XML document-grammar comparison: related problems and applications

Joe Tekli, Richard Chbeir, Agma Traina, Caetano Traina

XML Document-Grammar Comparison: Related Problems and Applications

Joe Tekli [1*], ***Richard Chbeir*** [2], ***Agma J.M. Traina*** [1], ***and Caetano Traina Jr.*** [1]

[1] ICMC - Computer Science and Statistics Department, University of Sao Paulo (USP), 13566-590 - São Carlos, SP, Brazil

[2] LE2I Laboratory UMR-CNRS, Department of Computer Science, University of Bourgogne, 21078 Dijon Cedex France [*]

ABSTRACT

XML document comparison is becoming an ever more popular research issue due to the increasingly abundant use of XML. Likewise, a growing interest fosters the development of XML grammar matching and comparison, due to the proliferation of heterogeneous XML data sources, particularly on the Web. Nonetheless, the process of comparing XML documents with XML grammars, i.e., XML document and grammar similarity evaluation, has not yet received the attention it deserves. In this paper, we provide an overview on existing research related to XML document/grammar comparison, presenting the background and discussing the various techniques related to the problem. We also discuss some prominent application domains, ranging over document classification and clustering, document transformation, grammar evolution, selective dissemination of XML information, XML querying, as well as alert filtering in intrusion detection systems, and Web Services matching and communications.

Keywords: XML; Semi-structured data; XML Grammar; DTD; XSD; Structural Similarity; Classification; Clustering; Structure Transformation; Selective dissemination; Grammar evolution.

1. Introduction

XML documents represent hierarchical data instances, made of atomic and complex elements (i.e., containing sub-elements) as well as atomic attributes, incorporating structure and content in one entity (cf. Fig. 1). One of the main characteristics that distinguish XML documents from plain semi-structured data is the notion of XML grammar. An XML grammar (i.e., DTD [15] or XSD [65]) is a set of definitions and declarations for modeling XML documents, defining the elements and attributes of the documents they describe, as well as element/attribute structural positions and the rules they adhere to in the documents [67] (cf. Fig. 2). XML grammars can be viewed as schemas in traditional DBMS, necessary for the efficient indexing, storage, and retrieval of corresponding document instances.

Due to the unprecedented abundant use of XML, XML-based document comparison has become a central issue, thoroughly investigated in the database and information retrieval communities [17, 38, 86]. Likewise, a growing interest is recently underlined in XML grammar matching and comparison [30, 81, 87], with the proliferation of heterogeneous XML data sources, particularly on the Web. Nonetheless, the process of comparing XML documents with XML grammars, i.e., XML document and grammar similarity evaluation, has not yet received strong attention as it should, due to the importance of dealing with semi-structured data.

Performing XML document/grammar comparison, i.e., evaluating the similarity between an XML document and an XML grammar, is useful in various application domains, such as the classification of XML documents against a set of grammars declared in an XML database [12, 63], XML document retrieval [12, 37] (a query being represented as an XML grammar), as well as in the selective dissemination of XML documents (user profiles being expressed as grammars against which the incoming XML document stream is matched) [12].

In this paper, we provide a concise and comprehensive review on the methods related to XML document/grammar comparison. The objective of this study is to briefly describe, compare, and classify the different techniques and methods related to the problem. We also illustrate some of the potential application scenarios that can benefit from XML document/grammar similarity evaluation. To our knowledge, this is the first review study dedicated to the XML document/grammar similarity domain. The remainder of this paper is organized as follows. Section 2 presents the main properties of XML documents and grammars. Section 3 reviews the background and state of the art in XML document/grammar similarity evaluation and related problems. Section 4 develops the main applications and potential uses of XML document/grammar comparison, before concluding in Section 6.

2. Background

This section provides an overview on the basic notions and common properties related to XML documents and XML grammars.

2.1. XML Document Representation Model

XML documents represent hierarchically structured information and are generally modeled as Ordered Labeled Trees (OLTs, cf. Fig. 1.b). In a traditional DOM (Document Object Model) ordered labeled tree [93], nodes

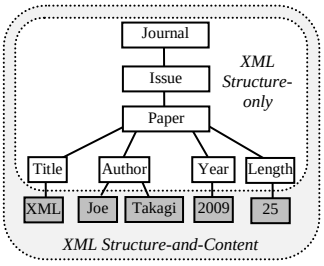
[*] The author is supported in part by the Research Support Foundation of the State of Sao Paulo, FAPESP Post-doctoral Fellowship n# 2010/00330-2.

represent XML elements, and are labeled with corresponding element tag names, organized following their order of appearance in the document. Each edge in the XML tree represents the membership of the element corresponding to the child node, under the element corresponding to the parent node in the XML document. Element attributes mark the nodes of their containing elements. Some studies have considered OLTs with distinct attribute nodes, labeled with corresponding attribute names [63, 99]. Attribute nodes appear as children of their encompassing element nodes, sorted by attribute name, and appearing before all sub-element siblings [63].

In the comparison process, element/attribute values can be disregarded (*structure-only*) or considered (*structure-and-content*), following the requirements of the application scenario at hand (cf. Fig. 1.b). In general, element/attribute values are disregarded when evaluating the structural properties of heterogeneous XML documents, i.e., documents originating from different data-sources and not conforming to the same grammar, so as to perform XML structural classification/clustering [27, 63] or structural querying (i.e., querying the structure of documents, disregarding content [12]). Nonetheless, values are usually taken into account with methods dedicated to XML data warehousing (change management and version control) [20, 25], data integration (providing the user with a unified view of the XML data) [39, 52], and XML *structure-and-content* querying applications [76], where XML content management is required. Here, XML documents tend to have relatively similar structures, and probably conform to the same grammar. With such methods, XML text sequences can be decomposed into words, mapping each word to a leaf node labeled with the respective word [76, 77].

```
<?XML>
<Journal>
  <Issue>
    <Paper Title= 'XML'>
      <Author> Joe Takagi </Author>
      <Length> 25 </Length>
      <Year> 2009 </Year>
    </Paper>
  </Issue>
</Journal>
```

a. XML Document.



b. XML document tree (OLT).

Fig. 1. Sample XML document with corresponding OLT.

Notice that most existing approaches in the context of XML document/grammar comparison disregard element/attribute values (contents), and mainly focus on heterogeneous document structure comparison (as we will show in the following).

2.2. XML Grammars

An XML grammar (i.e., Document Type Definitions - DTD [15], or XML Schema Definition – XSD [65]) is a structure consisting of a set of XML elements, sub-elements and attributes, linked via the containment relation. It identifies element/attribute structural positions, data-types, and the rules they adhere to in the XML document (cf. sample XML grammars in Fig. 2). The structural properties of XML grammar languages are essentially captured by *regular tree languages* [59].

```
<!ELEMENT Journal (Issue*)>
<!ELEMENT Issue (Paper+)>
<!ELEMENT Paper ((Author+ | Publisher), Year, Length?)>
<!ATTLIST Paper Title CDATA>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Length (#PCDATA)>
```

a. XML grammar in DTD syntax.

```
<?XML>
<schema>
<element name = 'Journal'>
  <element name = 'Issue' MinOccurs = '0' MaxOccurs = 'unbound'>
    <element name= "Paper" MaxOcc = 'unbound'>
      <Attribute name= 'Title' type= 'String'/>
      <sequence>
        <choice>
          <element name= "Author" MaxOccurs= 'unbound'/>
          <element name= 'Publisher' type= 'String'/>
        </choice>
        <element name= 'Year' type='gYear'/>
        <element name= 'Length' type='Integer' MinOccurs='0'/>
      </sequence>
    </element>
  </element>
</element>
</schema>
```

b. XML grammar in XSD syntax (allowing a more detailed description of data-types and constraints).

Fig. 2. Sample XML grammars, in DTD and XSD syntaxes.

2.2.1. Origins in Formal Language Theory

In formal language theory [43], a grammar G mainly consists of a set of rules to transform strings. Rules are generally represented in the form: $V \rightarrow w$, where V is a single non-terminal symbol[1], and w is a string of terminal and/or non-terminal symbols (as well as the *empty* symbol). The language $L(G)$, defined based on grammar G, consists of all the possible strings that can be generated following the set of symbols and rules defined in G.

In this context, XML grammars can be viewed as special *regular tree grammars* [22, 59, 61], where non-terminal symbols underline composite element labels, terminal symbols underline simple (leaf node) elements labels or attribute labels, and such as the right hand side of their production rules are made of regular expressions instead of classic strings, i.e., $A \rightarrow r_A$ where r_A is the rule associated with element label A.

Consider for instance the sample XML grammar in Fig. 2.a. The main productions rules describing the structure of the grammar are as follows:

$$V_{Journal} \rightarrow V_{Issue}*$$
$$V_{Issue} \rightarrow V_{Paper}+$$
$$V_{Paper} \rightarrow W_{Paper}, (W_{Author} \mid W_{Publisher}), W_{Year}, W_{Length}?$$

[1] In formal language theory, terminal symbols are those to which we do not separately associate production rules. In other words, terminal symbols cannot be broken do wn to smaller units.

Here, $V_{Journal}$ corresponds to the root element in the grammar, such as $V_{Journal}$, V_{Issue}, and V_{Paper} are non-terminal symbols, representing composite elements, whereas the remaining are terminal symbols, representing simple elements/attributes.

Special production rules are introduced in XML grammar languages to encode XML element data-types (which do not exist in traditional tree languages [43]). This can be done by adding new non-terminal symbols associated to external predicates representing each basic data-type, which allows verifying whether the text (in the XML document) can be converted into a value of the required type.

For instance, data-type rules for elements *Author* and *Length* in Fig. 2.b can be represented as $V_{Author} : D_{String}$ and $V_{Length} : D_{Integer}$, where D_{String} and $D_{Integer}$ underline XSD *String* and *Integer* data-types respectively.

A detailed study highlighting the correlation between XML grammar languages (namely DTD and XSD) and regular tree languages is provided in [59].

2.2.2. Data-types

The DTD language [15], i.e., the basic XML grammar language introduced in the XML specification, allows only a handful of element data-types usually constrained to textual contents (namely PCDATA, Any, and *Composite*). Nonetheless, the XSD language [65], i.e., the main XML grammar language used nowadays, is far more expressive and supports 19 predefined content types (e.g., *Boolean*, *Integer*, *Decimal*, *Date*,…, *Composite*) which could occur in XML documents. XSD also allows the definition and derivation (i.e., extension and/or restriction) of new data-types based on built-in ones (e.g., deriving an *Integer* type whose values are restricted in a given interval).

Consider for instance the sample XML grammar in Fig. 2.b. It contains three *composite* elements: *Journal*, *Issue* and *Paper*, and three simple ones: *Author* of type *String*, Y*ear* of type *gYear* (a special *date* type), and *Length* of type *Integer*, as well as attribute *Title* of type *String*.

2.2.3. Constraint Operators

These are operators employed to specify constraints on the existence, repeatability and alternativeness of XML elements/attributes. With DTD constraint operators, it is possible to specify whether an element is optional ('?'), whether it may occur several times ('*' for 0 or more times, and '+' for 1 or more times), or whether an attribute is optional ("*Implied*") or mandatory ("*Required*"). The XSD language, however, introduces more expressive cardinality operators: *MinOccurs* and *MaxOccurs*, specifying respectively the minimum and maximum number of occurrences an element/attribute can appear in the corresponding XML documents. An element/attribute with no constraints is mandatory and should appear once.

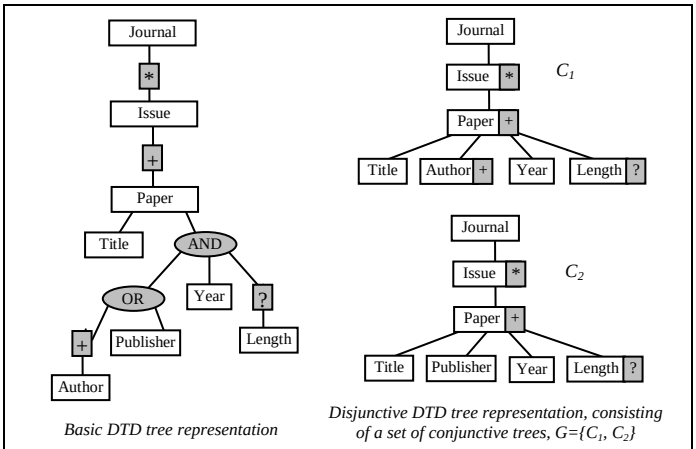
It is also possible to specify whether siblings are alternative w.r.t.[1] each other (using the *Or* operator, represented as '|' in DTDs, and, *choice* in XSDs) such as when one and only one of the concerned elements should appear in the document, or whether they are grouped in a sequence (using the *And* operator, represented as ',' in DTDs, and *sequence* in XSDs) such as when each element is required to appear in the document.

Consider for instance the XML grammars in Fig. 2.a and Fig. 2.b. Here, elements *Issue*, *Paper* and *Author* are repeatable, whereas element *Length* is optional. In addition, elements *Author* and *Publisher* are connected via the *Or* operator, and underline an alternative of elements, whereas their siblings represent sequences of elements.

2.2.4. XML Grammar Representation

While XML documents can be naturally represented as labeled tree structures (cf. Section 2.1), XML grammar representations are usually more intricate. That is due to the various types of constraints associated to elements and attributes, as well as composite element expressions. In practice, XML grammars are abstracted as special tree structures and/or graphs (when recursive definitions come to play) [49, 83], usually simplifying grammar constraints and/or disregarding element/attribute data-types [12, 85], depending on the approach and application at hand.

For instance, Fig. 3 depicts two tree representation variants describing the same sample XML grammar. The tree representation in Fig. 3.a contains special nodes to describe XML grammar constraint operators [12]. The tree in Fig. 3.b follows the disjunctive normal form [76] to represent alternative declarations (resulting from the *Or* operator) as a set of conjunctive elements, integrating cardinality constraints (?, +, …) within the corresponding element nodes, aiming at preserving parent-child relations [85]. Both tree representations are constrained to the DTD language and do not consider XSD-based constraints which are more complex and expressive (e.g., *MinOccurs* and *MaxOccurs*). In addition, both approaches in [12, 85] (similarly to most existing methods) focus on XML structure, thus neglecting element values and data-types.



a. Tree representation based on the approach in [12].

b. Tree representation following the approach in [85].

Fig. 3. Tree representations describing the DTD grammar in Fig. 2.a.

[1] With respect to

3. State of the Art in XML Document/Grammar Comparison and Related Problems

XML document/grammar comparison underlines a relatively novel research area w.r.t. XML document comparison [17, 38, 86] and XML grammar comparison [30, 81, 87], which have attracted much research activity in the past decade.

In this section, we provide a summarized review on the most prominent issues and techniques related to XML document and grammar comparison. Section 3.1 discusses the seemingly related issue of approximate pattern matching with *Variable Length Don't Cares (VLDC)*. Section 3.2 covers XML document/grammar validation. Section 3.3 discusses XML document transformation/ correction. Consequently, we go over existing XML document/grammar similarity evaluation algorithms in Section 3.4, and sum up with discussions in Section 3.5.

3.1. Approximate Pattern Matching with VLDC

An intuitive XML document/grammar comparison approach could be that of approximate matching with the presence of *Variable Length Don't Cares (VLDC)*. In strings, a *VLDC* symbol (e.g., U) in a given pattern substitutes zero or more symbols in the data string [3, 46]. Approximate *VLDC* string matching means that, after the best possible substitutions, the pattern still does not match the data string and thus a matching distance is computed. For example, "*comp*U*ng*" matches "*commuting*" with distance 1 (representing the cost of removing the "*p*" from "*comp*U*ng*" and having the "U" substituting for "*mmuti*"). The string *VLDC* problem has been generalized for trees [97], introducing *VLDC* substitutions for whole paths or sub-trees. It has also been investigated in the context of Web data extraction, using structural patterns with special *VLDC* symbols (substituting single nodes, sets of siblings, and/or sub-trees) to identify data-rich information in Web document [68]. Nonetheless, despite being comparable, *VLDC* symbols are fairly different from repeatability and alternativeness operators in XML grammars (cf. Section 2.2.3). *VLDC* symbols can replace any string (w.r.t. string matching) or sub-tree (w.r.t. tree matching) whereas the XML grammar operators specify constraints on the occurrence of a particular and well known node (and consequently the sub-tree rooted at that node). For example, the DTD operator "?" associated with a given element *dummy* (*dummy?*) designates that the specific node entitled *dummy* (and not any other node, as with *VLDC* symbols) can appear *0* or *1* time.

3.2. XML Document/Grammar Validation

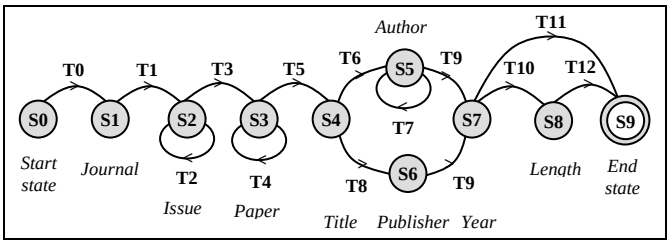
Another issue related to XML document/grammar comparison is XML document validation, which has recently gained attention as one of the aspects of XML data management. The basic idea adopted in this family of methods [9, 10, 14, 24, 45, 79] is to abstract DTDs/XSDs as extended Context-Free Grammars (CFG).

While the structural properties of XML grammar languages are essentially captured by *regular tree languages* (cf. Section 2.2.1), a CFG is a regular grammar expressing the fact that non-terminal symbols are rewritten without regard to the context in which they occur. A CFG for XML requires that the definition of a given element is independent of its position (i.e., context) in the document, the element being identified by its label.

For instance, the following sample DTD element declarations cannot coexist in the same grammar: *<!ELEMENT Author (Name)>* and *<!ELEMENT Author(FName, LName)>*. Such declarations would indicate that the content of element *Author* changes following its location in the grammar, which is not allowed in the DTD language, based on the CFG model[1]. Consequently, verifying if an XML document *S* conforms to XML grammar *G* requires to check whether the document tree is comprised in the language defined by the grammar, i.e., if $S \in L(G)$.

T0: *Start* → Journal	**T7:** Author → Author +
T1: Journal → Issue	**T8:** β → Publisher
T2: Issue → Issue +	**T9:** δ → Year, θ
T3: Issue → Paper	**T10**: θ → Length,
T4: Paper → Paper +	**T11:** θ → *End*
T5: Paper → Title , (α | β), δ	**T12:** Length → *End*
T6: α → Author	

a. State transitions describing the structural validation of document instances corresponding to the XML grammar in Fig. 3.a.



b. Simulating the XML grammar automaton.

Fig. 4. XML grammar automaton to perform document validation.

The standard procedure for testing membership in a formal language simulates the automaton that accepts the language on the input strings [43]. Thus, with DTDs/XSDs described as CFGs, one has first to produce the corresponding automaton for performing validation, and consequently run the document trees against the automaton. Standard procedures for producing automata and testing the membership of data instances w.r.t. a given automaton have been thoroughly studied in language theory [43, 62]. XML validation approaches [9, 10, 14, 24, 40, 45, 79] extend the latter techniques to deal with the special case of XML

[1] Note that the more expressive XSD language allows context-dependent declarations, due to the decoupling between element labels and data-types (e.g., *<element name= 'Author' type= 'SimpleName'/>* and *<element name= 'Author' type= 'CompositeName'/>* are allowed in the same XSD grammar) [51]. Yet, in the context of XML validation, XSDs are usually transformed into special DTDs, following the CFG model, so as to simplify the validation task and gain in efficiency [5, 34].

CFGs and XML document trees. In general, the validation is performed in $O(|S|\times\log(|G|))$ time, where $|S|$ and $|G|$ designate respectively the cardinalities (i.e., size in number of nodes) of the XML document and XML grammar at hand [9, 10]. However, the validation process becomes exponential in the size of the XML grammar, when the latter encompasses recursive declarations [79].

A sample automaton simulating the structure of the XML grammar in Fig. 2.a is shown in Fig. 4 (most document validation methods have been developed for DTDs, and thus usually disregard data-types which amount to string values, i.e., PCDATA). One can realize that the XML document in Fig. 1 follows the structural and the cardinality constraints set by the automaton, and is hence structurally valid w.r.t. the corresponding grammar.

Notice that methods for XML document validation generate a *Boolean* result indicating whether the XML document is valid or not w.r.t. the XML grammar. Nonetheless, they do not produce a similarity value (e.g., $Sim(Doc, Gram) \in [0, 1]$) quantifying the actual resemblance between the document and grammar at hand.

3.3. XML Document Transformation and Correction

A few methods for document-to-grammar transformation and/or correction have been recently proposed in the literature [13, 21, 64, 84]. An approach for identifying the edit script transforming a given XML document to another document conforming a given DTD grammar is proposed in [84]. An XML document is represented as an ordered labeled tree S whereas a DTD is abstracted as a triplet $D=(\Sigma, d, l_0)$ where Σ is the set of element labels, d a mapping from Σ to a set of regular expressions over Σ, and l_0 the starting label (corresponding to the DTD root element). The author makes use of three node operations, insertion, deletion, and update. The method consists of three main steps. The first step builds a special graph structure G, based on the XML document tree S, identifying all possible operations applicable to S. Graph nodes represent XML elements, whereas edges represent the different kinds of operations that could be applied on the elements. The second step refines the graph by removing all looping edges, i.e., edges representing unnecessary operations (e.g., deleting and then inserting the same node). The third step consists of the algorithm for finding the minimum cost edit script transforming the XML tree. The algorithm goes through graph G, and verifies which paths have sequences of labels that satisfy the DTD regular expressions. This is achieved via the use of NFAs (Non-deterministic Finite Automatons), each of which is of the form $(P, \Sigma, p_s, F, \delta)$ where: P is a set of states, Σ the set of labels, $p_s \in P$ is the start state, $F \subset P$ is a set of final states, and $\delta: e \times R \rightarrow p$ is a transition function where $e \in \Sigma$, R is a regular expression over P, and $p \in P$ (a simple example, in the context of document validation, is shown in Fig. 4). Consequently, those sequences of minimum costs are identified and form the minimum cost transformation script. The authors show that their approach is of polynomial complexity ($O(|\Sigma|\times |S^2| \times (|S|^4 + R^2))$ where R is the size of the largest regular expression in D) when the cost of an operation on a node only depends on the label of the node itself, and that it becomes *NP-Complete* otherwise.

In [60, 64], the authors address event-based document transformation, as a way to prevent loading the entire XML document into memory before starting tree manipulations. The authors argue that a DOM-like parsing/transformation strategy [93], which consists in fitting the whole XML document structure in main memory prior to launching the transformation process, could be a serious problem when the size of the input document is very large or when the size of the memory is relatively small. Hence, they address event-based document transformation, as an alternative way for identifying document transformations following the SAX (event-based) parsing model [57], by translating a finite automaton into an interactive transformer program scattered into small action codes responding to parsing events. The approach is developed for simplified and less-expressive grammar structures, disregarding various XML grammar constraints, and is thus of average linear time. Yet, the authors confirm that their event-based transformer has less expressive power than tree-based transformations.

Notice that the approaches in [60, 64, 84] are developed in the context of XML data transformation, and thus only target the transformation operations between the XML document and the (simplified) DTD grammar. They do not however address the issue of XML document/grammar similarity.

A problem comparable to document-to-grammar transformation is that of document-to-grammar correction. In the context of XML document validation in [13, 21], the authors tackle the complementary problem of XML document-to-grammar correction. The considered scenario is that of dynamic XML documents which are modified and updated frequently (e.g., documents modified by different users in an XML data warehouse [56], or those constantly updated on the Web describing commercial information for instance [31]), underlining the need to continuously test their conformance w.r.t. the corresponding grammars. While the classic strategy utilized in traditional database change management systems consists in disregarding all updates and modifications yielding data (e.g., XML documents) that are not valid w.r.t. the database schemas (e.g., XML grammars) [41], the authors in [13, 21] prioritize updates and thus propose an approach to correct the modified XML documents so that they become valid w.r.t. the corresponding grammars. In short, the authors propose to correct the sub-trees in the modified XML document where validation fails. The main idea consists in identifying the set of possible sub-tree corrections that yield structures conforming to the grammar. An XML grammar is simulated by a tree automaton (i.e., a NFA similar to the one presented in the previous paragraph), which allows the identification of the set of possible valid sub-trees w.r.t. the modified sub-tree at hand (i.e., the tree language defined by the grammar). Consequently, the authors exploit a classic tree edit distance algorithm [80] to compute the distance

between the modified sub-tree and each of the possible candidate sub-tree corrections, providing the user a choice of different possible corrections, such as when their distances from the original sub-tree are within a given threshold. The overall complexity of the approach, including NFA and edit distance computations, is shown to be exponential in the size of document node fan-out (i.e., maximum number of children nodes – maximum node degree – w.r.t. all nodes in the XML document tree). While the method produces distance (similarity) values between each of the original and the corrected sub-trees, the authors do not discuss the issue of computing an overall similarity score to evaluate the resemblance between the XML document and grammar at hand.

3.4. XML Document/Grammar Similarity

Few approaches have been proposed to measure the similarity between XML documents and grammars, i.e., produce a similarity score ($Sim(Doc, Gram) \in [0, 1]$) to quantify the amount of resemblance between the document and grammar at hand. To the best if our knowledge, the only methods are provided in [12, 37, 85, 94].

3.4.1. Semi-Structured Data and Data-guide Comparison

In [37], the authors address the problem of determining whether semi-structured data satisfy a given data-guide, in the context of approximate querying. A basic assumption in this work is that users specify a distortion transducer, through which a data-guide is distorted via elementary transformations. Here, the authors view distortions as a set of transformation operations (i.e., label insertions, deletions and substitutions), applied to the original data-guide, to obtain a transformed data-guide. A data-guide is viewed as a concise summary of the semi-structured database (defined as a labeled edged graph), i.e., a schema that the data-base should conform to. Hence, the user would define a distortion transducer, through which the data-guide can be distorted via elementary transformations and then employed to test if the database conforms to the resulting data-guide. The authors utilize the same technique to compare semi-structured data to a given query, evaluating the distorted query against the semi-structured database. A transducer comes down to an NFA defined as (P, Σ, Σ_O, f, p_s, F), with a finite set of states P, an input alphabet Σ, an output alphabet Σ_O, a starting state p_s, a set of final states F, and a transition-output function f: $P \times \Sigma \rightarrow P \times \Sigma_O$. Hence, regular transducers are extended by assigning non-negative weights to the transitions, in order to compute an overall data-guide (query) transformation cost. Consequently, data approximately conforming to a given data-guide (the approximate answers to a given query) come down to the data actually conforming the corresponding distorted data-guides (answers of the distorted queries) generated via the associated transducer and are ranked following data-guide (query) transformation costs. The approach is of $O(|A| \times N \times |P|^3)$ where $|A|$ is the number of states in the automaton describing the data-guide (query), N is the number of objects in the database (i.e., cardinality of the semi-structured document), and $|P|$ the number of states in the distortion transducer.

Note that the approach in [37] is not developed for XML documents and DTDs/XSDs, but for generic semi-structured data and data-guides (i.e., there are no constraints on the repeatability and alternativeness of elements).

3.4.2. XML Document and Grammar Matching

To the best of our knowledge, the first approach to specifically address the issue of XML document/grammar similarity, particularly DTDs, is proposed by Bertino *et al.* in [12]. Here, XML documents and DTDs are modeled as labeled trees. DTD trees include additional nodes for representing cardinality and alternativeness operators (i.e., '?', '*', '+, '*And*', '*Or*', cf. example in Fig. 3.a). The proposed algorithm takes into account the level (i.e., depth) in which the elements occur in the hierarchical structure of the XML and DTD tree representations. Elements at higher levels are considered more relevant, in the comparison process, than those at lower levels. The algorithm also considers element complexity (i.e., the cardinality of the sub-tree rooted at the element) when computing similarity values. While it does not explicitly identify a mapping between the XML and DTD tree nodes being compared, the proposed algorithm relies on the identification and evaluation of i) elements appearing both in the document and in the DTD, referred to as common elements, ii) elements appearing in the document but not in the DTD, referred to as plus elements, iii) and elements appearing in the DTD but not in the document, referred to as minus elements. Different weights can be assigned to each group of elements so as to tune the similarity measure following the user's needs.

The authors state that their approach is of exponential complexity in the general case, and argue that it can become polynomial ($O(\Gamma^2 \times (|S|+|D|))$ where $|S|$ is the number of nodes – elements/attributes – in the XML document tree S, $|D|$ the number of nodes – elements/attributes as well as ?, *, +, *And*, *Or* operators – in the DTD tree D, and Γ the maximum XML document node fan-out, i.e., maximum node degree) when the following assumption holds: *In the declaration of a DTD element, two sub-elements with the same tag are forbidden* (e.g., declaration *<!ELEMENT root(b, b, c)>* is forbidden since node *b* appears twice). In addition, the approach does not consider repeatable alternative expressions (e.g., *(A | B)+*), or recursive declarations. The authors also confirm the heuristic nature of their approach, stating that wrong matches could occur in the general version of their algorithm.

3.4.3. TED-based Document/Grammar Comparison

Other projects dealing with the XML document/grammar comparison issue are presented in [85, 94]. These are based on the concept of Tree Edit Distance (TED) as a more effective solution to comparing XML tree structures. Tree Edit Distance is a dynamic programming technique for

finding the cheapest sequence of edit operations transforming one tree structure into another [96]. TED-based algorithms have been widely exploited for comparing semi-structured data, namely XML document trees [27, 63, 86], and have provided optimal results in comparison with less accurate structural comparison methods [17].

Nonetheless, while XML documents can be naturally represented as tree structures, recall that XML grammar representations are usually more intricate due to the presence of grammar constraints (cf. Section 2.2). In [94], the author simplifies DTD definitions, eliminating all kinds of cardinality and alternative constraint operators (cf. Section 2.2.3), and introduces a TED-based formulation for comparing an XML document tree to the simplified DTD grammar. Yet, the author considers recursive DTD declarations (which are disregarded in most existing approaches). The proposed approach is of $O(|S|\times|G|\times \log(|D|))$ time, where $|D|$ is the size of the grammar and $|S|$ is the size of the XML document tree.

In [85], the authors propose a new TED-based algorithm considering some of the basic XML grammar constraints. The authors introduce an XML grammar tree representation model considering the basic constraints on the existence, repeatability and alternativeness of elements/attributes, while being comparable to XML document trees. An XML grammar is represented here as a set of conjunctive trees, corresponding to its disjunctive normal form. A conjunctive grammar C is composed of conjunctive element declarations, i.e., declarations defining sequences of elements. Hence, the disjunctive normal form [76] of an XML grammar G is the set of conjunctive grammars, $\{C\}_G$ covering all alternative declarations in G, (declarations resulting from the use of the *Or* operator). For instance, the alternative declaration *(Author|Publisher)* in the DTD grammar of Fig. 2.a is split into *Author* and *Publisher*, each represented as a separate conjunctive declaration where nodes only constitute sequences of elements (cf. corresponding DTD tree representation in Fig. 3). After transforming grammars into their disjunctive normal form, the authors in [85] provide a novel tree edit distance algorithm to evaluate the distance (similarity) between the document and grammar trees and hand. The proposed approach is polynomial in document (S) and grammar (G) size, i.e., $O(|S|\times|G^2|)$. Nonetheless, the approach targets simplified DTD grammars (DTDs lacking optional/repeatable expressions – e.g., *(A | B)+*, and recursive declarations) and only considers the basic DTD constraints (e.g., *?*, +, and *) in the computation process.

3.5. Discussion

To conclude, few approaches have been proposed to compare an XML document with an XML grammar. While methods for XML document validation [9, 10, 14, 79], and XML document transformation/correction [13, 21, 64, 84] seem related to the problem of XML document/grammar similarity, they do not produce a similarity value, but either generate a *Boolean* result (indicating whether a document is valid w.r.t. a given grammar), or produce a transformation script (transforming a document into another document valid w.r.t. the grammar). The authors in [37] address the problem of comparing generic semi-structured data to a given data-guide. However, they do not consider constraints on the repeatability and alternativeness of elements (since the latter do not exist in data-guides). To our knowledge, the only methods to specifically target XML document/grammar comparison, are provided in [12, 85, 94]. The proposed algorithms consider DTD constraints with certain simplifications and/or restrictions. The methods in [12, 85] do not consider XSD *MinOccurs* and *MaxOccurs* operators, nor do they discuss the special cases of repeatable alternative expressions and recursive expressions. The approach in [94] considers recursive declarations, yet disregards all kinds of XML grammar repeatability and alternativeness constraints.

Table 1 summarizes the properties of XML document/ grammar comparison methods and related approaches. Note that most solutions in the literature do not provide empirical performance analyses. Hence, discussing and contrasting corresponding performance levels requires a dedicated experimental evaluation study which is out of the scope of this paper (to be addressed in a future work).

Table 1. Characteristics of the main methods related to XML document/grammar comparison.

Approach	Performs Document Validation	Generates Transformation Script	Computes similarity value ($\in[0, 1]$)	Considers grammar Constraints	Considers recursive declarations	Dedicated to XML Grammars	Complexity level
Segoufin *et al.* [79]	✓	×	×	✓	✓	✓ (DTD)	Exp(G)
Barbosa *et al.* [10]	✓	×	×	✓	×	✓ (DTD/XSD)	$O(S\times\log(G))$
Balmin *et al.* [9]	✓	×	×	✓	×	✓ (DTD/XSD)	$O(S\times\log(G))$
Bouchou *et al.* [13]	✓	*Partial*	*Partial*	✓	×	✓ (DTD/XSD)	Exp(*MaxDeg*(S))
Suzuki [84]	×	✓	×	✓	×	✓ (DTD)	*NP-Complete*
Bouchou *et al.* [14]	✓	×	×	✓	×	✓ (DTD/XSD)	$O(S\times\log(G))$
Grahne & Thomo [37]	✓	×	✓	×	×	× (semi-struct data)	$O(\|A_G\|\times N\times\|T\|^3)$
Bertino *et al.* [12]	✓	×	✓	✓ (restricted)	×	✓ (DTD)	Exp(G)
Tekli *el al.* [85]	✓	✓	✓	✓ (restricted)	×	✓ (DTD)	$O(\|S\|\times\|G\|^2)$
Xin G. [94]	✓	✓	✓	×	✓	✓ (DTD)	$O(\|S\|\times\|G\|\times\log(\|G\|))$

4. Potential Applications of XML Document and Grammar Comparison

The use of XML document/grammar similarity is central in a wide spectrum of applications, ranging over: i) XML document classification, ii) XML document transformation, iii) selective dissemination of XML documents, iv) XML grammar evolution, v) XML document clustering, vi) XML structural querying, as well as more specialized application scenarios including vii) alert filtering in intrusion detection systems, and viii) Web Services matching and communications.

4.1. XML Document Classification

XML document/grammar similarity evaluation enables the classification of XML documents gathered from the web against a set of grammars declared in an XML database (data/type comparison layer). A scenario provided by Bertino *et al.* [12] comprises a number of heterogeneous XML databases that exchange documents with each other, each database storing and indexing the local documents according to a set of predefined DTD grammars. Consequently, XML documents introduced in a given database are matched, via an XML structural similarity method, against the local DTDs. Note that *matching*, in such an application, can be undertaken using an XML document/grammar comparison method (like the ones investigated in this study, cf. Section 3.4) or via an XML document structure comparison method [17, 38, 86]. Following the latter strategy, the XML grammar will be exploited as a generator of XML document structures. The set of possible document structures valid for the grammar is considered. Then, for each document structure, algorithms for measuring the similarity between XML document structures [17, 38, 86] can be applied. The match resulting in the highest similarity value is considered as the best match, the corresponding similarity value being considered as the structural similarity degree between the document and the XML grammar.

In such an application, a similarity threshold is identified, designating the minimal degree of similarity required to bind an XML document to a grammar. The XML grammar for which the similarity degree is the highest is selected, given that the similarity value is above the specified threshold. Thus, the XML document at hand is accepted as valid for that grammar. When the similarity degree is below the threshold, for all grammars in the XML database, the XML document is considered *unclassified* and is stored in a repository of unclassified documents. As a result, none of the access protection, indexing and retrieval facilities specified at the XML grammar level can be applied to such documents (similarly to schemas and traditional DBMS) [12].

4.2. XML Document Transformation

When populating an XML database on the web, an issue complementary to XML document classification is document transformation. After having migrated a set of documents collected from various data sources into an XML database (defined by a set of grammars), the collected documents may be similar to, but may not satisfy any grammar in the database. Hence, storing and managing the documents in the database require: i) identifying which of the grammars is the most similar to the document at hand (i.e., classification phase [12], where the similarity measure itself is needed), and then ii) transforming the documents into valid ones w.r.t. their most similar grammars (i.e., document transformation, also known as document revalidation or correction [13, 21, 50]). Here, the advantage of using an XML document/grammar comparison method based on the concept of edit distance [96] becomes obvious. With such a method, a mapping between the nodes in the compared structures is provided in terms of the edit script, along with the similarity value itself. The mapping thus describes the set of transformation operations to be applied to the document, so that it becomes valid w.r.t. the grammar under which it is classified.

4.3. Selective Dissemination of XML Documents

SDI (Selective Dissemination of Information) systems for XML-based data become increasingly popular with the growing use of XML on the web [5, 19, 29, 82]. An SDI system basically manages user preferences to identify the users to whom incoming web documents should be broadcasted. Users can set their preferences when they first connect to the system or the preferences can be dynamically discovered by monitoring the documents that the users frequently access. SDI systems allow the filtering of XML document streams w.r.t. user preferences. A key capability of an SDI system is the adaptability of user profiles to varying user preferences [12], which is essential in a dynamic environment such as the web.

XML classification and evolution techniques can be employed to build an effective SDI system dedicated to XML-based data. While most existing techniques to XML SDI usually exploit XML querying paradigms (e.g., XQuery [18] and/or XPath patterns [11]) as filters for the documents of interest, a user profile can be described with a higher degree of expressiveness as an XML grammar (DTD or XSD) [12]. Consequently, XML classification can be utilized to filter documents based on their similarities w.r.t. the considered grammars. The grammar, describing the user profile, is initially specified by the user, or automatically inferred from documents previously deemed valuable by the user using document clustering [27, 63] and structure extraction techniques [35]. The selective dissemination of XML data can then be undertaken by matching each XML document in the incoming data stream against the grammars modeling the user profiles. Documents are distributed to the users whose document/grammar similarities are above a predefined threshold.

4.4. XML Grammar Evolution

XML grammar evolution involves the modification of a grammar describing a class of documents (a user profile),

in order to capture more accurately the structural characteristics of corresponding XML documents. In other words, it allows to reduce the divergence between the structures of the documents being classified under the grammar (accessed by the user), and the structure as specified by the corresponding grammar (user profile). However, the evolution phase is a costly process [12] since it requires the use of data mining association rules [47] and structure extraction techniques [35], so as to capture frequent patterns of element structures in the document instances to generate the updated grammars [26]. Hence, it ought to be triggered when the grammar (profile) is not anymore representative of its classified documents (accessed documents) [12], which is where XML document/grammar similarity comes to play. Here, the user/administrator can i) specify an XML document/grammar similarity threshold, that a minimum number of documents in the XML document class must respect, so as to prevent the evolution phase, ii) or specify the maximum number of non-conforming documents a class must encompass (i.e., documents with $Sim_{XDoc\_XGram}<1$ w.r.t. the grammar associated with the class at hand). When the number of non-conforming documents is higher than the threshold, the corresponding class is deemed *not representative* of its instances, which requires launching the evolution phase [12].

4.5. XML Document Clustering

Grouping similar XML documents together can improve data storage indexing [78], and thus positively affect the retrieval process. For instance, if two documents/elements are similar, it is likely that they both either satisfy or not a given query. Therefore, when grouped together, similar documents/elements would be much easier to retrieve than when scattered at different locations in the storage device [51]. Clustering can also be critical in information extraction. Current information extraction methods either implicitly or explicitly depend on the structural features of documents [17, 68]. Structural clustering allows to automatically identify the sets of XML documents and/or document patterns that are useful in information extraction algorithms, in order to produce meaningful results [68].

In [95], the authors present a method for XML document clustering based on document/grammar similarity evaluation. The approach consists in extracting grammars (DTDs) to represent predefined sets of XML documents (i.e., the original clusters) [35]. Consequently, an incoming XML document is compared to each of the cluster representatives (i.e., DTDs) and is allocated to the cluster with which it shares maximum document/grammar similarity [94]. While the approach seems interesting, its effectiveness and performance levels depend on two major factors: i) the availability of a predefined set of relevant clusters, which is not always obtainable, and ii) the complexity of the grammar (DTD) generation phase [35], which has been shown to be NP-Hard [33].

4.6. XML Structural Querying

Recent approaches to XML document retrieval exploit the structure of documents to improve both accuracy and efficiency. Such queries are referred to as structural queries [12]. In such a context, XML grammars could be exploited as structural queries representing structural constraints on the queried documents. Using a comparison method between XML documents and XML grammars, it is possible to verify whether a document is an answer to a query (XML grammar) following their degree of similarity. If the structural similarity between the query (XML grammar) and the XML document tree is above a given threshold, the document is added to the set of answers for the query. The query answer set is ranked following the computed similarity degrees.

Note that additional constraints can be added on the values (data contents) of XML elements/attributes in XML grammars. These become the so-called *content-and-structure* queries. Systems that enable *content-and-structure* XML querying have been recently receiving a lot of attention, especially through the INEX[1] (INitiative for the Evaluation of XML Retrieval) campaigns. The proposed approaches tend to extend classical information retrieval techniques [70, 72] by taking into account the structural aspect of XML data (see [86] for a concise review of information retrieval based XML similarity methods).

4.7. Alert Filtering in Intrusion Detection Systems

With the significant increase in security threats and the number of attacks on information systems over the past decade, information security technologies such as authentication, cryptography, and more recently intrusion detection have been gaining more attention. An Intrusion Detection Systems (or IDS) monitors an information system for evidence of attacks. Once attacks have been detected, the IDS raises alerts, which are then presented to experts and/or a knowledge system that evaluate them and initiate an adequate response. Nonetheless, evaluating intrusion detection alerts remains a delicate and non-trivial process. For instance, it has been observed in [8, 44] that IDSs can trigger thousands of alerts per day, up to 99% of which are deemed false alerts, which makes it difficult to identify the hidden true positives.

Another recent breakthrough in IDSs is the emergence of IDMEF (Instruction Detection Message Exchange Format) [28] as an XML-based format for sharing information of interest to intrusion detection and response systems, namely encoding and exchanging alert messages. This underlines a great opportunity to exploit XML-based techniques in order to improve the effectiveness of IDS systems. An original study to cluster IDMEF alters via an XML-based document similarity algorithm, in order to identify groups of relevant messages and process them together, has been proposed in [54]. In addition to IDMEF

[1] http://inex.is.informatik.uni-duisburg.de/.

message (document) comparison, XML document/grammar similarity evaluation techniques can be particularly useful in filtering false alerts. A potential scenario consists in matching and classifying XML-based (IDMEF) alert messages against a set of grammars (predefined by security experts or a knowledge system) describing the most common categories of intrusions that the IDS system usually faces. Alert messages with similarity scores above a given threshold are accepted as true alerts w.r.t. the intrusion category corresponding to the grammar at hand. Otherwise, messages with lower similarities are deemed false alters, and are disregarded by the system. This would most likely reduce the amount of false alters processed by the system, and consequently facilitate the identification and evaluation of meaningful intrusion alerts.

4.8. Web Services Matching and SOAP Processing

Another interesting application area which requires XML document/grammar similarity evaluation is the matching, search, and composition of Web Services (WS). WS are software systems designed to support interoperable machine-to-machine interactions over a network (namely the Internet) [23]. An individual web service generally comes down to a self-contained, modular application that can be described, published and invoked over the Internet, and executed on the remote system where it is hosted [71]. WS rely on two standard XML schemata: WSDL (Web Service Description Language) [23] allowing the definition of XML grammar structures to support the machine-readable description of a service's interface and the operations it supports, and SOAP (Simple Object Access Protocol) [92] for XML-based communications and message exchange among WS end-points.

Hence, when searching for WS achieving specific functions, XML-based service requests can be issued, to which are consequently matched and ranked service WSDL descriptions, thus identifying those services answering the required computation needs. In this context, matching and ranking WS descriptions against WSDLs requires effective XML document/grammar comparison techniques. Likewise for WS composition: grouping together a series of services requires the processing and comparison of corresponding WSDL descriptions, so as to execute a specific composite task. In addition, XML-based similarity and differential encoding can be exploited to enhance SOAP performance: comparing new SOAP messages to predefined WSDL grammars, processing only those parts of the messages which differ from the corresponding WSDL. Identifying the common parts of SOAP messages, and repeating the processing for only those parts which are different from the WSDL schema would avoid a large amount of unnecessary overhead, and thus allow reducing processing cost in SOAP parsing [89], serialization [2], de-serialization [1], and communications [91]. For more details, a comprehensive review on (XML) similarity-based SOAP performance enhancement techniques can be found in [88].

5. Future Research Directions

Despite the recent efforts conducted around the XML document/grammar similarity problem, yet various issues and challenges remain unaddressed. We present some of these issues in the remainder of this section. First, we discuss the limitations of current approaches w.r.t. the structural characteristics of XML data in Section 5.1. Section 5.2 covers XML content-based similarity. In Section 5.3, we address the combination of XML structure and semantic similarity in improving the comparison results. Then, Section 5.4 concludes with a brief discussion concerning the trade-off between the effectiveness and efficiency of XML document/grammar comparison.

5.1. XML Structure-based Similarity

On one hand, most existing approaches to XML document/grammar similarity evaluation induce various simplifications in the XML grammars in order to perform the comparison task. Three major *hard-to-match* declarations are usually disregarded, including: i) repeatable sequence expressions (i.e., a sequence of elements, connected via the *And* operator, and associated a cardinality constraint, such as DTD declarations (*A, B, C*)+ and (*A, B, C*)*) [85], ii) repeatable alternative expressions (i.e., an alternative of elements, connected via the *Or* operator, and associated a cardinality constraint, such as DTD declarations (*A* | *B* | *C*)+ and (*A* | *B* | *C*)*) [12, 85], and iii) recursive expressions (which could induce infinite loops of elements, and thus have been proven complicated to handle in the comparison task [79, 94]).

On the other hand, most existing approaches are constrained to the DTD grammar syntax, and do not address the expressive XSD language. In particular, XSD constraints *MinOccurs* and *MaxOccurs*, specifying respectively the minimum and maximum number of times an element/attribute can appear in the corresponding XML document, and which are remarkably more expressive than their ?, *, and + DTD counterparts, are currently disregarded in most approaches, and would have to be considered in order to obtain more effective and accurate document/grammar comparison methods.

5.2. XML Content-based Similarity

Most existing methods to XML document/grammar comparison focus solely on XML structure (i.e., hierarchical relations and ordering of elements/attributes, identified by their labels), disregarding element values in documents and element data-types in grammar declarations. Nonetheless, as discussed in Section 4.6, considering XML contents could prove to be extremely useful, namely in XML *content-and-structure* querying applications which have been recently gaining momentum in both database and information retrieval research [6, 7].

In this context, a recent study in [34] focuses on comparing XSD grammar elements based on their simple data-types (e.g., *String*, *Integer*, *Date*…) and/or complex

data-types (i.e., types declared by a sequence and/or alternative of elements). The author specifically focuses on derived complex data-types and introduces the notion of *type hierarchy* (i.e., a taxonomy linking types following their XSD e*xtension*/*restriction* operator relations). While developed for XML grammar matching (i.e., comparing the elements of two XML grammars), such an approach could be extended and/or adapted to XML document/grammar comparison in order to compare XSD data-types with corresponding XML document element/attribute values.

5.3. XML Semantic-based Similarity

Combining structural and semantic XML similarity is one of the hot topics recently being investigated in the XML document comparison [75, 86, 90] and XML grammar comparison [30, 36, 81] literatures. Nonetheless, most existing XML document/grammar similarity approaches focus exclusively on the structure of documents and grammars, ignoring the semantics involved (semantic meaning of XML element/attribute labels – and values, when the latter are considered – given a reference semantic information source such as WordNet[1] [58]). Evaluating the semantic relatedness between documents and grammars (mainly those published on the Web) can prove to be of key importance to improving search results: finding documents semantically related to a given grammar, and given a set of documents, effectively ranking them according to their semantic similarity.

```
<!ELEMENT Journal (Issue*)>
<!ELEMENT Issue (Paper+)>
<!ELEMENT Paper ((Author+ | Publisher), Year, Length?)>
<!ATTLIST Paper Title CDATA>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Length (#PCDATA)>
```

a. DTD grammar reported from Fig. 2.a.

```
<Transactions>
  <Volume Nb = '…'>
    <Article>
      <Author>…</Author>
      <Date>…</Date>
    </Article>
  </Volume>
</Transactions>
```

b. Sample document

Fig. 5. Semantically related XML document and grammar.

For instance, using existing structure comparison methods, e.g., [12, 85, 94], the document and grammar in Fig. 5 are deemed completely different, since all elements bear syntactically different labels. However, one can obviously recognize that they describe highly related information, reflected by the semantic relatedness of their labels: *Journal*-*Transactions*, *Paper*-*Article*, *Issue*-*Volume*, and *Year*-*Date*. In other words, semantic similarity evaluation supports the identification of entities that are conceptually close, but not exactly identical, which is crucial in settings such as heterogeneous XML repositories, particularly on the Web where users have different backgrounds and no precise definitions about the matter of discourse [55].

[1] WordNet is a domain independent online lexical reference system, developed at Princeton University NJ USA, in an attempt to model the lexical knowledge of a native English speaker. It organizes nouns, verbs, adjectives and adverbs into synonym sets, each representing an underlying lexical concept (http://www.cogsi.princeton.edu/cgibin/webwn).

In this context, a vast arsenal of methods to determine the semantic similarity between concepts in a knowledge base (semantic network) has been developed in the fields of Information Retrieval and Natural Language Processing [16, 74, 98]. These can be categorized as edge-based approaches (evaluating the distance separating two concepts in the reference semantic network, e.g., [48, 66]) and node-based approaches (estimating concept information contents using corpus based statistics, e.g., [53, 69]). In short, while semantic similarity measures have been thoroughly investigated in the literature [16, 74, 98], nonetheless, efficiently integrating such techniques within XML document/grammar comparison remains an open issue yet to be investigated. To sum up, considering the semantic factor in XML similarity computations would clearly amend comparison results, and is central in most application scenarios discussed in Section 4, namely in XML querying, classification, clustering, and the selective dissemination/filtering of semantically related XML data.

5.4. XML Comparison Efficiency

While the effectiveness (accuracy) of XML document/ grammar comparison is a major concern, nonetheless, the efficiency (performance) of the proposed solutions remains equally important, especially for Web-based applications. In this context, most existing approaches simplify XML grammar representations [85, 94] or utilize various heuristics [12] in order to gain in processing speed. We also stumbled on an original approach in [60, 64], developed for document/grammar transformation, where the authors exploit event-based XML processing as a way to prevent loading the entire XML document into memory before starting tree manipulations. While the method is extremely efficient (of average linear complexity) [60, 64], it simplifies XML grammars and provides less expressive power in describing transformations, than its tree-based counterpart. On the other hand, and in contrast to simplifying grammars, the authors in [79] consider most XML grammar (DTD) constraints in conducting document/grammar validation, including recursive declarations. Yet, the proposed solution is exponential in the size of the XML grammar at hand.

Hence, a comprehensive empirical analysis addressing the trade-off between: i) the effectiveness and ii) the efficiency levels of XML document/grammar comparison methods, (considering the amount of simplification in the grammars being compared, as well as the different kinds of techniques utilized to compute similarity) is required in order to identify and better understand the ups and downs of each approach, so as to develop more sophisticated solutions. In addition, recent techniques related to performance enhancement in XML document similarity (such as Entropy [42] and Structural Pattern Indexes [73]) and XML grammar similarity (such as Prufer sequence encoding [4] and B-Tree indexing [32]), could be investigated (and possibly adapted or combined) to improve the performance levels of XML document/grammar comparison solutions.

6. Conclusion

In this paper, we provided an overview on existing research related to XML document/grammar comparison. Existing methods were roughly organized into three major groups, targeting: i) document/grammar validation, ii) document/ grammar transformation and correction, and iii) document/ grammar similarity evaluation. On one hand, methods for XML document validation produce a *Boolean* result indicating whether a document is valid w.r.t. a given grammar. On the other hand, methods for XML document transformation/correction produce a modification script, transforming a document into another document valid w.r.t. a given grammar. Nonetheless, few approaches have been designed to produce a similarity value, quantifying the amount of resemblance between an XML document and a grammar.

In brief, most existing approaches to XML document/grammar similarity evaluation induce various simplifications in the grammars (disregarding various XML grammar cardinality and alternativeness constraints) so as to perform the comparison task. In addition, most existing approaches are constrained to the DTD grammar syntax, and do not address the expressive XSD language. Furthermore, most methods focus on XML structure (i.e., hierarchical relations and ordering of elements/attributes, identified by their labels), disregarding element values in documents, and element data-types in grammar declarations. The semantic meanings of XML element/attribute labels are also disregarded in most approaches. To sum up, the XML document/grammar similarity domain is still in its infancy, with a large spectrum of problems to be addressed in the near future.

We also discussed some of the possible applications of XML document/grammar comparison in various fields, ranging over XML document classification, clustering, selective dissemination, document transformation, grammar evolution, XML querying, as well as more specialized application scenarios including alert filtering in intrusion detection systems, and Web Services matching and communications.

We hope that our presentation of XML document/ grammar comparison and related problems in this paper will contribute to strengthen further research on the subject.

ACKNOWLEDGEMENT

This work was supported in part by the Research Support Foundation of the State of Sao Paulo, FAPESP Post-doctoral Fellowship n# 2010/00330-2.

REFERENCES

[1] Abu-Ghazaleh N. and Lewis M.J., *Differential Deserialization for Optimized SOAP Performance*. Proceedings of the ACM/IEEE Conference on Supercomputing, 2005. pp. 21-31, Seattle.

[2] Abu-Ghazaleh N.; Lewis M.J. and Govindaraju M., *Differential Serialization for Optimized SOAP Performance.* Proceedings of the 13th International Symposium on High Performance Distributed Computing (HPDC'04), 2004. pp. 55-64.

[3] Akatsu T., *Approximate String Matching with Don't Care Characters.* Information Processing Letters, 1995. (55):235-239.

[4] Algergawy A.; Schallehn E. and G. Saake, *Improving XML schema matching using Prufer sequences.* Data and Knowledge Engineering, 2009. 68(8):724–747.

[5] Altinel M. and Franklin M. J., *Efficient Filtering of XML Documents for Selective Dissemination of Information.* Procedings of the 28th International Conference on Very Large Data Bases (VLDB'00), 2000. pp. 53-64.

[6] Amer-Yahia S. and Shanmugasundaram J., *XML Full-Text Search: Challenges and Opportunities.* Proceedings of the International Conference on Very Large Data Bases, 2005. Tutorial Slides, http://www.vldb2005.org/program/slides/fri/s1368-amer-yahia.ppt.

[7] Amer-Yahia S.; Case P.; Rolleke T.; Shanmugasundaram J. and Weikum G., *Report on the DB/IR Panel at SIGMOD 2005.* Sigmod Record, 2005. 34(4):71-74.

[8] Axelsson S., *The Base-Rate Fallacy and the Difficulty of Intrusion Detection.* ACM Transactions on Information and System Security, 2000. 3(3):186–205.

[9] Balmin A.; Papakonstantinou Y.; and Vianu V., *Incremental validation of XML documents.* ACM Transactions on Database Systems, 2004. 29(4):710-751.

[10] Barbosa D.; Mendelzon A. O.; Libkin L.; Mignet L.; and Arenas M., *Efficient Incremental Validation of XML Documents.* Proceedings of the international Conference on Data Engineering (ICDE), 2004. IEEE Computer Society, pp. 671-682.

[11] Berglund et al., *XML Path Language (XPath) 2.0.* W3C Recommendation, January 2007. http://www.w3.org/TR/xpath20/.

[12] Bertino E.; Guerrini G.; and Mesiti, M., *A Matching Algorithm for Measuring the Structural Similarity between an XML Documents and a DTD and its Applications.* Elsevier Information Systems, 2004. (29):23-46.

[13] Bouchou B.; Cheriat A.; Halfeld Ferrari M. and Savary A., *XML Document Correction : Incremental Approach Activated by Schema Validation*. Proceedings of the International Database Engineering and Applications Symposium (IDEAS), 2006. pp. 228-238

[14] Bouchou B.; Cheriat A.; Halfeld Ferrari M.; Laurent D.; Lima M. A. and Musicante M., *Efficient Constraint Validation for XML Database.* Informatica (http ://ai.ijs.si/informatica/), 2007. 31(3): 285-309.

[15] Bray T.; Paoli J.; Sperberg-McQueen C.; Mailer Y.; and Yergeau F. *Extensible Markup Language (XML) 1.0 - 5th Edition. W3C Recommendation*, 2008. http://www.w3.org/TR/REC-xml/

[16] Budanitsky A. and Hirst G., *Evaluating WordNet-based Measures of Lexical Semantic Relatedness.* Computational Linguistics, 2006. 32(1): 13-47.

[17] Buttler D., *A Short Survey of Document Structure Similarity Algorithms.* Proceedings of the International Conference on Internet Computing (ICOMP), 2004. pp. 3-9.

[18] Chamberlin D.; Florescu D.; Robie J.; Simeon J. and Stefanescu M. *XQuery : A Query Language for XML*. 2001, [cited May 2010]. http://www.w3.org/TR/2001/WD-xquery-20010215.

[19] Chan C.Y.; Felber P.; Garofalakis M. and Rastogi R., *Efficient Filtering of XML Documents with XPath Expressions.* The VLDB Journal, 2002. 11(4):354-379. .

[20] Chawathe S.; Rajaraman A.; Garcia-Molina H.; and Widom J., *Change Detection in Hierarchically Structured Information.* Proceedings of the ACM International Conference on Management of Data (SIGMOD), 1996. pp. 26-37. Montreal.

[21] Cheriat A.; Savary A.; Bouchou B. and Halfeld Ferrari M., *Incremental String Correction : Towards Correction of XML Documents.* Proceedings of the Prague Stringology Conference(PSC), 2005. pp. 201-215.

[22] Chidlovskii B., *Using Regular Tree Automata as XML Schemas.* Proceedings of the IEEE Advances in Digital Libraries (ADL'00), 2000. pp. 89-98.

[23] Chinnici R.; Moreau J.J.; Ryman A. and Weerawarana S. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation 26 June 2007, http://www.w3.org/TR/wsdl20/.* [cited 25 August 2009].

[24] Chitic C. and Rosu D., *On Validation of XML Streams using Finite State Machines.* Proceedings of the 7th International Workshop on the Web and Databases (WebDB '04) 2004. pp. 85–90, New York, NY, USA, ACM Press.

[25] Cobéna G.; Abiteboul S.; and Marian A., *Detecting Changes in XML Documents.* Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2002. pp. 41-52.

[26] Da Luz R.; Halfeld Ferrari Alves M.; Musicante M. A., *Regular expression transformations to extend regular languages (with application to a Datalog XML schema validator).* Journal of Algorithms, 2007. 62(3-4):148-167.

[27] Dalamagas T.; Cheng T.; Winkel K.; and Sellis T., *A Methodology for Clustering XML Documents by Structure.* Information Systems, 2006. 31(3):187-228.

[28] Debar H.; Curry D. and Feinstein B., *The Intrusion Detection Message Exchange Format (IDMEF).* http://www.ietf.org/rfc/rfc4765.txt, 2005.

[29] Diao Y.; Fischer P.; Franklin M.J. and To R., *YFilter: Efficient and Scalable Filtering of XML Documents.* Proceedings of the International Conference on Data Engineering (ICDE'02), 2002.

[30] Do H. and Rahm E., *Matching Large Schemas: Approaches and Evaluation.* Information Systems, 2007. 32(6): 857-885.

[31] Doan A.; Domingos P.; and Halevy A., *Learning to Match the Schemas of Data Sources: A Multistrategy Approach.* Machine Learning, 2003. 50(3):279-301.

[32] DuChateau F.; Bellahsene Z.; Hunt E.; Roantree M., a.R.M., *An Indexing Structure for Automatic Schema Matching.* The 23rd International Conference on Data Engineering (ICDE) - Workshops, 2007. pp. 485-491.

[33] Fernau H., *Extracting Minimum Length Document Type Definitions Is NP-Hard.* Grammatical Inference: Algorithms and Applications (ICGI'04) 2004. pp. 277-278.

[34] Formica A., *Similarity of XML-Schema Elements: A Structural and Information content Approach.* The Computer Journal, 2008. 51(2):240-254.

[35] Garofalakis M.; Gionis A.; Rastogi R.; Seshadri S.; and Shim K., *Xtract: A system for extracting document type descriptors from XML documents.* Proceedings of the ACM International Conference on Management of Data (SIGMOD), 2000. pp. 165-176. Dallas, Texas, USA.

[36] Giunchiglia F.; Yatskevich M. and Shvaiko P., *Semantic matching: Algorithms and implementation* Journal on Data Semantics (JoDS), 2007, 9:1-38.

[37] Grahne G. and Thomo A., *Approximate Reasoning in Semi-structured Databases.* Proceedings of the International Workshop on Knowledge Representation meets Databases (KRDB), 2001. Vol. 45, Rome.

[38] Guerrini G.; Mesiti M. and Sanz I., *An overview of similarity measures for clustering XML documents.* In A. Vakali and G. Pallis, editors, Web Data Management Practices: Emerging Techniques and Technologies. IDEA Group, 2006.

[39] Guha S.; Jagadish H.V.; Koudas N.; Srivastava D.; and Yu T., *Approximate XML Joins.* . Proceedings of ACM International Conference on Managemenet of Data (SIGMOD), 2002, 287-298.

[40] H.;, M.M.a.H., *Validation Algorithm for Attribute-Element Constraints of RELAX NG.* Extreme Markup Languages, 2003. Montreal, Canada.

[41] Halfeld Ferrari Alves M., *Aspects Dynamiques de XML et Spécification des Interfaces de Services Web avec PEWS.* Rapport de HDR, Université François Rabelais de Tours, 2007.

[42] Helmer S., *Measuring the Structural Similarity of Semistructured Documents Using Entropy* Proceedings of the International Conference on Very Large Databases (VLDB), 2007, 1022-1032.

[43] Hopcroft J. E.; Motwani R. and Ullman J. D., *Introduction to Automata Theory, Languages, and Computation.* 2001. Addison Wesley, 2nd edition.

[44] Kayacik H.G. and Zincir-Heywood A.N., *A Case Study of Three Open Source Security Management Tools.* Proceedings of 8th IFIP/IEEE International Symposium on Integrated Network Management, 2003. pp. 101–104.

[45] Kim S.K.; Lee M. and Lee K.C., *Validation of XML Document Updates Based on XML Schema in XML Databases.* International Conference on Database and Expert Systems Applications (DEXA'03), , 2003. LNCS 2736, pp. 98–108.

[46] Landau G. M and Vishkin U., *Fast Parallel and Serial Approximate String Matching.* Journal of Algorithms, 1989. (10):157-169.

[47] Lee G.; Lee K. and Chen A., *Efficient Graph-based Algorithms for Discovering and Maintaining Association Rules in Large Databases.* Knowledge and Information Systems, 2001. 3(3):338-355.

[48] Lee J.; Kim M.; and Lee Y., *Information Retrieval Based on Conceptual Distance in IS-A Hierarchies.* Journal of Documentation, 1993. 49(2):188-207.

[49] Lee M.; Yang L.; Hsu W. and Yang X., *XClust: Clustering XML Schemas for Effective Integration.* Proc. of the Inter. Conf. on Information and Knowledge Management (CIKM), 2002, 292-299.

[50] Leonardi E.; Hoai T.T.; Bhowmick S.S. and Madria S., *DTD-Diff: A Change Detection Algorithm for DTDs.* Proceedings of the Database Systems for Advanced Applications conference (DASFAA), 2006. pp. 384-402.

[51] Lian W.; Cheung D.; Mamoulis N.; and Yiu S., *An Efficient and Scalable Algorithm for Clustering XML Documents by Structure.* IEEE Transactions on Knowledge and Data Engineering, 2004. 16(1):82-96.

[52] Liang W.; and Yokota H., *LAX: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration.* Proceedings of the British National Conference on Databases (BNCOD), 2005. pp. 82-97.

[53] Lin D., *An Information-Theoretic Definition of Similarity.* Proceedings of the International Conference on Machine Learning (ICML), 1998. pp. 296-304. Morgan Kaufmann Pub. Inc.

[54] Long J.; Shwartz D. and Stoecklin S., *Distinguishing False from True Alerts in Snort by Data Mining Patterns of Alerts.* Proceedings of SPIE'06, the International Society for Optical Engineering, 2006.

[55] Maguitman A.; Menczer F.; Roinestad H.; and Vespignani A., *Algorithmic Detection of Semantic Similarity.* Proc. of the Inter. Conf. on the World Wide Web (WWW), 2005. pp. 107-116.

[56] Marian A.; Abiteboul S. and Mignet L., *Change-Centric Management of Versions in an XML Warehouse.* Proceedings of the International Conference on Very Large Data Bases (VLDB), 2001. pp. 581-590.

[57] Megginson D. et al. *The Simple API for XML* http://www.megginson.com/SAX/ [cited February 2010].

[58] Miller G., *WordNet: An On-Line Lexical Database.* International Journal of Lexicography, 1990. 3(4).

[59] Murata M.; Lee D.; Mani M. and Kawaguchi K., *Taxonomy of XML Schema Languages Using Formal Language Theory.* ACM Transactions on Internet Technology (ACM TOIT), 2005. 5(4):660-704.

[60] Nakano K. and Nishimura S., *Deriving Event-Based Document Transformers from Tree-Based Specifications.* In Mark van den Brand and Didier Parigot, editors, Electronic Notes in Theoretical Computer Science, 2001. Volume 44. Elsevier Science Publishers.

[61] Neumann A., *Parsing and Querying XML Documents in SML.* PhD thesis, University of Trier, Trier, Germany, 2000.

[62] Neumann A. and Seidl H., *Locating Matches of Tree Patterns in Forests.* In V. Arvind and R. Ramamujan, editors, Foundations of Software Technology and Theoretical Computer Science, (18th FST&TCS), volume 1530 of Lecture Notes in Computer Science, 1998. pp. 134–145, Heidelberg, 1998.

[63] Nierman A. and Jagadish H. V., *Evaluating structural similarity in XML documents.* Proceedings of the ACM SIGMOD International Workshop on the Web and Databases (WebDB), 2002. pp. 61-66.

[64] Nishimura S. and Nakano K., *XML Stream Transformer Generation through Program Composition and Dependency Analysis.* Science of Computer Programming Journal, 54(2-3):257--290, 2005.

[65] Peterson D.; Gao S.; Malhotra A.; Sperberg-McQueen C.; and Thompson H. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. http://www.w3.org/TR/xmlschema11-2/ [cited January 2009].

[66] Rada R.; Mili H.; Bicknell E.; and Blettner M., *Development and Application of a Metric on Semantic Nets.* IEEE Transactions on Systems, Man, and Cybernetics, 1989. 19(1):17-30.

[67] Ray E.T., *Introduction à XML*, ed. O'Reilly. 2001, Paris. p. 327.

[68] Reis D. C.; Golgher P. B.; Silva A. S. and Laender A. F., *Automatic Web News Extraction using Tree Edit Distance.* Proceedings of the 13th International Conference on the World Wide Web (WWW '04), 2004. pp. 502-511, ACM, New York, NY, USA.

[69] Resnik P., *Using Information Content to Evaluate Semantic Similarity in a Taxonomy.* Proc. of the Inter. Joint Conf. on Artificial Intelligence (IJCAI), 1995. Vol 1, pp. 448-453.

[70] Rijsbergen van C. J., *Information Retrieval*. 1979: Butterworths, London.

[71] Sahai A. and Machiraju V., *Enabling fo the Ubiquitous e-services Vision on the Internet* Hewlett-Packard Laboratories, HPL-2001-5, 2001.

[72] Salton G. and Mcgill M.J., *Introduction to Modern Information Retrieval,*. 1983. McGraw-Hill, Tokio.

[73] Sanz I.; Mesiti M.; Guerrini G.; Berlanga La R.; and Berlanga Lavori R., *Approximate Subtree Identification in Heterogeneous XML Documents Collections.* XML Symposium, 2005, 192-206.

[74] Saruladha K.; Aghila G. and Raj S., *A Survey of Semantic Similarity Methods for Ontology Based Information Retrieval.* Proceedings of the International Conference on Machine Learning and Computing (ICMLC'10). 2010. pp. 297 - 301

[75] Schenkel R.; Theobald A.; and Weikum G., *Semantic Similarity Search on Semistructured Data with the XXL Search Engine* Information Retrieval 2005. (8):521-545.

[76] Schlieder T., *Similarity Search in XML Data Using Cost-based Query Transformations.* Proceedings of the ACM SIGMOD International Workshop on the Web and Databases (WebDB), 2001. pp. 19-24.

[77] Schlieder T. and Meuss H., *Querying and Ranking XML Documents.* Journal of the American Society for Information Science, Special Topic XML/IR, 2002. 53(6):489-503.

[78] Schöning H., *Tamoni – A DBMS Designed for XML.* Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2001. pp. 149-154.

[79] Segoufin L. and Vianu V., *Validating Streaming XML Documents.* Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), 2002. pp. 53-64.

[80] Selkow S. M., *The Tree-to-Tree Editing Problem.* . Information Processing Letters, 1977. 6(6):184–186.

[81] Shvaiko P. and Euzenat J., *A Survey of Schema-Based Matching Approaches.* Journal of Data Semantics IV, 2005. pp. 146-171.

[82] Stanoi I.; Mihaila G. and Padmanabhan S., *A framework for the selective dissemination of XML documents based on inferred user profiles.* In Proceedings of the International Conference on Data Engineering, 2003. pp. 531 – 542.

[83] Su H.; Padmanabhan S. and Lo M.L., *Identification of Syntactically Similar DTD Elements for Schema Matching.* Proceedings of the International Conference on Advances in Web-Age Information Management (WAIM), 2001. pp. 145-159.

[84] Suzuki N., *Finding an Optimum Edit Script between an XML Document and a DTD.* Proceedings of the ACM Symposium on Applied Computing (ACM SAC), 2005. pp. 647-653.

[85] Tekli J.; Chbeir R. and Yétongnon K., *Structural Similarity Evaluation between XML Documents and DTDs.* Proceedings of the International Conference on Web Information Systems Engineering (WISE), 2007. pp. 196-211.

[86] Tekli J.; Chbeir R. and Yétongnon K., *An Overview of XML Similarity: Background, Current Trends and Future Directions.* Elsevier Computer Science Review, 2009. 3(3):151-173.

[87] Tekli J.; Chbeir R. and Yétongnon K., *XML Grammar Similarity: Breakthroughs and Limitations.* Second Edition of the Encyclopedia of Multimedia Technology and Networking, Information Science Reference, 2009. Hershey - New York, 2 (1): 140-148.

[88] Tekli J.; Damiani E.; Chbeir R. and Gianini G., *SOAP Processing Performance and Enhacement.* To appear in IEEE Transactions on Service Computing (IEEE TSC), 2011.

[89] Teraguchi M.; Makino S.; Ueno K. and Chung H.V., *Optimized Web Services Security Performance with Differential Parsing.* Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC'06), 2006. pp. 277-288.

[90] Theobald A. and Weikum G., *Adding Relevance to XML.* Proceedings of the 3rd International Workshop on the Web Databases (WebDB), 2000. pp. 105-124. Dallas, USA.

[91] Werner C.; Buschmann C. and Fischer S., *WSDL-Driven SOAP Compression.* International Journal of Web Services Research, 2005. Vol. 2, Issue 1, pp. 18-35.

[92] Word Wide Web Consortium. *SOAP Version 1.2.* W3C Recommendation (Second Edition) 2007, http://www.w3.org/TR/soap/ [cited February 2010].

[93] World Wide Web Consortium. *The Document Object Model*. http://www.w3.org/DOM [cited 28 May 2009].

[94] Xing G., *Fast Approximate Matching Between XML Documents and Schemata.* The Asia Pacific Web Conference (APWeb'06), 2006. pp. 425-436.

[95] Xing G.; Xia X. and Guo J., *Clustering XML Documents Based on Structural Similarity.* International Conference of Database Systems for Advanced Applications (DASFAA'07), 2007, 905-911.

[96] Zhang K. and Shasha D., *Simple Fast Algorithms for the Editing Distance between Trees and Related Problems.* SIAM Journal of Computing, 1989. 18(6):1245-1262.

[97] Zhang K.; Shasha D. and Wang J., *Approximate Tree Matching in the Presence of Variable Length Don't Cares.* Journal of Algorithms, 1994. (16) 33-66.

[98] Zhang X.; Jing L.; Hu X.; Ng M. and Zhou X., *A Comparative Study of Ontology Based Term Similarity Measures on PubMed Document Clustering.* Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA' 07), 2007. pp. 115-126.

[99] Zhang Z.; Li R.; Cao S.; and Zhu Y., *Similarity Metric in XML Documents.* Knowledge Management and Experience Management Workshop, 2003.