Mining Semi-structured Data

Olfa Arfaoui\textsuperscript{1} and Minyar Sassi Hidri\textsuperscript{2}

\textsuperscript{1,2}Université de Tunis el Manar, Ecole Nationale d’Ingénieurs de Tunis, BP. 37, Le Belvédère 1002 Tunis, Tunisia
\{olfa.arfaoui, minyar.sassi\}@enit.rnu.tn

Abstract. The need for discovering knowledge from XML documents according to both structure and content features has become challenging, due to the increase in application contexts for which handling both structure and content information in XML data is essential. So, the challenge is to find an hierarchical structure which ensure a combination of data levels and their representative structures. In this work, we will be based on the Formal Concept Analysis-based views to index and query both content and structure. We evaluate given structure in a querying process which allows the searching of user query answers.

Keywords: XML Mining, Formal Concept Analysis, Conceptual Scaling, XQuery, eXist

1 Introduction

The widespread use of XML (eXtensible Markup Language)\textsuperscript{[1,2]} across the Web and in business as well as scientific databases has prompted the development of methodologies, techniques and systems for effectively and efficiently managing and analyzing XML data.

This has increasingly attracted the attention of different research communities, including databases (DB), information retrieval (IR), pattern recognition, and machine learning, from which several proposals have been offered to address problems in XML data management and knowledge discovery\textsuperscript{[3]}.

XML Structure and Content Mining is one of these problems. It has its roots in problems which originally arose from several applications in semi-structured data management, such as querying data sources and query processing.

Hence the recourse to the development of new indexing and querying systems whose aim is to provide a fast and a reliable XML data access because indexing technique influences the reliability of the querying process in terms of research time and treatment queries.

For this, several studies have been introduced\textsuperscript{[4,5,6]}. They belong to both the community of DB and the recent researches in XML language. The main purpose of these methods is to develop indexing approaches own to XML technology.

The main problem of these methods is how to find information in a document while taking into account structure and content. This presents a great challenge.
when we want mining large volumes of XML data. The structural dimension must be taken into consideration for the users’s needs.

However, maintaining the hierarchical structure of elements of an XML document and their order is important to avoid recalculating each time this order and so avoid a sequential access to data in order to determine the relationship between elements.

The recourse to the Formal Concept Analysis (FCA) \[7\] to mining XML documents appears effective to find solutions to the indexing and querying problems while putting into account i) The extraction of the most representative words in the documents (key-words) and their structural information; ii) The structural aspect is assigned to the content and the following questions appear: How we can index the document structure?, How we can connect this structure to the document content? and Depending on what dimension the indexing terms should be weighted?

To answer these questions, the contributions from this work should allow i) The reconstruction of the XML document decomposed in handling structures; ii) The processing of the path expressions on the XML structure; iii) The processing of precise predicates on the XML documents content and iv) The search data by key-words.

In this work, we propose to summarize XML data into a conceptual scaling and generate a generalized view seen as Concept Lattice, a FCA-based structure, to come up with the proposed mining method.

The rest of the paper is organized as follows: section 2 describes and evaluates our mining semi-structured data model. Section 3 concludes the paper and gives future work.

2 Mining Semi-structured Data

2.1 Overview of our Mining XML Data Model

Before querying XML data, we must proceed to index them. Recently, a new indexing approach have been proposed based on FCA and gives answers on several abstraction levels \[10\].

The main idea consists of using FCA based-theory on XML data in order to index them and subsequently facilitate the querying process of such data. The steps which compose our approach are:

- XML tree traversal: is to traverse the XML tree and extract the textual data in the form of a set \(E\),
- Conceptual classification: is to build the concept lattice associated to each parent nodes generated following an ascending traversal of the document,
- Conceptual scaling: the lattice structures obtained are combined into a single structure called nested lattice base on conceptual scaling.

After indexing data, we proceed to the querying step which consists of three steps:
Assembling: is to generalize the different concepts lattices into a generalized one.

Updating: is to transform a user query in a concept and insert it into the structure (concept lattice).

Coursing: is to course the concept lattice for generation of query answers.

Fig. 1 shows the overview of our approach.

2.2 Indexing XML Data

The overall process of indexing step is detailed in [10]. From the nested concepts lattice (Conceptual Scaling) generated from the indexing step, we generalize the nested structure to generalized concept lattice.

Fig. 2 shows an example of an XML document. The data that we may have in our example are extracted from leaves and nodes as follow: Beginner, CSS 2, Daniel Glazman, Eyrolles, Training...XML, Michael J. YOUNG, Microsoft Press, Intermediate, Eng, Training ... ASP.Net, Richard Clark.

Let take \( E = \{D_1, D_2, ..., D_{11}\} \) the data set representing the leaf nodes of the XML tree.

The first step consists of extraction structure data of parent node. According to our example, the first book is a parent node and it has as structural data.

After extraction, Galois lattices associated to each parent node generated following an ascending traversal of the XML tree are constructed.

Table 1 shows an example of formal context of the XML data presented in 2. The processed node is \(<\text{book}>[0]\).

For this node, the construction of the corresponding context which consists of the set of words \( E \) and the set of the child nodes \(<\text{level}>, <\text{title}>, ...\)
Fig. 2. Example of an XML tree.

<author>, <publisher> can started. They represent respectively the set of objects and attributes of this context.

Table 1. Binary context of the node <book>[0].

|       | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 |
|-------|----|----|----|----|----|----|----|----|----|-----|-----|
| <level> | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0   | 0   |
| <lang>  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0   | 0   |
| <title> | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0   | 0   |
| <author>[0] | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0   | 0   |
| <author>[1] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0   | 0   |
| <publisher> | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0   | 0   |

Similarly, <book>[1] and <book>[2] are defined. The nodes presented above have the same parent <bib>. So all the child nodes of these nodes become leaves.

Therefore in the context node <bib>, the lines represent the set of objects and the columns the set of attributes which are respectively the set of parent nodes (<book>[0], <book>[1], <book>[2]) and all the leaf nodes (<level>, <title>, <lang>, <author>, <author>, <Publisher>). Table 2 illustrates this context.

The interest of a concept lattice is to organize information about groups of objects with common properties.

Taking the example of Table 2, \{<book>[0], <book>[1], <book>[2]>\}, \{<level>, <title>\} and \{<book>[0], <book>[1], <book>[2]>\}, \{<publisher>\} are both concepts.
Table 2. Binary context of the root node <bib>.

|       | <book>[0] | <book>[1] | <book>[2] |
|-------|-----------|-----------|-----------|
| <level> | 1         | 1         | 1         |
| <lang>  | 0         | 0         | 1         |
| <title> | 1         | 1         | 1         |
| <author>[0] | 1     | 1         | 1         |
| <author>[1] | 0     | 1         | 0         |
| <publisher> | 1     | 1         | 1         |

The second concept means that objects <book>[0], <book>[1] and <book>[2] have in common the attribute <publisher>.

Several algorithms have been proposed for the construction of concept lattice. Their complexity is exponential and several techniques have been developed to reduce computation time [11].

Fig.3 shows the concepts lattices for <book>[0], <book>[1], <book>[2] and <bib> respectively.

Fig. 3. Concepts lattice for (a) <book>[0], (b) <book>[1], (c) <book>[2] and (d) <bib>.
If we want to index an XML document, it will be necessarily to focus on the content and structure. This is insufficient because the binary context is not satisfactory for indexing multi-valued attributes, hence the use of nested lattice based-structure (conceptual scaling), which aims to extend the interest of the simple Galois lattice.

This structure has been developed by Ganter and Wille [9]. The general process in the Conceptual Scaling begins with the representation of knowledge in a data table with arbitrary values and missing values probably. These data tables are formally described by multi-valued context \( (G, M, W, I) \), where \( G \) is a set of objects, \( M \) is a set of multi-valuated attribute, \( W \) is a set of values and \( I \) is a ternary relation, \( I \subseteq m \times g \times W \), such that for all \( g \in G, m \in M \) there is at most one value \( w \) satisfying \( (g, m, w) \in I \).

Therefore, a multi-valuated attribute \( m \) is generally interpreted as a measurement function (partial) and we write \( m(g) = w \) if and only if \( (g, m, w) \in I \) such as presented in [12].

2.3 Generalizing Concept Lattices

From the nested concepts lattice (Conceptual Scaling) generated from the indexing step, we generalize the nested structure to generalized concept lattice. This lattice structure corresponds to the generalized view.

From this, we will be able to find answers to satisfy a given query. This step involves building the first concept associated with the user query. From this concept, we check the query feasibility. If it is, the concept is built and then inserted into the generalized view.

The answer to the query is then provided by the extraction of objects belonging to the upper bounds of extensions of concepts in the query concept.

For the construction of generalized view, we provide the following structural steps:

- Identification of global concepts: characterization of full nodes of the nested lattice (Conceptual Scaling);
- Calculation of the intention and extension of each concept;
- Calculation of the hedging relationship of the lattice (immediate predecessors of a concept).

2.4 Updating Generalized View

To better explain the steps of evaluation, we consider an example of an XML data and we use XQuery language [14].

Consider the following query: Returns a sequence of elements of type (publisher author) who are children of the first book element type.

According to XQuery language, this query can be rewriting as follows: \( \text{document(bib.xml)/bib/book[1] } //\text{(publisher, author)}; \)

Once the generalized view is built, the search for answers can begin. For this, we define a query which is a concept. The concept extension is sought that all
nodes of the XML document "bib.xml" and the set of objects sought by the query.

After that, the query is parsed and validated. These tokens are provided to XQuery Analyzer. It analyzes tokens and create separate one path: SearchPath, ConditionalPath and ReturnPath. This is illustrated by the following query:

For $b$ in doc (bib.xml)/bib/book ← SearchPath
Where $b$/author = Daniel GLAZMAN← ConditionalPath
Return $b$/book ← ReturnPath;

where:
- **SearchPath()**: is a process which returns the elements in the path specified in the FOR clause.
- **ConditionalPath()**: is a process which returns the subset of elements returned by processForPath() satisfying the condition in the WHERE clause of the XQuery query.
- **ReturnPath()**: is a process which returns the elements (usually descendants of the elements returned by ConditionalPath() process specified by the RETURN clause.

The set of attributes is determined by the following algorithm:

| Algorithm 1 Construction of query concept |
|-------------------------------------------|
| **Require**: XQuery query $Q$            |
| **Ensure**: Query concept $Q = (Q_A, Q_B)$|
| Read the query                           |
| Check for grammatical error              |
| If No-Error-Found then                   |
| - Create the SearchPath                  |
| - Create the ConditionalPath             |
| - Create the ReturnPath                  |
| - Evaluate the query                     |
| End if                                    |
| If $\exists$ SearchPath in PathDictionary then |
| $Q_B ←$ ConditionalValue                 |
| Else                                      |
| - Display Not-Found-Element              |
| End if                                    |

Once defined the concept query $Q$, it is inserted into the concept $T(C_\leq)$ by using the method of incremental construction of Godin [15]. The generalized view obtained corresponds to the concept lattice motion is noted $T_\oplus(C_\leq \oplus Q)$ where $C_\leq \oplus Q$ is the new set of concepts resulting from the insertion of the application in $T(C_\leq)$.

We consider all nodes in the initial generalized view $T(C_\leq)$. The insertion of the query concept $Q$ follows the following properties.
Property 1. We consider all nodes in the initial generalized view $T$. We add the new element $Q$ to all extensions of nodes representing the search path.

Property 2. The new concepts are of form $(A \bigcup \text{Query}, B \bigcap f(\text{Query}))$ for some concepts $(A, B)$. So, we have new sets $A'$ and $B'$. In this case, the concept is called a generator for the new pairs.

Property 3. Each new item in the set $A'$ in $V$ is the result of the intersection of $f(X)$ such as $x \subset X$ with a $B'$ already present in $O$.

Property 4. The son’s concepts of former concepts doesn’t change. The generator is also the only pair that becomes old son a new pair. A new pair may have a son but it is also a new concept.

Property 5. Parents of older concepts that do not generate new concepts remain unchanged. In addition, parents of modified concepts do not change.

The goal is to generate $T_\oplus(C \leq \oplus Q)$ from $T(C \leq)$ and changing $X$ and $X'$. The new concepts are derived from concepts of generators within properties mentioned above.

2.5 Searching Query Answers

Once the concept is inserted into the generalized view, the course of this structure can begin to search answers.

Property 6. An object $o$ is relevant for a given query $Q = (Q_A, Q_B)$ if and only if it is characterized by at least one of the data $Q_B$.

Given a query $Q = (Q_A, Q_B)$, all relevant objects are within reach of $Q$ and their upper bound in generalized view $T_\oplus(C \leq \oplus Q)$ since the intent of each of these concepts is included in $Q_B$.

Let be consider $R_o(Q, C)$ all relevant objects for query $Q$ considered in the set of formal concepts $C$. Intuitively, the search algorithm of relevant objects, try to insert the query concept in $T(C \leq)$ to produce $T_\oplus$. Then, all objects appear in the extension of $Q$ in $T_\oplus$ are inserted into the list.

The following algorithm gives the all steps of this process.

3 Conclusion

The need for a mining model for XML documents becomes important. So, the principal idea of this work is to propose a FCA-based model for indexing and mining both XML structure and content.

For this reason, we have proposed a FCA-based model which aims is to ensure both the indexing and the querying of XML data while achieving a data conceptual classification.
Algorithm 2 Searching query answers

Require: A query $Q = (Q_A, Q_B)$ where $Q_A = \emptyset$ and the generalized view $T(C_{\leq})$
Ensure: $T_\oplus(C_{\leq} \oplus Q)$ and a set of answers $R_\oplus(T_\oplus, C, level)$

1: Build the concept $Q = (Q_A, Q_B)$
2: Insert $Q$ in $T(C_{\leq})$
3: Search in $T_\oplus$ the new concept $Q = (Q_A \cup Q_B')$
4: $level = 0$
5: $UpperBounds(Q, C, level) \leftarrow Q$
6: $R_\oplus(T_\oplus, C, level) \leftarrow \emptyset$
7: repeat
8:   for all $C = (A, B) \in maj(Q, C, level)$ do
9:      if $B \neq \emptyset$ then
10:         $R_\oplus(T_\oplus, C, level) \leftarrow R_\oplus(T_\oplus, C, level) \cup A$
11:     end if
12:   end for
13:   $level \leftarrow level + 1$
14: until $maj(Q, C, level) \leftarrow \emptyset$

The recourse to the FCA, as a solid mathematical foundation, is proved its efficiency in the indexing process of XML documents.

The aim of this process is to facilitate the querying one while ensuring the XML tree traversal, the conceptual classification and the conceptual scaling by generating a nested-based structure.

After indexing both structure and content, the querying process consists of generalizing concepts lattices into a generalized view. After that, a concept XQuery query is defined to be able to insert it in this view.

Then, the coursing process began which permits the searching of answers following user’s queries.

As future work, we propose to i) compare our FCA-based model with other classic mining models, ii) extend this model on a flexible querying process while using fuzzy predicates, iii) extend this model for querying multi-structured documents and finally iv) exploit this approach to facilitate querying XML data and data warehouse, in which a native XML DB stores data and performs multi-dimensional OLAP (On-Line Analytical Processing queries).

References

1. Bradely, N.: The XML Companion. Paperback, Subsequent Edition (2001).
2. Extensible Markup Language (XML) Recommendation, http://www.w3.org/TR/2000.
3. Tagarelli, A.: XML Data Mining: Models, Methods, and Applications. University of Calabria, Italy (2011).
4. Luk, R. W., Leong, H., Dillon, T.S., Shan, A.T., Croft, W.B., Allan, J.: A survey in indexing and searching XML documents. In: Journal of the American Society for Information Science and Technology, vol. 53(3), pp.415–435 (2002).
5. Weigel, F., Meuss, H., Bry, F., Schulz, K. U.: Content-aware data-guides: Interleaving IR and DB indexing techniques for efficient retrieval of textual XML data. In: European Conference on Information Retrieval, pp. 378–393. Sunderland, UK (2004).

6. Li, Q., Moon, B.: Indexing and querying XML data for regular path expressions. In: Proceedings of the 27th International Conference on Very Large Databases Conference, Roma, Italy (2001).

7. Ganter, B., Wille, R.: Formal concept analysis. Mathematical foundations. Springer Verlag, Berlin (1999).

8. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. Ordered sets, vol. 23, Rival Editor, pp. 445–470 (1982).

9. Ganter, B., Wille, R.: Formal concept analysis, Mathematical foundations. Springer Verlag, Berlin (1999).

10. Ayadi, D., Arfaoui, O., Sassi-Hidri M.: Using conceptual scaling for indexing XML native databases. In: The 4th International Workshop on XML Data Management, pp. 309–318 (2012).

11. Li, Q., Moon, B.: Indexing and Querying XML Data for Regular Path expressions. In: Proceedings of 27th International Conference on Very Large Databases, pp. 361–370. Rome, Italy (2001).

12. Kyu, Y., Yoo, S., Yoon, K.: Index Structures for Structured Documents. In: IEEE International Conference on Development and Learning, pp. 91-99 (1996).

13. Valtchev, P.: An algorithm for minimal insertion in a type lattice. Computational Intelligence, vol. 15(1), pp. 63-78 (1999).

14. Chamberlin, D.: XQuery: An XML Query Language. IBM Systems Journal, vol. 41(24), pp. 597–615 (2002).

15. Godin, R., Missaoui, R., Alauoi, H.: Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices. Computational Intelligence, vol. 11, pp. 246–267 (1995).