, frequency of change, and frequent subtree pattern.

I propose two different ways to mine association rules from dynamic XML documents (see Related topics):

XML data mining, Part 1: Survey several approaches to XML data mining
• Interesting knowledge (in this case, association rules) from the collection of historic versions of the XML documents. Consider the changes, in both structure and content, affecting the multi-versioned XML documents in a given period of time. You can dynamically determine the most current valid association rules between elements or substructures of the XML documents involved without running the full discovery algorithm on the latest XML documents versions.
• Association rules extracted from the actual changes between versions of XML documents. Look into the actual changes between versions of XML documents (stored as delta documents or a consolidated delta document) and find the associations between them.

Clustering dynamic XML documents

There has not been much work involving clustering dynamic XML documents. Two possibilities are outlined below.

• One technique, which my colleagues and I proposed (see Related topics), is to reassess the distances between clustered dynamic XML documents after they change and calculate the effects of the changes on the distances before documents have changed.
• Another technique, proposed by Nayak and Xu (see Related topics), is to cluster series of XML documents. Given a series, or streams, of XML documents, a distance is calculated between each incoming XML document and the existing clusters using the level structure. This distance is determined by matching the nodes from the incoming document to the nodes of the existing clusters. In this way, the similarity is determined at the cluster level, rather than pair-wise, for individual documents in the clusters.

Table 3 summarizes the approaches for mining dynamic XML documents.

Table 3. Mining dynamic XML documents

<table>
<thead>
<tr>
<th>Works</th>
<th>Mine association rules from multiple versions</th>
<th>Mine association rules from deltas</th>
<th>Cluster series of XML documents</th>
<th>Cluster multi-versioned (dynamic) XML documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen et al., 2004</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Nayak and Xu, 2006</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Rusu et al., 2006a;</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Rusu et al., 2006b;</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Rusu et al., 2008</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

Summary

Other articles in this series

• Part 2, Mining XML association rules
• Part 3, Clustering XML documents to improve data mining

In this article, you learned about techniques in XML data mining to discover association rules from XML documents or to cluster XML documents based on structure or content. Most of the
techniques deal with static XML documents. However, the pervasive dynamic information available nowadays requires techniques that can also work with dynamic (multi-versioned) XML documents.

Stay tuned for Part 2 in this series, which will go into detail about mining XML association rules.
Related topics

- **Part 2: Mining XML association rules**: Dig into XML data mining, a facet of XML data analysis. Learn about static and dynamic XML association rules and how to create version-based association rules when the mined XML documents change in Part of this three-part series.
- **Part 3: Clustering XML documents to improve data mining**: Check out a non-redundant methodology to recalculate the new clusters of XML documents after changes. Work through a step-by-step case example to understand the technique and apply it in the third part of this series.
- **Extracting Variable Knowledge from Multiversioned XML Documents** (L.I. Rusu, W. Rahayu, and D. Taniar, 2006a): Read about a novel technique of determining how the knowledge discovered from initial XML documents changes in time when the documents’ content and/or structure fluctuates.
- **Mining Changes from Versions of Dynamic XML Documents** (L.I. Rusu, W. Rahayu, and D. Taniar, 2006b): Learn about an approach for mining association rules from changes between versions of dynamic XML documents, in a simple manner, by using the information contained in the consolidated delta.
- **Intelligent Dynamic XML Documents Clustering** (L.I. Rusu, W. Rahayu, and D. Taniar, 2008): Learn a time-efficient technique to reassess pair-wise distances between clustered dynamic XML documents which change in time, without performing redundant calculations but considering the previously known distances and the set of changes which might have affected the documents’ versions.
- **Data mining and XML documents** (R. Nayak, R. Witt, and A. Tonev, 2002): Examine taxonomy of XML mining as a stepping stone to further XML mining research.
- **Fast Algorithms for Mining Association Rules** (M.H. Marzahuy and A.A. Mitwaly, 2005): Explore the problem of discovering association rules between items in a large database of sales transactions. To solve the problem, two new algorithms that are fundamentally different from the known algorithms are presented.
- **Mining Association Rules from XML data** (D. Braga, A. Campi, M. Klemettinen, and P.L. Lanzi, 2002): Review the association rules from native XML documents and discuss the new challenges and opportunities that this topic sets to the data mining community.
- **Discovering Interesting Information in XML with Association Rules** (D. Braga, A. Campi, and S. Ceri, 2003): Get an introduction to association rules for XML data. The article proposes a new operator, based on XPath and inspired by the syntax of XQuery, which lets you express complex mining tasks compactly and intuitively.
- **Mining Association Rules from Structural Deltas of Historical XML Documents** (L. Chen, S.S. Bhowmick, and L.T. Chia, 2004): Read about the sequence of changes to the structures of an XML document to find out which subtrees in the XML structure frequently change together.
- **Distance-based Clustering of XML Documents** (F. De Francesca, G. Gordano, R. Ortale, and A. Tagarelli, 2003): Learn about a novel methodology for clustering XML documents, focusing on the notion of XML cluster representative (a prototype XML document subsuming the most relevant features of the set of XML documents within the cluster).
• **An XML–Enabled Association Rules Framework** (L. Feng, T. Dillon, H. Wiegand, and E. Chang, 2003): Check out an extended XML-enabled association rule framework, which is flexible and powerful enough to represent both simple and complex structured association relationships inherent in XML data.


• **XCLS: A Fast and Effective Clustering Algorithm for Heterogeneous XML Documents** (R. Nayak and S. Xu, 2006): Read about a novel clustering algorithm to group the XML documents by similar structures.

• **Evaluating Structural Similarity in XML Documents** (H. Nierman and V. Jagadish, 2002): Learn about a tree edit distance based measure suited to this task, taking into account XML issues such as optional and repeated sub-elements.

• **Clustering Schemaless XML Documents** (Y. Shen and B. Wang, 2003): Explore the issue of semantically clustering the increasing number of the schemaless XML documents.

• **Clustering XML Documents Based on Structural Similarity** (G. Xing, Z. Xia, and J. Guo, 2007): Learn about a framework for clustering XML documents based on structural similarity between XML documents.

• **Extracting Association Rules from XML documents using XQuery** (J.W. Wan and G. Dobbie, 2003): See how any XML document can be mined for association rules using only the query language XQuery without any pre-processing or post-processing.

• **Mining association rules from XML data using XQuery** (J.W. Wan and G. Dobbie, 2004): Learn how extracting association rules from XML documents without any preprocessing or post-processing using XQuery is possible. Analyze the XQuery implementation of the well-known Apriori algorithm.

• **XQuery**: Learn more about this XML Query Language recommended by the W3C.

• **BitCube: A Three-Dimensional Bitmap Indexing for XML Documents** (J.P. Yoon, V. Raghavan, V. Chakilam, and L. Kerschberg, 2001): Read about how to represent and index an XML document collection using a bitmap indexing technique.

• **Mining Association Rules from Structural Deltas of Historical XML Documents** (L. Chen, S. S. Bhowmick, and L.T. Chia, 2004): See how to mine a novel type of association rules from a sequence of changes to XML structure, which are called XML Structural Delta Association Rule (XSD-AR).

• **XML area on developerWorks**: Find the resources you need to advance your skills in the XML arena, including DTDs, schemas, and XSLT. See the XML technical library for a wide range of technical articles and tips, tutorials, standards, and IBM Redbooks.

• **IBM certification**: Find out how you can become an IBM-Certified Developer.

• **developerWorks on Twitter**: Join today to follow developerWorks tweets.

• **IBM product evaluation versions**: Get your hands on application development tools and middleware products.