A Novel XML Document Structure Comparison Framework based-on Sub-tree Commonalities and Label Semantics

Joe Tekli [a,1] and Richard Chbeir [b]

[a] ICMC Computer Science and Statistics Institute, University of Sao Paulo, 13566-590 Sao Carlos, SP, Brazil
[b] LE2I Laboratory UMR-CNRS, University of Bourgogne, 21078 Dijon Cedex France

ABSTRACT

XML similarity evaluation has become a central issue in the database and information communities, its applications ranging over document clustering, version control, data integration and ranked retrieval. Various algorithms for comparing hierarchically structured data, XML documents in particular, have been proposed in the literature. Most of them make use of techniques for finding the edit distance between tree structures, XML documents being commonly modeled as Ordered Labeled Trees. Yet, a thorough investigation of current approaches led us to identify several similarity aspects, i.e., sub-tree related structural and semantic similarities, which are not sufficiently addressed while comparing XML documents. In this paper, we provide an integrated and fine-grained comparison framework to deal with both structural and semantic similarities in XML documents (detecting the occurrences and repetitions of structurally and semantically similar sub-trees), and to allow the end-user to adjust the comparison process according to her requirements. Our framework consists of four main modules for i) discovering the structural commonalities between sub-trees, ii) identifying sub-tree semantic resemblances, iii) computing tree-based edit operations costs, and iv) computing tree edit distance. Experimental results demonstrate higher comparison accuracy with respect to alternative methods, while timing experiments reflect the impact of semantic similarity on overall system performance.

Keywords: XML, Semi-structured Data, Structural Similarity, Tree Edit Distance, Semantic similarity, Information Retrieval, Vector Space Model.

1. Introduction

In the past few years, XML has emerged as the main standard for data exchange on the Web. The ever-increasing amount of information available on the Internet has reflected the need to bring more structure and semantic richness, and thus more flexibility, in representing data, which is where W3C's XML (eXtensible Markup Language) comes to play. The use of XML covers data description and storage (e.g., complex multimedia objects such as SVG images [86], X3D graphics [82], MPEG-7 meta-data [50] …), database information interchange, data filtering, as well as web services interaction.

Owing to the increasing web exploitation of XML, XML document comparison becomes a central issue in the database and information retrieval communities. The applications of XML document comparison range over: change management and data warehousing (finding, scoring and browsing changes between different versions of a document, support of temporal queries and index maintenance) [12-14], data integration (identifying and merging similar documents to provide a more complete view of the data) [29, 39], XML retrieval (finding and ranking results according to their similarity) [66, 90], as well as the clustering of XML documents gathered from the web [16, 55] which would improve storage indexing [68] and thus positively affect the retrieval process.

The main goal of our study is the comparison of rigorously structured *heterogeneous* XML documents, i.e., documents originating from different data-sources and not conforming to the same grammar (DTD/XSD), which is the case of a lot of XML documents found on the Web [55]. In fact, a range of solutions for comparing semi-structured (XML) data has been proposed in the literature. On one hand, most algorithms make use of techniques for finding the edit distance between tree structures [12, 16, 55], XML documents being treated as Ordered Labeled Trees (OLTs) [85]. On the other hand, some works have focused on extending conventional information retrieval methods, e.g., [5, 11], so as to provide efficient XML similarity assessment. In this study, we bound our presentation to the former group of methods, i.e., edit distance based approaches, since they target rigorously structured XML documents and are usually more fine-grained, mainly exploited in data-warehousing, version control, structural querying and XML classification and clustering applications (Information retrieval based methods, on the other hand, target loosely structured XML data with long text fields – text-rich, and are usually coarse-grained, mainly useful for fast simple XML retrieval [26, 28]). We particularly focus on comparing XML document structures,

[1] Corresponding author. Tel: +55-16-33739677, Fax: +55-16-33739751 ; E-mail address: joe.tekli@icmc.usp.br, jtekli@gmail.com
The author is currently with the Antonine University (UPA), Lebanon

i.e., the structural disposition and ordering of element/attribute tag names[1] (central in XML structural classification and clustering applications, e.g., [10, 55]), and disregard XML contents (i.e., element/attribute values). In short, we view XML document structure comparison as an independent line of study, as well as an essential and indispensable step to consequently address element/attribute contents efficiently. In this context, two main problems arise:

- *Elements' structural similarity*: this consists in considering parent/child relationships and ordering among XML elements, identified by their tag labels. In essence, a thorough investigation of the most recent and efficient XML structural similarity approaches [12, 16, 55] led us to pinpoint certain cases where the comparison outcome is inaccurate. These inaccuracies correspond to undetected sub-tree structural similarities, as we will see in the motivating examples.
- *Elements' semantic similarity*: this consists in evaluating the semantic meanings of XML element/attribute labels. Most existing XML comparison approaches focus exclusively on the structure of XML documents, ignoring the semantics involved. However, evaluating the semantic relatedness between documents (mainly those published on the Web) is of key importance to improving search results: finding related documents, and given a set of documents, effectively ranking them according to their similarity [44].

The relevance of semantic similarity in Web search mechanisms, as well as the increasing use of XML-based structured documents on the Web, motivated us to study XML similarity in both its *structural* and *semantic* facets and to provide a hybrid XML similarity method for comparing heterogeneous XML documents. We aim to develop a parameterized XML comparison approach able to i) efficiently detect XML structural similarity (preliminary work has appeared in [74, 76]), ii) consider semantic relatedness while comparing XML documents, and iii) allow the user to tune XML comparison according to the scenario and application requirements by assigning more importance to either structural or semantic similarity (using an input structural/semantic parameter). The contributions of our study can be summarized as follows. First, we provide a unified framework in which we extend and combine existing structure comparison approaches, mainly those provided in [12, 55], in order to consider the various sub-tree structural similarities while comparing XML document trees. Second, we expand XML structural similarity evaluation, combining the traditional vector space model in information retrieval [47] and semantic similarity assessment [41], to consider sub-tree semantic similarities in comparing XML documents. Such similarities encompass the evaluation of semantic relatedness between XML node labels w.r.t. (with respect to) a reference semantic information source. Third, we implement our framework as an experimental prototype to test and evaluate our approach. Experimental results reflect our method's high accuracy and performance levels in comparison with existing solutions.

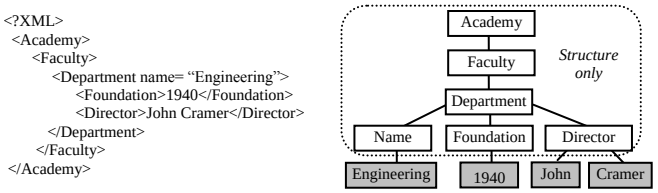
[1] Note that the XML tree structure is different from topological tree structure since it relies on labels (i.e., element/attribute tag names) in identifying corresponding tree nodes, whereas the latter only considers the interconnections among nodes, disregarding the nodes labels.

The remainder of this paper is organized as follows. Section 2 reviews background and related works in XML structural comparison and semantic similarity evaluation. Section 3 presents motivation examples highlighting different kinds of undetected XML similarities to be addressed in our study. Section 4 develops our integrated XML document comparison approach. Section 5 provides theoretical and computational comparative analyses, evaluating our method against existing solutions. Section 6 presents our prototype and experimental tests. Section 7 concludes with ongoing works.

2. Background

2.1 XML Data Model

XML documents represent hierarchically structured information and are generally modeled as Ordered Labeled Trees or OLTs (Fig. 1). In a traditional DOM (Document Object Model) ordered labeled tree [85], nodes represent XML elements, and are labeled with corresponding element tag names, ordered following their order of appearance in the document. Attributes usually appear as children of their encompassing element nodes, sorted by attribute name, and appearing before all sub-element siblings [55, 90]. Other types of nodes, such as entities, comments and notations, are commonly disregarded in most XML comparison approaches, e.g., [12, 16, 23, 31, 55], since they underline complementary information and are not part of the core XML data.



a. XML Document. **b.** XML document tree (OLT).

Fig. 1. A sample XML document with corresponding OLT.

In general, element/attribute values are disregarded when evaluating the structural properties of *heterogeneous* XML documents (originating from different data-sources and not conforming to the same grammar), so as to perform XML structural classification/clustering [16, 31, 55, 58] or structural querying (i.e., querying the structure of documents, disregarding content [6, 64]). Nonetheless, values are usually taken into account with methods dedicated to XML change management [13, 14], data integration [29, 40], and XML *structure-and-content* querying applications [66, 67], where documents tend to have similar structures (probably conforming to the same grammar [36, 83]).

2.2 Structural Similarity and Tree Edit Distance

Various methods for estimating the similarities between hierarchically structured data, particularly between XML documents, have been proposed in the literature. Most of them exploit the concept of tree edit distance, deriving, in one way or another, the dynamic programming techniques for finding the edit distance between strings [37, 81, 84].

In the following, we provide the basic notions related to the concept of tree edit distance, and briefly review the corresponding literature.

2.2.1 Tree Edit Distance: Basic Notions and Concepts

Hereunder, we provide two basic definitions describing the concept of tree edit distance.

Definition 1 – Edit Script: It is a sequence of edit operations $ES = \prec op_1, op_2, \ldots, op_k \succ$. When applied to a tree T, the resulting tree T' is obtained by applying edit operations of the edit script ES to T, following their order of appearance in the script. By assigning a cost, $Cost_{Op}$, to each edit operation, the cost of an edit script is defined as the sum of the costs of its component operations: $Cost_{ES} = \sum_{i=1}^{|ES|} Cost_{Op_i}$ [7, 12] ●

Definition 2 – Tree Edit Distance: The edit distance between two trees A and B is defined as the minimum cost of all edit scripts that transforms A to B, $TED(A, B)=Min\{Cost_{ES}\}$. Thus, the problem of comparing two trees A and B, i.e., evaluating the structural similarity between A and B, is defined as the problem of computing the corresponding tree edit distance, i.e., minimum cost edit script [89] ●

As for tree edit operations, they can be classified in two groups: *atomic* operations and *complex* operations [16]. An atomic edit operation on a tree (i.e., rooted ordered labeled tree) is either the deletion of an inner/leaf node, the insertion of an inner/leaf node, or the replacement (i.e., update) of a node by another one. A complex tree edit operation is a set of atomic tree edit operations, treated as one single operation, e.g., the insertion of a whole tree as a sub-tree in another tree (which amounts to a sequence of atomic node insertion operations), the deletion of a whole tree (i.e., a sequence of atomic node deletion operations), or moving a sub-tree from one position into another in its containing tree (i.e., a sequence of atomic node insertion/deletion operations). In Section 4.1, we provide the formal definitions for each of the tree edit operations utilized in our approach.

2.2.2 Current Tree Edit Distance Methods

Tree edit distance algorithms can be distinguished by the set of edit operations that they allow as well as their overall complexity/performance and optimality/efficiency levels.

Early approaches: In [72], the author introduces the first non-exponential algorithm to compute the edit distance between ordered labeled trees, allowing insertion, deletion and substitution (relabeling) of inner nodes and leaf nodes. The resulting algorithm has a complexity of $O(|A||B|\times depth(A)^2\times depth(B)^2)$ when finding the edit distance between two trees A and B ($|A|$ and $|B|$ denote tree cardinalities while $depth(A)$ and $depth(B)$ are the depths of the trees). Similarly, early approaches in [70, 89] allow insertion, deletion and relabeling of nodes anywhere in the tree. Yet, they remain greedy in complexity. For instance, the algorithm in [70] is of $O(|A||B|\times depth(A) \times depth(B))$. In addition, the approaches in [70, 72, 89] were not developed in the XML context, and thus might yield results that are not appropriate to XML data.

Quality Versus Performance: In [13, 14], the authors restrict insertion and deletion operations to leaf nodes and add a move operator that can relocate a sub-tree, as a single edit operation, from one parent to another. Yet, algorithms in [13, 14] do not guarantee optimal results. In [13], the documents being compared should match specific criterions and assumptions without which the algorithm would yield suboptimal results. The algorithm's complexity simplifies to $O(n\times e + e^2)$, where n is the total number of leaf nodes in the trees being compared and e is the corresponding weighted edit distance[1]. On the other hand, the authors in [14] trade some quality (the edit distance obtained is not always minimal, some sets of move operations not being optimal) to get an algorithm which runs in average linear time: $O(N\ log(N))$ where N is the number of nodes in the compared trees.

Methods in [13, 14] were developed for XML change management and version control. They consider XML element/attribute values (XML *structure-and-content*, Fig. 1.b) in contrast with remaining methods in this section which target the structural properties of XML documents (*structure-only*).

Combining efficiency and performance: The approach provided in [12] restricts insertion and deletion operations to leaf nodes (which are viewed as natural operations in the XML context [16]), and allows the relabeling of nodes anywhere in the tree, while disregarding the move operation. The proposed algorithm is a direct application of the famous Wagner-Fisher algorithm [81] which optimality has been accredited in a broad variety of computational applications [2, 84]. It is also among the fastest tree edit distance algorithms available. Chawathe [12] extends his algorithm for external-memory computations and identifies respective I/O, RAM and CPU costs. The overall complexity of Chawathe's algorithm is of $O(N^2)$.

Sub-tree similarity: In [55], the authors stress the importance of identifying sub-tree structural similarities in XML comparison, due to the frequent presence of *repeated* and *optional* elements in XML document trees. Repeating elements often induce multiple occurrences of similar element/attribute sub-trees (presence of optional elements/attributes) or identical sub-trees in the same document (such as sub-trees B_1 and B_2 in XML tree B, Fig. 2) which reflects the need to consider these sub-tree resemblances while comparing documents.

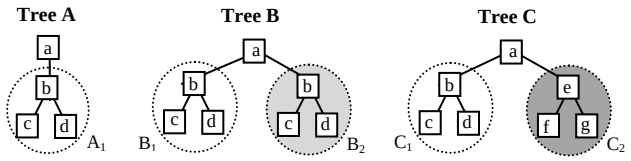


Fig. 2. Sample XML trees, with sub-tree repetitions.

The authors in [55] extend the approach of Chawathe [12] by adding two new operations: *insert tree* and *delete tree*, to discover sub-tree similarities, making use of the *contained in* relation between trees/sub-trees. A tree S is said to be *contained in* a tree T if all nodes of S occur in T, with the same

[1] Let $S = \prec op_1, op_2, \ldots, op_n \succ$ be the cheapest sequence of edit operations that transforms tree A to B, then the weighted edit distance is given by $e = \sum_{1\le i\le n} w_i$ where w_i, for $1\le i\le n$, is equal to *1* if op_i is an insert or delete operation, and *0* otherwise.

parent/child edge relationship and node order. Following [55], when comparing two trees *A* and *B*, a sub-tree *S* may be inserted (deleted) in *A* only if *S* is already *contained in* the source tree *A* (destination tree *B*). Therefore, the proposed approach captures the sub-tree structural similarities between XML trees *A/B* in Fig. 2, transforming *A* to *B* in a single edit operation (inserting sub-tree B_2 in *A*, sub-tree B_2 occurring in tree *A* as A_1), which is less costly (and thus yields a lower distance, i.e., higher similarity) than transforming *A* to *C*, which requires three operations (inserting nodes *e*, *f* and *g*).

The overall complexity of the algorithm in [55] simplifies to $O(N^2)$, including a pre-computation phase for determining the costs of tree insert/delete operations (which is of $O(2\times N + N^2)$ time). Structural clustering experiments in [55] show that the proposed algorithm outperforms those in [12, 89].

Structural summaries: On the other hand, Dalamagas *et al.* [16] provide an edit distance algorithm combining features from both [12, 55] and propose to apply it on XML tree structural summaries, instead of whole trees, in order to gain in performance. Structural summaries are produced using a special repetition/nesting reduction process (e.g., the structural summary of tree *B* of Fig. 2 would be tree *A*). The algorithm is of $O(N^2)$ time. Experimental results in [16] show improved clustering quality w.r.t. Chawathe's algorithm [12]. Note that while it might be useful for structural clustering tasks, Dalamagas *et al.*'s reduction process yields inaccurate comparison results in the general case (e.g., $Dist(A, B) = 0$ despite their differences) which is why it is disregarded in the remainder of our discussions.

Other methods to XML structural similarity have also been proposed. They exploit various techniques (e.g., edge matching [38], path similarity [58], the Fast Fourier Transform [23], and entropy [31], etc.), usually providing approximations of (more complex and accurate) tree edit distance approaches. Such tree edit distance alternative and approximation methods have been thoroughly investigated in [77], and thus will not be covered in this paper. Here, we consider tree edit distance to be the "optimal" technique for assessing similarity among structured documents [9], and hence focus on tree edit distance for XML structural comparison.

2.3 Semantic Similarity

Measures of semantic similarity are of key importance in evaluating the effectiveness of Web search mechanisms in finding and ranking results [44]. In the fields of Natural Language Processing (NLP) and Information Retrieval (IR), knowledge bases (i.e., ontologies, thesauri and/or taxonomies, such as ODP [44], Roget's thesaurus [88], WordNet [48], etc.) provide a framework for organizing words/expressions into a semantic space [33]. A knowledge base usually comes down to a semantic network made of a set of concepts representing groups of words/expressions (or URLs such as with ODP), and a set of links connecting the concepts, representing semantic relations (synonymy, hyponymy, etc. [48, 61], Fig. 3). Hence, evaluating semantic similarity between words/expressions comes down to comparing the underlying concepts in the semantic space.

Indeed, several methods have been proposed to determine semantic similarity between concepts (and consequently related terms) in a knowledge base (semantic network). They can be categorized as: edge-based approaches and node-based approaches [33].

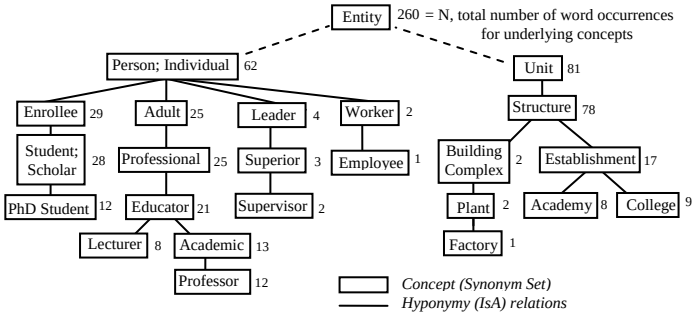


Fig. 3. A (weighted) taxonomy fragment extracted from WordNet. The numbers next to concepts represent concept frequencies (computed based on the Brown text corpus [25]).

2.3.1 Edge-based Approaches

Edge-based methods underline an intuitive and straightforward way to evaluate semantic similarity in a semantic network. They generally estimate similarity as the shortest path (in edges, or number of nodes) between the two concepts being compared: the shorter the path from one node to another, the more similar they are [34, 35, 57]. On the other hand, the authors in [71, 87] evaluate semantic similarity between two concepts by identifying their most specific common ancestor. The similarity measures employed consider the distance between the compared nodes and their common ancestor, as well as the distance separating the common ancestor from the root of the semantic network.

However, a known problem with edge-based approaches is that they often rely on the idea that links, in the semantic network, represent uniform distances [33, 60]. In real semantic networks, the distance covered by a single link can vary with regard to network density, node depth and information content of corresponding nodes [61]. The authors in [33] add that link distances could also vary according to link type (i.e., semantic relation type). In an attempt to solve the varying distance problem, the authors in [33, 61] suggest weighting links according to the above mentioned characteristics.

2.3.2 Node-based Approaches

Node-based approaches get round the problem of varying link distances by incorporating an additional knowledge source: corpus statistical analysis, to augment the information already present in the semantic network. In fact, with node-based approaches, the definition of similarity is estimated as the maximum amount of information content they share in common [33, 60]. In a hierarchical semantic network (i.e., taxonomy, cf. Fig. 3), this common information carrier can be identified as the most specific common ancestor (also known as Lowest Common Ancestor, or LCA) that subsumes both concepts being compared [60] (e.g., $LCA(Lecturer, Professor) = Educator$ in Fig. 3). Consequently, the similarity between two concepts is defined as the information content of their

lowest common ancestor, obtained by estimating its probability of occurrences in a large text corpus [60].

Definition 3 – Information Content: In information theory, the information content of a concept or class c is quantified as the negative log likelihood $-\log p(c)$ where $p(c)$ is the probability of encountering an instance of c [60] ●

Definition 4 – Probability of a Concept: It is generally quantified with respect to the frequency of occurrence of the words/expressions, subsumed by the corresponding concept, in a given corpus [33, 60] ●

Slightly different mathematical formulations [33, 60] have been utilized to compute concept probabilities. Here, we present the basic formulation by Resnik [60]:

$$p(c) = \frac{Freq(c)}{N} \quad \textbf{(1)}$$

- $Freq(c) = \sum_{w \in words(c)} count(w)$: Number of occurrences of words subsumed by c, in a given corpus,
- N: Total number of words encountered in the corpus.

Since in hierarchical semantic networks (i.e., taxonomies, consisting mainly of hierarchical semantic relationships, e.g., *Is-A*, *Part-Of*…), concepts subsume those lower in the hierarchy, $Freq(c)$ and consequently $p(c)$ increase as one moves up the hierarchy (the occurrence of a word is counted for its corresponding concept, as well as the concept's ancestors). Thus, following Definition 4, nodes higher in the hierarchy (with higher probabilities) are less informative (more abstract). If the semantic network has a root node (otherwise a virtual root is usually added), then its probability would be equal to *1*, its information content being equal to *0*.

Fig. 3 depicts an extract of WordNet weighted with pre-computed concept frequencies based on a sample text corpus (e.g., Brown Corpus of American English [25]). *Formula (2)* presents a variation of the node-based measures by Resnik [60]:

$$Sim_{Node}(w_1, w_2,) = Sim_{Node}(c_1, c_2, \overline{SN}) = -\log(p(c_0)) \quad \textbf{(2)}$$

- c_1 and c_2 are the semantic concepts corresponding to the words (expressions) w_1 and w_2 being compared [1],
- c_0 is the most specific common ancestor of c_1 and c_2,
- $p(c_0)$, the occurrence probability of concept c_0 (cf. Definition 4, *Formula (1)*),
- $\overline{SN}$ underlines the weighted semantic network (cf. Fig. 3), i.e., a semantic network *SN* augmented with concept frequencies (i.e., concept weights).

Following Resnik [60], the semantic similarity between two concepts in the semantic network is approximated by the information content of their most specific common ancestor. Resnik's experiments [60] show that his similarity measure is a better predictor of human word similarity ratings, in comparison with a variant of the edge-based methods [35, 57].

Improving on Resnik's method [60], Lin [41] presents a formal definition of the *intuitive* notion of similarity, and derives an information content measure from a set of predefined assumptions regarding commonalities and differences. Following [41], the commonality between two concepts is underlined by the information content of their lowest common ancestor (identified by Resnik's measure [60]). However, the difference between concepts depends on their own information contents (disregarded in [60]):

$$Sim_{Lin}(c_1, c_2, \overline{SN}) = \frac{2 \log p(c_0)}{\log p(c_1) + \log p(c_2)} \quad \textbf{(3)}$$

- c_0 is the most specific common ancestor of c_1 and c_2,
- $p(c_0)$ denotes the occurrence probability of concept c_0.

When comparing two concepts c_1 and c_2, Lin's measure [41] takes into account each concept's information content ($-\log p(c_1) + -\log p(c_2)$), as well as the information content of their most specific common ancestor ($-\log p(c_0)$), in a way to increase with commonality (information content of c_0) and decrease with difference (information content of c_1 and c_2). Lin's experiments [41] show that the latter information content measure yields higher correlation with human judgment in comparison with Resnik's [60] measure. Furthermore, Lin's measure which targets hierarchical structures, i.e., taxonomies (as most existing semantic similarity measures) is generalized in [44] to deal with ontologies of hierarchical (made by *Is-A* links) and non-hierarchical components (made by cross links of different types, e.g., *RelatedTo*…). Another interesting extension of Lin's measure is provided in [24] to semantically compare two groups of concepts, and to evaluate concept similarity in geographic information systems [15]. A more recent variation of Lin's measure was introduced in [69], providing a new approach to compute information content based solely on the hierarchical structure of a semantic network (namely WordNet [48]), disregarding corpus statistics.

2.4 Integrating Structural and Semantic Similarity

In recent years, there have been a few attempts to integrate semantic and structural similarity assessment in the XML comparison process. The INEX (INitiative for the Evaluation of XML Retrieval[2]) campaigns have stressed the relevance of semantic similarity assessment in XML retrieval. One of the early approaches to propose such a method is [78], where the authors make use of a textual similarity operator and utilize Oracle's InterMedia text retrieval system to improve XML similarity search. In a recent extension of their work [65], the authors define a generic ontological model, built on WordNet, to account for semantic similarity (instead of utilizing Oracle InterMedia). However, INEX related approaches focus on textual similarity (i.e., similarity between element/attribute values made of long text fields) which is out of the scope of our study since in structure-based similarity, values are commonly disregarded.

Recent XML structure-based methods in [6, 64] identify the need to support tag similarity (synonyms and stems) instead of tag syntactic equality while comparing XML documents. In [42], the authors introduce a structure and

[1] Semantic concepts are identified after several linguistic pre-processing operations such as tokenization, stemming, and word sense disambiguation. These are briefly discussed in Section 4.1.

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

content based method for comparing XML documents having the same grammar (i.e., not heterogeneous), and consider semantic similarity evaluation between element/attribute values, using a variation of the edge-based methods. In [73], the authors introduce a hybrid XML similarity approach integrating Chawathe's tree edit distance algorithm [12], with semantic similarity using Lin's measure [41] to compare XML tag names. Methods in [42, 73] produce asymmetric similarity measures.

2.5 Discussion

On one hand, various methods have been proposed to evaluate XML structural similarity (i.e., comparing the hierarchical relations and ordering among XML elements, identified by their labels). Most methods in this family are based on the concept of tree edit distance as an optimal technique to compare structured data. On the other hand, a range of techniques have been developed for semantic similarity evaluation (comparing word/expression concepts in a reference knowledge base). Most methods in this category compare the information content values of concepts in a semantic network. Nonetheless, despite the rich literatures in XML similarity and semantic similarity, few methods have addressed the problem of integrating XML structure and XML tag (or value) semantics to improve similarity evaluation. That is probably due to the relative novelty of the XML semantic/structural similarity problem. As will be shown in the following sections, various kinds of XML (sub-tree related) structure and semantic similarities remain unaddressed by most existing methods. Taking into account such resemblances would obviously amend XML comparison effectiveness.

Note that the issue of integrating structural and semantic similarity evaluation has also been investigated in the contexts of schema matching/integration [3, 4, 18], as well as ontology mapping/mediation [46, 51, 52]. Yet, while comparable to tree-based XML documents, schemas and ontologies often underline more intricate graph structures, and thus require graph-based algorithms and heuristics in evaluating similarity, which are out of the scope of this paper (here, we limit our presentation to XML tree-based approaches).

3. Motivations

The main objective of this study is to provide a fine-grained method that captures both structural and semantic similarities when comparing XML document structures. Hereunder, we discuss the motivations of our work, highlighting the relevance of *structural* and *semantic* similarity evaluation in XML document comparison. We specifically focus on similarities left unaddressed in current approaches, which we aim to capture with our XML document similarity measure.

3.1. Structural Similarity

XML documents can encompass many optional and repeated elements [55]. Such elements induce recurring sub-trees of similar or identical structures. As a result, algorithms for comparing XML documents should be aware of such repetitions/similarities to effectively assess structural similarity.

Our examination of existing XML structural comparison approaches, particularly fine-grained approaches based on tree edit distance, e.g., [12, 16, 55], have led us to identify certain cases where sub-tree structural similarities are disregarded. These undetected similarities can be distinguished as:

- Repetitions of structurally similar sub-trees,
- Structural similarity between sub-trees occurring at different depths,
- Similarity between a sub-tree on one hand, and the whole XML tree on the other,
- Repetitions of leaf node sub-trees.

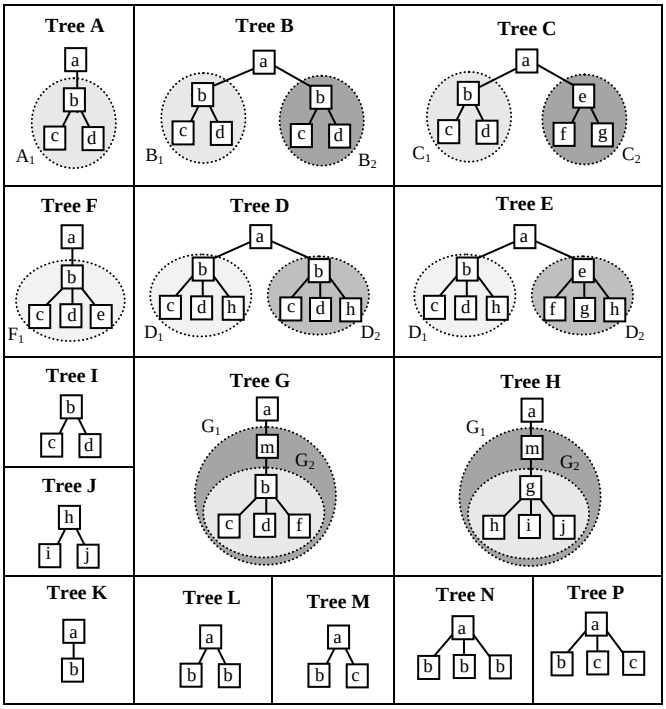


Fig. 4. Dummy XML trees, depicting various kinds of sub-tree structural similarities.

The authors in [55] make use of tree insertion and tree deletion operations, coupled with the *contained in* relation between trees, to capture sub-tree repetitions, such as the case of XML trees *A*/*B* and *A*/*C* mentioned in Section 2.2.2 (repetition of sub-tree B_1). Yet, when the containment relation is not fulfilled, certain structural similarities are ignored.

Consider, for instance, trees *A* and *D* in Fig. 4. Here, the XML document sub-trees being repeated are not *contained in* the source tree, but are similar (e.g., D_2 and A_1 are similar). Since D_2 is not contained in *A*, it is inserted *via* four edit operations instead of one (insert tree), while transforming *A* to *D*, ignoring the fact that part of D_2 (sub-tree of nodes *b*, *c*, *d* [1]) is identical to A_1. Therefore, equal distances are obtained when comparing trees *A*/*D* and *A*/*E*, disregarding *A*/*D*'s structural resemblances (here, we assume the general case where atomic insertion/deletion operations are of unit costs, =1):

- $Dist(A, D) = Cost_{Ins}(h) + Cost_{Ins}(b) + Cost_{Ins}(c) + Cost_{Ins}(d) + Cost_{Ins}(h) = 1 + 4 = 5$
- $Dist(A, E) = Cost_{Ins}(h) + Cost_{Ins}(e) + Cost_{Ins}(f) + Cost_{Ins}(g) + Cost_{Ins}(h) = 1 + 4 = 5$

[1] In the examples, we designate nodes by their labels for simplicity.

Other types of sub-tree structural similarities that are missed by existing approaches can also be identified when comparing trees *F*/*G* and *F*/*H*, as well as *F*/*I* and *F*/*J*. The *F*, *G*, *H* case is different than its predecessor (the *A*, *D*, *E* case) in that the sub-trees sharing structural similarities (F_1 and G_2) occur at different depths (whereas with *A*/*D*, A_1 and D_2 are at the same depth). Here, the approaches in [12, 16, 55] for instance yield identical distance values when comparing trees *F* and *G*, as well as *F* and *H*, disregarding the structural similarity between sub-trees F_1 and G_2 (in comparison with F_1 and H_2):

- Dist(F, G) = Dist(F, H) = 7, which is the cost of updating node *b*, transforming it into *m*, deleting nodes *c*, *d* and *e* of tree *F*, and inserting sub-tree G_2 (H_2) into tree *F*.

On the other hand, the *F*, *I*, *J* case differs from the previous ones since structural similarities occur not only among sub-trees, but also at the sub-tree/tree level (e.g., between sub-tree F_1 and tree *I*). Such similarities are usually disregarded with existing methods, e.g., [12, 16, 55]:

- Dist(F, I) = Dist(F, J) = 6, which is the cost of updating root node *a* of tree *F*, transforming it into *b* (*h*), updating node *b* into *c* (*i*), deleting nodes *c*, *d* and *e*, and inserting node *c* (*j*) into tree *I* (*J*).

In addition, none of the approaches mentioned above is able to effectively compare documents made of repeating leaf node sub-trees. For example, following [12, 16, 55], identical similarity values are obtained when comparing document *K*, of Fig. 4, to documents *L* and *M*. That is because most existing approaches consider minimum unit (*=1*) operations costs, regardless of the leaf nodes involved in the operations.

- Dist(*K*, *L*) = $\text{Cost}_{\text{Ins}}(b) = 1$
- Dist(*K*, *M*) = $\text{Cost}_{\text{Ins}}(c) = 1$

Nonetheless, one can realize that document trees *K* and *L* are more similar than *K* and *M*, node *b* of tree *K* appearing twice in tree *L*, and only once in XML tree *M*. Likewise, identical distances are attained when comparing document trees *K*/*N* and *K*/*P*, despite the fact that the node *b* is repeated three times in tree *N*, and only once in tree *P*. In this study, we explicitly mention the case of leaf node repetitions since i) leaf nodes are a special kind of sub-trees: *single node sub-trees*, ii) leaf node repetitions are usually as frequent as sub-tree repetitions in XML documents, and iii) detecting leaf node repetitions would help increase the discriminative power of XML comparison methods as described in the above examples.

3.2. Semantic Similarity

In order to stress the need for semantic relatedness assessment in XML document comparison, we first report from [73] the sample XML document trees in Fig. 5. Using classic edit distance computations (e.g., [12, 16, 55]), the same structural similarity value is obtained when document *X* is compared to documents *Y* and *Z*:

- Dist(X, Y) = Dist(X, Z) = 3, corresponding to the cost of updating root node of label *Academy* transforming it into *College* (*Factory*), updating node *Professor* transforming it into *Lecturer* (*Supervisor*), and deleting node *Student*.

However, despite having similar structural characteristics, one can easily recognize that sample document *X* shares more semantic characteristics with document *Y* than with *Z*. For instance, node labels *Academy-College* and *Professor-Lecturer*, from documents *X* and *Y*, can be commonly viewed as semantically more similar than *Academy-Factory* and *Professor-Supervisor*, from documents *X* and *Z* (considering a domain independent semantic network such as WordNet [48], describing concepts found in everyday language,). Therefore, taking into account the semantic factor in XML similarity computations would obviously amend similarity results.

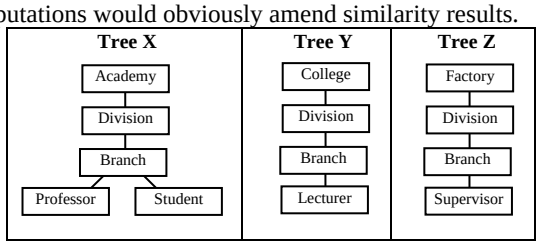


Fig. 5. Sample XML document trees, with semantically meaningful node labels.

The example in Fig. 5 underlines semantic similarities between XML nodes with identical structural positions (i.e., identical depth and ordering). Such relatively simple semantic similarities have been covered in [73]. The authors in [73] complement Chawathe's tree edit distance algorithm [12], with a 'semantic cost scheme' taking into account semantic similarities between XML node labels in assigning edit operations costs. They make use of Lin's semantic similarity measure developed in [41], provided a given reference semantic network. Nonetheless, the approach in [12] was not designed to capture sub-tree repetitions and resemblances, making use of single node-based edit operations (i.e., node update, leaf node insertion and leaf node deletion, cf. Section 2.2.2). The same goes for its extension in [73] which is not concerned with repetitions of semantically similar sub-trees.

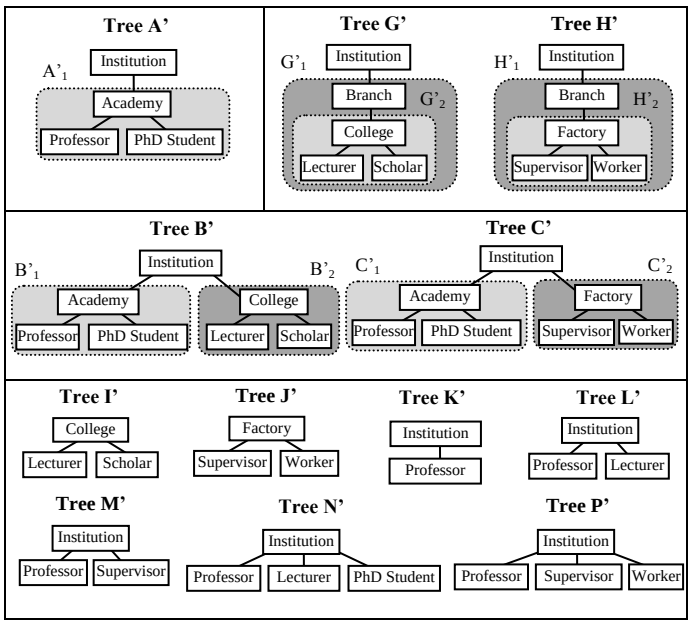


Fig. 6. Sample XML trees with sub-tree semantic similarities.

Consider the sample XML document trees in Fig. 6. Here, as with the sub-tree structural similarity examples mentioned in the previous section, different types of undetected sub-tree semantic similarities can also be identified:

- Occurrence of semantically similar sub-trees,
- Semantic similarity between sub-trees occurring at different depths,
- Semantic similarity between a sub-tree on one hand, and the whole XML tree on the other,
- Occurrence of semantically similar leaf node sub-trees.

Recall the *A*, *B*, *C* and *A*, *D*, *E* comparison cases in Fig. 4. XML trees *A* and *B* (likewise *A* and *D*) are structurally more similar than *A* and *C* (respectively *A* and *E*) due to the occurrence of structurally identical (similar) sub-trees, i.e., A_2 (D_2) in tree *B* (tree *D*). In Fig. 6, XML document trees *A'*, *B'* and *C'* underline a similar scenario. While trees *B'* and *C'* are structurally indistinguishable w.r.t. tree *A'*, one can realize that *A'* is semantically more similar to *B'*, than to *C'*. Sub-tree A'_1 made of nodes *Academy*, *Professor* and *PhD Student* is semantically similar to sub-tree B'_2 (made of nodes *College*, *Lecturer* and *Scholar*) in tree *B'*, while it is semantically different than sub-tree C'_2 (of nodes *Factory*, *Supervisor* and *Worker*) in tree *C'* (w.r.t. a generic reference semantic network such as WordNet [48]). In other words, instead of only considering the occurrence and repetition of identical or structurally similar sub-trees (as discussed in the previous section), there is a need to consider the occurrences of sub-trees that are semantically similar as well.

In addition, as with the sub-tree structural similarity examples in Fig. 4, similar types of sub-tree semantic similarities can be identified when comparing trees *A'*/*G'* and *A'*/*H'*, *A'*/*I'* and *A'*/*J'*, *K'*/*L'* and *K'*/*M'*, as well as *K'*/*N'* and *K'*/*P'*. The *A'*, *G'*, *H'* case is different in that the sub-trees sharing semantic similarities (A'_1 and G'_2) occur at different depths. The *A'*, *I'*, *J'* case differs from its predecessors in that semantic similarities occur, not only among sub-trees, but also at the sub-tree/tree level (e.g., between sub-tree A'_1 and tree *I*). On the other hand, the *K'*, *L'*, *M'* and *K'*, *N'*, *P'* cases correspond to leaf node semantic similarities. Here, one can realize that document trees *K'* and *L'* are more similar than *K'* and *M'*, node *Professor* of tree *K'* being semantically more similar to node *Lecturer* in tree *L'*, than to node *Supervisor* in tree *M'*. Likewise for *K'*/*N'* with respect to *K'*/*P'* (node *Professor* in *K'* is semantically similar to *Lecturer* and *PhD Student* in tree *N'* while it is relatively different from nodes *Supervisor* and *Worker* in tree *P'*). Hence, we identify the need to detect, not only the occurrences of identical leaf nodes (as discussed in the previous section), but also the occurrences of leaf nodes baring semantically similar labels. Detecting such similarities would obviously amend comparison accuracy.

4. Proposal

We view the problem of XML document structure comparison as that of detecting the occurrences and repetitions of structurally/semantically similar sub-trees. In sub-trees, we underline structures made of multiple nodes, as well as single leaf nodes. Thus, we aim to provide a unified and fine-grained method to deal with both structural and semantic resemblances left addressed by existing comparison methods. Our XML comparison method consists of four main algorithms:

i. *Struct_CBS* for identifying the *Structural Commonality Between two Sub-trees*,
ii. *Sem_RBS* for quantifying the *Semantic Resemblance Between two Sub-trees*,
iii. TOC_{XDoc} for computing the *Tree edit distance Operations Costs*,
iv. TED_{XDoc} for computing the *Tree Edit Distance* between XML document trees.

In short, the *TOC* algorithm makes use of *Struct_CBS* and *Sem_RBS* to structurally and semantically compare all sub-trees in the XML documents being compared. The produced sub-tree similarity results are consequently exploited as edit operations costs (particularly tree insertion and tree deletion costs, which are central to detecting the occurrences and repetitions of similar sub-trees), in an adapted version of [55]'s main edit distance algorithm, which we identify as *TED* (cf. Fig. 15). Hence, the inputs to our XML comparison approach are as follows:

- The XML document trees to be compared,
- Parameter α enabling the user to assign more importance to the structural or semantic aspects of the XML documents being treated,
- A reference (weighted) semantic network $\overline{SN}$, for semantic similarity evaluation.

Consequently, the method outputs the similarity between the XML document trees being compared. Our method's overall architecture is depicted in Fig. 7.

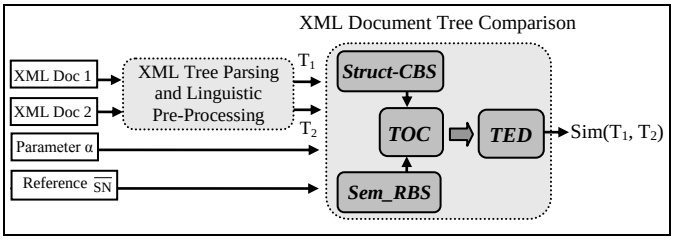


Fig. 7. Simplified activity diagram of our XML similarity approach

Note that the introduction of two separate algorithms: *Struct_CBS* to evaluate structure, and *Sem_RBS* to evaluate semantics (instead of one single hybrid algorithm), is a design choice to: i) emphasize the modularity of our approach (allowing to easily integrate additional algorithms in the future, considering other XML-related information, such as element/attribute values and/or hyperlinks), and ii) enable the user to easily parameterize the similarity measure (assigning more importance to either structure or semantics) following her notion of similarity. In addition, note that *Struct_CBS* and *Sem_RBS* can be applied to whole trees. However, in our study, their use is coupled with sub-trees so as to capture the various kinds of sub-tree similarities.

In the remainder of this section, we detail each of the algorithms and processes mentioned above. Section 4.6

formally defines our XML document similarity measure, and evaluates its properties w.r.t. the formal definition of similarity. Consequently, time and space complexity analyses are discussed in Section 4.7.

4.1. Preliminaries

As described in Section 2.1, XML documents represent hierarchically structured information and can be modeled as Ordered Labeled Trees (OLTs) [85]. Recall that in our study, an XML document is represented as an OLT with a node corresponding to each XML element and attribute. Attribute nodes appear as children of their encompassing element nodes, sorted by attribute name, and appearing before all sub-element siblings. As mentioned previously, we disregard element/attribute values while studying the structural properties of heterogeneous XML documents (*structure-only* XML comparison).

Definition 5 – Ordered Labeled Tree: It is a rooted tree in which the nodes are labeled and ordered. We denote by *T[i]* the i^{th} node of *T* in preorder traversal, *T[i].ℓ* its label, and *T[i].d* its depth. *R*(*T*)=*T[0]* designates the root node of tree *T* ●

Definition 6 – Sub-tree: Given two trees *T* and *T'*, *T'* is a sub-tree of *T* if all nodes of *T'* occur in *T*, with the same parent/child edge relationship and node order, such as no additional nodes occur in the embedding of *T'* (e.g., F_1 in Fig. 4 is a sub-tree of *F*, whereas tree *I* does not qualify as a sub-tree of *F* since node *e* occurs in its embedding in *F*) ●

Definition 7 – First level sub-tree: Given a tree *T* with root *p* of degree *k*, the first level sub-trees, $T_1, \ldots, T_k$ of *T* are the sub-trees rooted at the children nodes of $p, p_1, \ldots, p_k$ ●

Definition 8 – Ld-pair representation of a node: It is defined, as the pair *(ℓ, d)* where *ℓ* and *d* are respectively the node's label and depth in the tree. We use *p.ℓ* and *p.d* to refer to the label and the depth of an *ld-pair* node *p* respectively ●

Definition 9 – Ld-pair representation of a tree: It is the list, in preorder, of the *ld-pairs* of its nodes (cf. Fig. 8). Given a tree in *ld-pair* representation $T = (t_1, t_2, \ldots, t_n)$, *T[i]* refers to the i^{th} node t_i of *T*. Thus, *T[i].ℓ* and *T[i].d* denote, respectively, the label and the depth of the i^{th} node of *T*, *i* designating the preorder traversal rank of node *T[i]* in *T* ●

Note that the *ld-pair* tree representation was introduced by Chawathe in [12], and will be exploited in our study in comparing XML sub-trees (cf. Section 4.2, *Struct-CBS*).

A_1 = ((b, 0), (c, 1), (d, 1)) A_{11} = (c, 0) A_{12} = (d, 1)	B_2 = ((b, 0), (c, 1), (d, 1)) B_{21} = (c, 0) B_{22} = (d, 0) B_1 = ((b, 0), (c, 1), (d, 1)) B_{11} = (c, 0) B_{12} = (d, 0)	C_1 = ((b, 0), (c, 1), (d, 1)) C_{11} = (c, 0) C_{12} = (d, 0) C_2 = ((e, 0), (f, 1), (g, 1)) C_{21} = (f, 0) C_{22} = (g, 0)

Fig. 8. *Ld-pair* representations of all sub-trees in XML trees *A*, *B*, and *C* of Fig. 4.

In the following, we present the definitions of the tree edit operations utilized in our approach (adapted from [12, 55]).

Definition 10 – Atomic node operations: An atomic operation is an edit operation applied on a single tree node. Our approach exploits three atomic operations: leaf node insertion (introducing a new leaf node in the tree), leaf node deletion (removing a leaf node from the tree), and node update (modifying the label on an existing tree node):

- *Insert leaf node:* Let *p* be a node in a tree *T*, and let $T_1, \ldots, T_m$ be the first level sub-trees of *p*. Given a node *x* not belonging to *T*, *Ins*(*x*, *i*, *p*, *ℓ*) is a node insertion applied to *T*, inserting *x* as the i^{th} child of *p*, yielding *T'* with first level sub-trees $T_1, \ldots, T_{i-1}, x, T_{i+1}, \ldots, T_{m+1}$, where *ℓ* is the label of *x*.
- *Delete leaf node:* Given a leaf node *x* in a tree *T*, *Del*(*x*) is a node deletion operation applied to *T* that removes *x* from *T*, yielding tree *T'* with first level sub-trees $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_m$.
- *Update node:* Given a node *x* in tree *T*, and a label *ℓ*, *Upd*(*x*, *ℓ*) is a node update operation applied to *x* resulting in *T'* which is identical to *T* except that in *T'*, *x* bears *ℓ* as its label. The update operation could be also formulated as follows: *Upd*(*x*, *y*) where *y.ℓ* denotes the new label to be assumed by *x* ●

Note that the update operation in our approach targets nodes of identical structural positions, i.e., nodes having identical depth and ordering in the trees being compared, transforming the label of one node into that of the other.

Definition 11 – Complex tree operations: A complex tree edit operation is an edit operation applied on a sub-tree of nodes. Our approach exploits two complex operations: tree insertion and tree deletion:

- *Insert Tree :* Given a tree *A* and a tree *T* with an inner node *p* having first level sub-trees $T_1, T_2, \ldots, T_m$, *InsTree*(*A*, *i*, *p*) is a tree insertion applied to *T*, inserting *A* as the i^{th} sub-tree of *p*, thus yielding *T'* with first level sub-trees $T_1, \ldots, T_{i-1}, A, T_{i+1}, \ldots, T_{m+1}$.
- *Delete Tree :* Given a tree *A* and a tree *T* with an inner node *p*, *A* being the i^{th} sub-tree of *p*, *DelTree(A, p)* is a tree deletion operation applied to *T* that yields *T'* with first level sub-trees $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_m$ ●

In addition, we provide the formal definition of a semantic network, adopted in our study.

Definition 12 – Semantic network: It can be formally represented as a 3-tuple *SN=(C, E, R, f)* where:

- *C*: set of concepts, synonym sets as in WordNet [48].
- *E*: set of edges connecting the concepts, $E \subseteq V_C \times V_C$.
- *R*: set of semantic relations, *R* = {*Is-A*, *Has-A*, *Part-Of*, *Has-Part*…}, the synonymous words/expressions being integrated in the concepts themselves.
- *f* : function designating the nature of edges, $f: E \rightarrow R$.

We designate by $\overline{SN}$ a weighted semantic network, i.e., a semantic network *SN* augmented with concept frequencies (cf. Fig. 3), based on a given text corpus (e.g., the Brown Corpus of American English [25]) ●

Note that XML element/attribute tag names generally consist of single words, simple concatenations of words (usually not more than two terms per label [79], using the underscore delimiter or Java-style upper/lower case letters to distinguish the individual terms), and/or word abbreviations [59, 79]. Nonetheless, semantically meaningful XML labels are usually obtained after several linguistic pre-processing operations such as *tokenization* (parsing names into tokens based on punctuation and case, to form simple expressions, e.g., *PhD_Std* → *PhD Student*), *expansion* (identifying abbreviations and acronyms, e.g., *CEO* → *Chief Executive Officer*) and *stemming* (reducing inflected or derived words to their stem, i.e., base or root, e.g., *housing*, *housed* → *house*) [17, 43]. In the case of polysemous words (i.e., words with multiple senses), *word sense disambiguation* techniques, e.g. [54, 56, 79], can be exploited in order to select the semantic concept that most likely describes the meaning of the label in the given XML document. Note that linguistic pre-processing operations are executed offline [79], using dedicated thesauri and/or dictionaries (in our case, WordNet), and do not affect the performance of our comparison approach (Fig. 7).

4.2. Structural Similarity between Sub-trees (*Struct-CBS*)

As shown in Section 3.1, sub-tree structural similarities are usually left undetected in current XML comparison approaches. In [55], the authors were the first to address the issue and were able to detect certain basic sub-tree structural similarities using tree insertion and tree deletion operations, coupled with the tree *contained in* relation (cf. Section 2.2.2). Nonetheless, when the containment relation is not fulfilled, various structural similarities are ignored, as discussed in the motivation examples.

Here, in order to capture the various kinds of sub-tree structural similarities pinpointed in Section 3.1, we identify the need to replace the tree *contained in* relation, making up a necessary condition for executing tree insertion and deletion operations in [55], by introducing the notion of *structural commonality* between two sub-trees.

Definition 13 – Structural commonality between sub-trees: Given two sub-trees $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$, we define the structural commonality between *A* and *B*, designated by *StructCom*(*A*, *B*), as the set of pairs of nodes *N* form *A* and *B*, $N=\{(a_r, b_u)\} \in A \times B$, such that $\forall\, a_r \in A, b_u \in B$, a_r and b_u occur in *A* and *B* respectively, with the same label, depth and relative order (in preorder traversal). For $1 \le r \le m$ and $1 \le u \le n$:

(1) $a_r.\ell = b_u.\ell$

(2) $a_r.d = b_u.d$

(3) For any $(a_s, b_v) \in N$ such as $r \le s$, then $u \le v$ ●

Following Definition 13, the problem of finding the structural commonality between two sub-trees SbT_i and SbT_j is equivalent to finding the maximum number of structurally matching nodes in SbT_i and SbT_j ($|StructCom(SbT_i, SbT_j)|$). However, the problem of finding the edit distance between SbT_i and SbT_j comes down to identifying the minimal number of edit operations that can transform SbT_i to SbT_j. Those are dual problems since identifying the edit distance between two sub-trees (trees) underscores, in a roundabout way, their maximum number of matching nodes. In other words, the greater the edit distance, the larger the edit script, the greater the number of edit operations, the greater the number of node transformations, the lesser the number of matching nodes.

Therefore, we introduce in Fig. 9 the pseudo-code of our *Struct_CBS* algorithm, based on the edit distance concept, to identify the structural commonality between sub-trees (similarly to the approach provided in [53], in which the author develops an edit distance based approach for computing the longest common sub-sequence between two strings). In *Struct_CBS*, sub-trees are treated in their *ld-pair* representations (cf. Definition 9, Fig. 8). Using the *ld-pair* tree representations, sub-trees are transformed into *modified* sequences (*ld-pairs*), making them suitable for *standard* edit distance computations. The algorithm starts by computing the sum of the costs of deleting every node in the source sub-tree (Fig. 9, line 3), and inserting every node of the destination tree (line 4). Consequently, it identifies the set of insertion/deletion operations having the minimum overall cost (lines 5-15). Structurally matching nodes are associated null costs (line 10). Note that the update operation is specifically disregarded in *Struct-CBS*, in order to allow the identification of structurally matching nodes (line 10). Consequently, the overall sum of the minimum operations' costs (i.e., minimum cost edit script, cf. Definition 2) underlines an edit distance, i.e., $Dist[|SBT_i|][|SbT_j|]$, between the sub-trees SbT_i and SbT_j being compared. Hence, the maximum number of matching nodes between SbT_i and SbT_j, $|StructCom\ (SbT_i, SbT_j)|$, is identified w.r.t. the edit distance score:

- Total number of deletions: we delete all nodes of SbT_i except those having matching nodes in SbT_j, $\sum_{\text{Deletions}} = |SbT_i| - |StructCom(SbT_i, SbT_j)|$
- Total number of insertions: we insert into SbT_i all nodes of SbT_j except those having matching nodes in SbT_i, $\sum_{\text{Insertions}} = |SbT_j| - |StructCom(SbT_i, SbT_j)|$
- Following *Struct_CBS*, using constant unit costs (=1) for node insertion and deletion operations, the edit distance between sub-trees SbT_i and SbT_j becomes as follows:

$$Dist[|SbT_i|][|SbT_j|] = \sum_{\text{Deletions}} \times 1 + \sum_{\text{Insertions}} \times 1 = |SbT_i| + |SbT_j| - 2 \times |StructCom(SbT_i, SbT_j)|$$

Therefore:

$$|StructCom(SbT_i, SbT_j)| = \frac{|SbT_i| + |SbT_j| - Dist[|SbT_i|][|SbT_j|]}{2} \quad \textbf{(4)}$$

Algorithm *Struct_CBS*

Input: SbT_i and SbT_j // Sub-trees in ld-pair representations
Output: Struct_CBS(SbT_i, SbT_j) // Normalized structural commonality

```
Begin
  Dist[][] = new [0...|SbT_i|][0…|SbT_j|]                                          1
  Dist[0] = 0                                                                      2
  For (n = 1 ; n ≤ |SbT_i| ; n++)   {Dist[n][0] = Dist[n-1][0] + Cost_Del(SbT_i[n])}   3
  For (m = 1 ; m ≤ |SbT_j| ; m++) {Dist[0][m] = Dist[0][m-1] + Cost_ins(SbT_j[m])}   4
  For (n = 1 ; n ≤ |SbT_i| ; n++)                                                  5
  {                                                                                6
    For (m = 1 ; m ≤ |SbT_j| ; m++)                                                7
    {                                                                              8
      Dist[n][m] = Min{                                                            9
        If (SbT_i[n].d = SbT_j[m].d & SbT_i[n].ℓ = SbT_j[m].ℓ) { Dist[n-1][m-1] }, 10
        Dist[n-1][m] + Cost_Del(SbT_i[n]),      // Simplified node                  11
        Dist[n][m-1] + Cost_ins(SbT_j[m])       // operations syntaxes.             12
        }                                                                          13
    }                                                                              14
  }                                                                                15
Return (|SbT_i| + |SbT_j| − Dist[|SbT_i|][|SbT_j|]) / (2×Max(|SbT_i|,|SbT_j|))   // Normalized commonality  16
End
```

Fig. 9. Algorithm *Struct_CBS* for identifying the structural commonality between sub-trees.

To obtain structural commonality values comprised in the *[0, 1]* interval, we normalize $|StructCom(SbT_i, SbT_j)|$ via corresponding sub-tree cardinalities, $Max(|SbT_i|, |SbT_j|)$. Thus:

$$\frac{|StructCom(SbT_i, SbT_j)|}{Max(|SbT_i|, |SbT_j|)} = 0$$ When there is no structural commonality: $|StructCom(SbT_i, SbT_j)| = 0$

$$\frac{|StructCom(SbT_i, SbT_j)|}{Max(|SbT_i|, |SbT_j|)} = 1$$ When sub-trees are identical: $|StructCom(SbT_i, SbT_j)| = |SbT_i| = |SbT_j|$

Table 1 shows the detailed computations and results of applying *Struct_CBS* to sample sub-trees A_1 and D_1 of Fig. 4 (reported in Fig. 10).

Table 1. Detailed computations, following *Struct_CBS*, when applied on sub-trees A_1 and D_1.

	0	D_1[1] (b, 0)	D_1[2] (c, 1)	D_1[3] (d, 1)	D_1[4] (h, 1)
0	0	1	2	3	4
A_1[1] (b, 0)	1	0	1	2	3
A_1[2] (c, 1)	2	1	0	1	2
A_1[3] (d, 1)	3	2	1	0	***1***

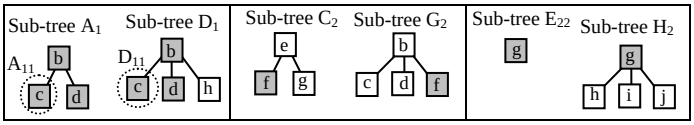


Fig. 10. Sample sub-trees bearing structural commonalities.

The first line of the distance matrix, i.e., *Dist[0][]*, corresponds to the sum of the costs of inserting every node of the destination sub-tree D_1. Likewise, the first column, *Dist[][0]*, underlines the sum of the costs of deleting every node of A_1. Consequently, the algorithm identifies the combination of insertion/deletion operations of minimum overall cost (cf. Fig. 9, lines 7-18) in populating the remainder of the matrix, $Dist[|A_1|][|D_1|]$ underlining the final distance value. Note that in Table 1, matching nodes are highlighted, while the (final) distance value is emphasized in *italic* format. Having $Dist[|A_1|][|D_2|] = 1$:

$$|StructCom(A_1, D_2)| = \frac{|A_1|+|D_2| - Dist[|A_1|][|D_2|]}{2} = \frac{3+4-1}{2} = 3,$$

(nodes *b*, *c*, *d*). Consequently, $Struct\_CBS(A_1, D_1) = \frac{3}{4} = 0.75$.

Similarly, $Struct\_CBS(C_2, G_2) = \frac{|StructCom(C_2, G_2)|}{Max(|SbT_i|, |SbT_j|)} = \frac{1}{4} = 0.25$

(having $|StructCom(C_2, G_2)|=1$).

Note that applying *Struct_CBS* to leaf node sub-trees comes down to comparing corresponding sub-tree root node labels (leaf node sub-trees consisting of sub-trees made of single nodes: the sub-tree root nodes themselves, bearing identical (=0) depth and ordering scores). For instance, $Struct\_CBS(A_{11}, D_{11}) = 1$, since sub-trees A_{11} and D_{11} consist of leaf nodes of label *c*. Similarly, when computing the commonality between a leaf node sub-tree (e.g., A_{11}) and a non-leaf node sub-tree (e.g., D_1), *Struct_CBS* compares the label of the root of the former ($R(A_{11})$, the leaf node itself) to that of the latter ($R(D_1)$):

- $Struct\_CBS(A_{11}, D_1) = 0$, roots of A_{11} (leaf node) and D_1 having different labels,
- $Struct\_CBS(E_{22}, H_2) = 1/4 = 0.25$ having $|StructCom(E_{22}, H_2)| = 1$ (since $R(E_{22}) = R(H_2) = g$) and $Max(|E_{22}|, |H_2|) = 4$.

4.3. Semantic Resemblance between Sub-trees (*Sem-RBS*)

In addition to sub-tree structural commonalities (i.e., considering parent/child relationships and ordering among XML elements, identified by their labels), we aim to consider sub-tree semantics in XML similarity evaluation (i.e., semantic meaning of sub-tree node labels). For the sake of clearness, we use expression '*semantic resemblance*', in the remainder of the paper, to avoid confusion between semantic and structural similarity, the latter designated as '*structural commonality*'.

Various methods for detecting the semantic similarity between pairs of words/expressions, based on a given reference semantic network, have been proposed (cf. Section 2.3). Nonetheless, capturing the semantic relatedness between two sets of words/expressions (e.g., node labels of two sub-trees) has not been effectively covered in the literature. To our knowledge, two complementary approaches have tackled the issue, i.e., [15, 24], developed in the context of concept similarity of ontology management systems [24], and concept similarity in geographic information systems [15]. While theoretically sound, the solution provided in [15, 24] does not seem practical, since it requires a minimum of $O(N!)$ time (a detailed mathematical analysis is provided in [75]).
Hence, to capture the semantic resemblance between two sub-trees, we provide a new approach entitled *Sem_RBS*, that combines the traditional vector space model in information retrieval [47], with semantic similarity evaluation (cf. Section 2.3). In detail, we proceed as follows. When comparing two sub-trees SbT_i and SbT_j, each would be represented as a vector

$\vec{V}$ ($\vec{V_i}$ and $\vec{V_j}$ respectively) with weights underlining the semantic similarities between their corresponding node labels. Recall, that XML tag names undergo several linguistic pre-processing operations (including *tokenization*, *expansion*, *stemming*, and *word sense disambiguation*, cf. Section 4.1) so as to obtain semantically meaningful labels prior to the comparison process.

Definition 14 – Sub-tree Vectors: Given two sub-trees SbT_i and SbT_j, we define corresponding sub-tree vectors $\vec{V_i}$ and $\vec{V_j}$ in a space which dimensions represent, each, a single node label $\ell_r \in SbT_i \cup SbT_j$, such as $1 < r < n$ where n is the number of distinct node labels in both SbT_i and SbT_j. The coordinate of a sub-tree vector $\vec{V_i}$ on dimension ℓ_r is noted $w_{\vec{V_i}}(\ell_r)$, underlining the semantic weight of label ℓ_r in SbT_i ●

Definition 15 – Semantic Sub-tree Node Weight: The semantic weight of a node v_r in vector $\vec{V_i}$, representing sub-tree SbT_i, is composed of two factors: a node/vector similarity factor $Sim(v_r, \vec{V_i})$ and a depth *D-factor*(v_r) factor:

$$w_{\vec{V_i}}(v_r) = \text{Sim}(v_r, \vec{V_i}) \times \text{D-factor}(v_r) \quad \in [0, 1] \qquad \textbf{(5)}$$

– $Sim(v_r, \vec{V_i})$ quantifies the semantic similarity between the label $v_r.\ell$ of node v_r and sub-tree vector $\vec{V_i}$. It is computed as the maximum semantic similarity between label $v_r.\ell$ and all node labels of SbT_i w.r.t. a reference (weighted) semantic network $\overline{\text{SN}}$ (cf. Definition 12). Formally:

$$\text{Sim}(v_r, \vec{V_i}) = \max_{v \in Vi} (\text{Sim}_{Label}(v_r.\ell, v.\ell, \overline{\text{SN}})) \quad \in [0,1] \qquad \textbf{(6)}$$

When $v_r \in SbT_i$, $Sim(v_r, \vec{V_i}) = 1$ underlines the node's occurrence in SbT_i.

– *D-factor* underlines the semantic influence of node depth on XML semantic similarity. It follows the intuition that information placed near the root node of an XML document is more important than information further down in the hierarchy [6, 90]. Thus, node labels higher in the XML tree hierarchy should have a greater semantic influence than their lower counterparts. This could be mathematically concretized using *Formula (7)*, adapted from [90]:

$$D\text{-}factor(v_r) = \frac{1}{1 + v_r.d} \quad \in [0,1] \qquad \textbf{(7)}$$

where $v_r.d$ underlines the depth of node v_r in the document ●

As for the label semantic similarity measure, Sim_{Label}, our investigation of the literature (Section 2.3) led us to consider Lin's method [41] in our XML comparison process (i.e., $Sim_{Label} \equiv Sim_{Lin}$). Lin's measure was proven efficient in evaluating semantic similarity, in comparison with its predecessors, i.e., [60, 87]. Its performance and theoretical basis are recognized and generalized by [44] to deal with hierarchical and non-hierarchical structures. However, it is important to note that our XML similarity approach is not sensitive, in its definition, to the semantic similarity measure used. Yet, choosing a performing measure would yield better similarity judgment.

Having transformed XML sub-trees into semantically weighted vectors, the semantic relatedness between two sub-trees can be evaluated using a measure of similarity between vectors such as the inner product, the cosine measure, the Jaccard measure, etc. Here, we adopt the cosine measure widely exploited in information retrieval [8, 63]:

$$Sem\text{-}RBS(SbT_i, SbT_j) = \text{Cos}(\vec{V_i}, \vec{V_j}) \quad \in [0,1] \qquad \textbf{(8)}$$

where $\vec{V_i}$ and $\vec{V_j}$ are the semantically weighted vectors corresponding to SbT_i and SbT_j respectively.

Algorithm *Sem_RBS* consists in building the vector space corresponding to the sub-trees being compared, as well as computing the semantic and cosine measures as explained above. It takes as input the sub-trees SbT_i and SbT_j to be compared, and the reference semantic network $\overline{\text{SN}}$, and generates the sub-tree semantic similarity score ($\in [0, 1]$). *Sem_RBS*'s pseudo-code is a straightforward consequence of Definition 14 and Definition 15, and is thus omitted for clearness of presentation (it can be found in [75]).

Sample computation examples when comparing sub-trees A'_1/B'_2 and A'_1/C'_2 of Fig. 11 (reported from Fig. 6) are shown hereunder.

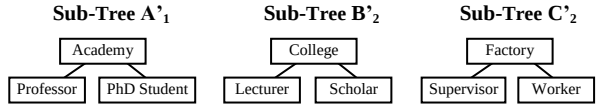


Fig. 11. Sample sub-trees bearing semantic resemblances.

When comparing A'_1 and B'_2, the corresponding vector space consists of 6 dimensions corresponding to each distinct node label in both sub-trees: *Academy*, *Professor*, *PhD Student*, *College*, *Lecture* and *Scholar*. Thus, 6-dimensional vectors $V_{A'1}$ and $V_{B'2}$ are produced:

	Academy	Professor	PhD Student	College	Lecturer	Scholar
$V_{A'1}$	1	1	1	0.7970	0.7674	0.8402
$V_{B'2}$	0.7970	0.7674	0.8402	1	1	1

a. Node label semantic similarity values.

	Academy	Professor	PhD Student	College	Lecturer	Scholar
$V_{A'1}$	1	0.5	0.5	0.7970	0.3837	0.4201
$V_{B'2}$	0.7970	0.3837	0.4201	1	0.5	0.5

b. Final weights, i.e., *Sim* × *D-factor*.

Fig. 12. Sub-tree vectors when comparing sub-trees A'_1 and B'_2.

Semantic similarity values are computed following Lin's semantic similarity measure [41] (cf. *Formula (3)*). Here, in computing label semantic similarities, we exploit the weighted semantic network in Fig. 3. Similarity values, following [41], between pairs *Academy*/*College*, *Professor*/*Lecturer*, and *PhD Student*/ *Scholar* are computed as follows:

$$\text{Sim}_{Lin}(Academy, College) = \frac{2 \log p(Establishment)}{\log p(Academy) + \log p(College)}$$

$$= \frac{2 \log(\frac{17}{260})}{\log(\frac{8}{260}) + \log(\frac{9}{260})} = 0.7970$$

Likewise, $\mathrm{Sim}_{Lin}(Professor, Lecturer) = 0.7674$ and $\mathrm{Sim}_{Lin}(PhD\ Student, Scholar) = 0.8402$.

Recall that the semantic weight of a given node v_r, of label ℓ_r, in vector $\overrightarrow{V_i}$, is computed as the maximum semantic similarity between ℓ_r and all node labels of $\overrightarrow{V_i}$ (cf. Definition 15). In our example, $Sim_{Lin}(Academy, College)$ underlines the maximum similarity value between label *Academy* and all labels of vector $\overrightarrow{V_{B'2}}$, and vice-versa for *College* and $\overrightarrow{V_{A'1}}$. The same is true for node labels *Professor* and *Lecturer*, as well as *PhD Student* and *Scholar*, w.r.t. $\overrightarrow{V_{B'2}}$ and $\overrightarrow{V_{A'1}}$ respectively (Fig. 12.b). Thus, final vector weights are obtained by multiplying both semantic similarity and depth factors $Sim_{Label} \times D\text{-}factor$ as shown in Fig. 12.c (Definition 15). Hence, the semantic resemblance between sub-trees A'_1 and B'_2, w.r.t. the reference semantic network $\overline{SN}$ in Fig. 3:

$$Sem\text{-}RBS(A'_1, B'_2) = \mathrm{Cos}(\overrightarrow{V_{A'1}}, \overrightarrow{V_{B'2}}) = 0.9754.$$

Similarity, when comparing sub-trees A'_1 and C'_2, the corresponding vector space consists of 6 dimensions corresponding to each distinct node label in both sub-trees:

	Academy	Professor	PhD Student	Factory	Supervisor	Worker
$V_{A'1}$	1	1	1	0.2662	0.3608	0.3608
$V_{C'2}$	0.2662	0.3608	0.3608	1	1	1

a. Node label semantic similarity values.

	Academy	Professor	PhD Student	Factory	Supervisor	Worker
$V_{A'1}$	1	0.5	0.5	0.2662	0.1804	0.1804
$V_{C'2}$	0.2662	0.1804	0.1804	1	0.5	0.5

b. Semantic weights, i.e., *SN-factor* × *D-factor*.

Fig. 13. Sub-tree vectors when comparing sub-trees A'_1 and C'_2.

Semantic similarities between pairs of labels are computed following Lin [41] (Fig. 13.a):

$$\mathrm{Sim}_{Lin}(Academy, Factory) = \frac{2 \log p(Structure)}{\log p(Academy) + \log p(Factory)} = 0.2662$$

Likewise, $\mathrm{Sim}_{Lin}(Professor, Supervisor) = 0.3608$ and $\mathrm{Sim}_{Lin}(PhD\ Student, Worker) = 0.3608$.

Consequently, semantic resemblance: $Sem\text{-}RBS(A'_1, C'_2) = \mathrm{Cos}(\overrightarrow{V_{A'1}}, \overrightarrow{V_{C'2}}) = 0.5303$.

Results show that sub-tree A'_1 (made of node labels *Academy*, *Professor* and *PhD Student*) is semantically more similar to sub-tree B'_2 (*College*, *Lecturer*, *Scholar*) than C'_2 (*Factory*, *Supervisor*, *Worker*).

4.4. Tree Edit Operations Costs (*TOC*)

As stated previously, *TOC* (Fig. 14) is an algorithm dedicated to computing tree edit distance operations costs, particularly the costs of tree insertion and tree deletion operations (cf. Definition 11, including single node insertions/deletions costs), which are central to detecting sub-tree similarities when comparing two XML document trees (note that the use and cost of the update operation, cf. Definition 10, are discussed in the following section). *TOC* combines the structural commonalities (*Struct_CBS*) and semantic resemblances (*Sem_RBS*) between each pair of sub-trees (SbT_i and SbT_j) in the source and destination XML trees (*A* and *B*) respectively, assigning tree insert/delete operations costs accordingly. Consequently, these costs are exploited via an adaptation of Nierman and Jagadish's main edit distance algorithm [55] (Fig. 15) providing an improved and more accurate XML document similarity measure.

Following *TOC*, the similarity between two XML sub-trees, $SS(SbT_i, SbT_j)$, is evaluated as the weighted average of their structural commonality (*Struct_CBS*) and semantic resemblance (*Sem_RBS*) scores:

$$SS(SbT_i, SbT_j, \alpha) = \alpha \times Struct\_CBS(\mathrm{SbT_i}, \mathrm{SbT_j}) + (1 - \alpha) \times Sem\_RBS(\mathrm{SbT_i}, \mathrm{SbT_j}) \quad \textbf{(9)}$$

where $\alpha \in [0, 1]$ is provided as input.

The user can thus assign more importance to either structural or semantic similarities by varying parameter $\alpha \in [0, 1]$:

- For $\alpha=1$, *TOC* will only consider structural commonalities in computing operations costs (via *Struct_CBS*).
- For $\alpha = 0$, only sub-tree semantic resemblances will be considered in computing operations costs (via *Sem_RBS*).

The fine-tuning of parameter α so as to effectively combine sub-tree structure similarity (*Struct-CBS*) and semantic similarity (*Sem-RBS*) comes down to an optimization problem such as α should be chosen to maximize the overall sub-tree similarity function (cf. *Formula (*9)). This can be solved using a number of known techniques that apply machine learning in order to identify the best weights for a given problem class [20, 22, 32, 45, 49]. The main idea with this family of techniques is to assign a higher (lower) weight with higher (lower) similarity values, acting like contrast filters in image processing by increasing the contrast on input matrixes. Providing such a capability, in addition to manual tuning, would enable the user to parameterize and adapt the XML comparison process following the scenario at hand, giving more emphasis to the structural or (inclusive) semantic aspects of the XML documents being compared. We do not further address the fine-tuning of parameter α here since it is out of the scope of this paper (and will be addressed in an upcoming technical study).

Thus, following *TOC*, tree operations costs vary as follows:

$$\mathrm{Cost}_{\mathrm{InsTree/DelTree}}(\mathrm{SbT_i}) = \sum_{\text{All nodes } x \text{ of } \mathrm{SbT_i}} \mathrm{Cost}_{\mathrm{Del}}(x) \times \frac{1}{1 + \underset{\text{all } \mathrm{SbT_j} \in \mathrm{T_2}}{Max} \{SS(\mathrm{SbT_i}, \mathrm{SbT_j}, \alpha)\}}$$

Maximum insert/delete sub-tree cost:

$$\text{- } \mathrm{Cost}_{\mathrm{InsTree/DelTree}}(\mathrm{SbT_i}) = \sum_{\text{All nodes } x \text{ of } \mathrm{SbT_i}} \mathrm{Cost}_{\mathrm{Ins/Del}}(x) \times 1$$

Minimum insert/delete sub-tree cost: **(10)**

$$- \text{Cost}_{\text{InsTree/DelTree}}(\text{SbT}_i) = \sum_{\text{All nodes } x \text{ of SbT}_i} \text{Cost}_{\text{Ins/Del}}(x) \times \frac{1}{2}$$

Lemma 1. Following *TOC*, the maximal insert/delete tree operation cost for a given sub-tree SbT_i (attained when no sub-tree *structural commonalities* nor *semantic resemblances* with SbT_i are identified in the source/destination tree) is the sum of the costs (unit costs, =1)[1] of inserting/deleting every individual node of SbT_i (the proof is evident) ●

Lemma 2. Following *TOC*, the minimal insert/delete tree operation cost for SbT_i (attained when a sub-tree identical to SbT_i is identified in the source/destination tree respectively) is equal to half its insert/delete tree maximum cost ●

The minimal tree operation cost is defined in such a way in order to guarantee that the cost of inserting/deleting a non-leaf node sub-tree will never be less than the cost of inserting/deleting a single node (single node operations having unit costs). In fact, *TOC* is based on the intuition that tree operations are more costly than node operations.

Proof: *The smallest non-leaf node sub-tree that can be treated via a tree operation is a sub-tree consisting of two nodes. For such a tree, the minimum insert/delete tree operation cost would be equal to 1 (its maximum cost being equal to 2), equivalent to the cost of inserting/deleting a single node. That is the lowest tree operation cost attainable, for a non-leaf node sub-tree, following TOC* □

Hence, for leaf node sub-trees, the maximum insert/delete tree operation cost is equal to 1, the cost of inserting/deleting the single node at hand:

- $\text{Cost}_{\text{InsTree/DelTree}}(\text{SbT}_i) = \text{Cost}_{\text{Ins/Del}}(x) \times 1 = 1$, when SbT_i is made of single node *x*.

The minimum cost for inserting/deleting a single node sub-tree is equal to 0.5, half its maximum insert/delete cost:

- $\text{Cost}_{\text{InsTree/DelTree}}(\text{SbT}_i) = \text{Cost}_{\text{Ins/Del}}(x) \times 1/2 = 0.5$, SbT_i consisting of single node *x*.

This is essential in order to detect the similarities and repetitions among leaf node sub-trees (such as with the *K*, *L*, *M* and *K*, *N*, *P* comparison cases in Fig. 4, discussed in the motivation section).

[1] An intuitive and natural way has been usually used to assign single node operation costs and consists of considering identical unit costs for *insertion* and *deletion* operations [13, 55].

Algorithm *TOC*

Input: A , B // XML document trees
α // Parameter for weighting *Struct_CBS* and *Sem_RBS*
$\overline{\text{SN}}$ // Weighted semantic network

Output: $\{\text{Cost}_{\text{DelTree}}\} \cup \{\text{Cost}_{\text{InsTree}}\}$ // Delete tree and insert tree operations costs

Begin

1 For each sub-tree SbT_i in A (including A) // Traversing the sub-trees of A
2 {
3 $\text{Cost}_{\text{DelTree}}(\text{SbT}_i) = \sum_{\text{All nodes } x \text{ of SbT}_i} \text{Cost}_{\text{Del}}(x)$
4 For each sub-tree SbT_j in B (including B) // Traversing the sub-trees of B
5 {
6 $\text{Cost}_{\text{InsTree}}(\text{SbT}_j) = \sum_{\text{All nodes } x \text{ of SbT}_j} \text{Cost}_{\text{Ins}}(x)$
7 $\text{Cost}_{\text{DelTree}}(\text{SbT}_i) = \text{Min}\{ \text{Cost}_{\text{DelTree}}(\text{SbT}_i), \sum_{\text{All nodes } x \text{ of SbT}_i} \text{Cost}_{\text{Del}}(x) \times \frac{1}{1+\text{SS}(\text{SbT}_i, \text{SbT}_j, \alpha)} \}$
8 $\text{Cost}_{\text{InsTree}}(\text{SbT}_j) = \text{Min}\{ \text{Cost}_{\text{InsTree}}(\text{SbT}_j), \sum_{\text{All nodes } x \text{ of SbT}_j} \text{Cost}_{\text{Ins}}(x) \times \frac{1}{1+\text{SS}(\text{SbT}_i, \text{SbT}_j, \alpha)} \}$
9 }
10 }
11 Return $\{\text{Cost}_{\text{InsTree}}\} \cup \{\text{Cost}_{\text{DelTree}}\}$

End

Fig. 14. Tree edit distance Operations Costs (*TOC*) algorithm.

On one hand, note that in our approach, single node insertions/deletions are undertaken via tree insert/delete operations (cf. Definition 11) applied on leaf node sub-trees. Insert/delete node operations (cf. Definition 10), which are assigned unit costs as with traditional edit distance approaches, are only utilized to compute tree insertion/deletion operations costs (cf. *Struct_CBS* in Fig. 9, and *TOC* in Fig. 14 - lines 3 and 6). They do not however contribute to the dynamic programming procedure adopted in our edit distance approach (similarly to [16, 55], cf. *TED* algorithm in Fig. 15).

On the other hand, algorithm *TOC* exploits tree insertion/deletion operations to identify not only the structural/semantic similarities between sub-trees (SbT_i, SbT_j) but also the similarities between the sub-trees and the whole XML trees (*A* and *B*) being compared (cf. Fig. 14, lines 1 and 4). This is necessary when one of the trees involved in the comparison process shares structural/semantic similarities with one (or more) of the sub-trees encompassed in the other XML document tree (e.g., the *F*, *I*, *J* case in Fig. 4 where tree *I* is structurally similar to sub-tree F_1, and the *A'*, *I'*, *J'* case in Fig. 6 where tree *I'* is semantically similar to sub-tree A'_1). Nonetheless, note that inserting/deleting the whole destination/ source trees is not allowed in our approach (cf. algorithm *TED* in Fig. 15). In fact, by allowing such operations, one could delete the entire source tree in one step and insert the entire destination tree in a second step, which completely undermines the purpose of the insert/delete tree operations.

To sum up, *TOC* computes the costs of tree insertion and deletion operations based on their corresponding sub-trees' *structural commonality* and *semantic resemblance* values (maximum values inducing minimum tree operations costs), to be exploited in the main tree edit distance algorithm (*TED*).

4.5. Tree Edit Distance (*TED*)

The pseudo-code of the tree edit distance algorithm *TED*, utilized in our study, is developed in Fig. 15. It is an adaptation of Nierman and Jagadish's main edit distance process [55]. In addition to tree insertion/deletion operations costs which vary w.r.t. the structural/semantic similarities between XML sub-trees, *TED* exploits update operations costs (Fig. 15, line 4) in computing the distance between two XML document trees. In short, the algorithm recursively goes through the sub-trees of both XML document trees being compared, combining node update, tree insertion and tree deletion operations so as to identify those of minimal cost. The node update operation (Definition 10) is applied to the roots of the XML trees being compared, as well as the roots of each pair of sub-trees considered in the recursive process (Fig. 15, line 4), whereas tree insertion and tree deletion operations are applied to corresponding first-level sub-trees (Fig. 15, lines 5-6, 13-14). Recall that the insertion/deletion of single nodes are undertaken via tree insertion/deletion operations applied on leaf node sub-trees (as described in the previous section).

While tree insertion/deletion operations' costs allow detecting the structural and semantic similarities between XML sub-trees (cf. *TOC*), the update operation cost is central in evaluating the similarity between the roots of the XML document trees being compared, as well as the roots of XML sub-trees considered in the recursive process (*TED*).

With classical edit distance approaches, the cost of the update operation underlines the equality/difference between node labels:

- Minimum cost when the compared element labels are identical, $Cost_{Upd}(a, b) = 0$ when $a.\ell = b.\ell$
- Maximum unit cost otherwise, i.e. $Cost_{Upd}(a, b) = 1$ when $a.\ell \neq b.\ell$

Nonetheless, to consider the semantic similarities between element labels (not only label equality/difference) in our study, we extend the update operation cost scheme as follows:

$$Cost_{Udp}(a, b, \alpha) = \begin{cases} [1 - (1-\alpha) \times (Sim_{Label}(a.\ell, b.\ell, \overline{SN}))] \times \dfrac{\alpha + D\text{-}fact(a.d)}{1 + \alpha\ D\text{-}fact(a.d)} & if\ a.\ell \neq b.\ell \\ 0 \quad otherwise \end{cases} \quad \textbf{(11)}$$

where $\alpha \in [0, 1]$

Parameter α (which is the same utilized in *TOC*) allows assigning more importance to either structural or semantic similarities:

- For $\alpha = 1$, we only consider label equality/difference in computing the cost of the update operation, as with traditional structural edit distance approaches,
- For $\alpha = 0$, node semantic similarities will be considered in computing the update operation cost. Here, the operation cost varies in the $[0, 1]$ interval w.r.t. the semantic similarity between the concerned node labels and corresponding depths (note that nodes treated via the update operation are of the same depth, i.e., $a.d = b.d$).

Algorithm *TED*

Input: A and B // XML document trees to be compared
$\{Cost_{DelTree}\} \cup \{Cost_{InsTree}\}$ // Sub-tree deletion/insertion operations costs
α // Parameter for structural/semantic weighting,
$\overline{SN}$ // Weighted semantic network

Output: TED(A, B) // Edit distance between A and B

Begin

M = Degree(A) // The number of first level sub-trees in A. 1
N = Degree(B) // The number of first level sub-trees in B. 2

Dist [][] = new [0...M][0…N] 3
Dist[0][0] = $Cost_{Upd}$(R(A), R(B), α) // Update operation 4

For (i = 1 ; i ≤ M ; i++) { Dist[i][0] = Dist[i-1][0] + $Cost_{DelTree}(A_i)$ } 5
For (j = 1 ; j ≤ N ; j++) { Dist[0][j] = Dist[0][j-1] + $Cost_{InsTree}(B_j)$ } 6

For (i = 1 ; i ≤ M ; i++) 7
{ 8
For (j = 1 ; j ≤ N ; j++) 9
{ 10
Dist[i][j] = min{ 11
$TED_{XDoc}(A_i, B_j, \{Cost_{DelTree}\} \cup \{Cost_{InsTree}\}, \alpha, \overline{SN})$ 12
Dist[i-1][j] + $Cost_{DelTree}(A_i)$, 13
Dist[i][j-1] + $Cost_{InsTree}(B_j)$ 14
} 15
} 16
} 17
Return Dist[M][N] // ≡ *TED(A, B)* 18
End

Fig. 15. Tree edit distance algorithm (*TED*).

Consider for instance document trees *X*, *Y* and *Z* in Fig. 5. With $\alpha = 0$, the corresponding root update operations costs would be as follows:

- $Cost_{Upd}(R(X), R(Y)) = 1 - (Sim_{Lin}(Academy, College) \times 1) = 0.2030$
- $Cost_{Upd}(R(X), R(Z)) = 1 - (Sim_{Lin}(Academy, Factory) \times 1) = 0.7337$

It is clear that the cost of updating *Academy* and *College* is lesser than that of transforming *Academy* into *Factory*, identifying the fact that the former couple is more semantically similar than the latter (Detailed computation examples are developed in the Appendix).

Here, as with *Sem-RBS*, we exploit Lin's measure [41] to assess the semantic similarity between node labels (i.e., $Sim_{Label} \equiv Sim_{Lin}$). However, recall that its use is not mandatory. We use it since it is among the most efficient measures available, as discussed previously.

4.6. XML Document Similarity Measure (Sim_{XDoc})

In our study, we adopt the formal definition of similarity as the inverse of a distance function [21], i.e., tree edit distance. Given XML document trees *A*, *B* and *C*:

$$Sim_{XDoc}(A, B) = 1 - \frac{TED(A, B)}{|A| + |B|} \quad \textbf{(12)}$$

Note that $TED(A, B) \equiv TED(A, B, \{Cost_{InsTree}\} \cup \{Cost_{DelTree}\}, \alpha, \overline{SN})$, and likewise $Sim_{XDoc}(A, B) \equiv Sim_{XDoc}(A, B, \alpha, \overline{SN})$, following our algorithms. Yet, we omit the α, $\overline{SN}$ and $\{Cost_{InsTree}\} \cup \{Cost_{DelTree}\}$ input parameters in *Formula (*12) for ease of presentation.

Our similarity measure is consistent with the formal definition of similarity [21, 44], and comes down to a *generalized metric* – i.e., a similarity (distance) function satisfying all metric properties except for *triangular inequality*:

i. $Sim_{XDoc}(A, B) \in [0, 1]$.
ii. $Sim_{XDoc}(A, B) = 1 \Rightarrow A$ and B are identical.
iii. $Sim_{XDoc}(A, B) = 0 \Rightarrow A$ and B have no common characteristics[1],
iv. Similarity increases with the commonality between A and B, and decreases with their difference.
v. $Sim_{XDoc}(A, A) = 1 \Rightarrow$ similarity is reflexive.
vi. $Sim_{XDoc}(A, B) = Sim_{XDoc}(B, A) \Rightarrow$ similarity is symmetric.

In fact, *triangular inequality* is controversially discussed and is usually domain and application-oriented [21]:

vii. $Sim_{XDoc}(A, C) \geq Sim_{XDoc}(A, B) \times Sim_{XDoc}(B, C) \Rightarrow$ Triangular inequality.

Regarding semantic similarity in particular, most methods in the literature (e.g., [60, 87], cf. Section 2.3) including Lin [41], do not satisfy *triangular inequality*. An example by Tversky [80], reported by Maguitman *et al.* in [44], illustrates the *impropriety* of triangular inequality with an example about the similarity between countries: *"Jamaica is similar to Cuba (geographical proximity); Cuba is similar to Russia (political affinity); but Jamaica and Russia are not similar at all"*. And since we evaluate semantic similarity via Lin's measure [41] in our approach, our integrated semantic/structural approach does not transitively satisfy *triangular inequality*.

Note that when parameter $\alpha=1$, i.e., when our approach is utilized as a purely structural XML comparison method (i.e., only *Struct_CBS* is taken into account), our method behaves similarly to existing XML structural comparison methods, e.g., [12, 16, 55], provided that the distance values generated by the latter are evaluated via the similarity variant in *Formula (*12).

4.7. Overall Complexity

4.7.1. Time Complexity

The overall complexity of our integrated structural and semantic similarity approach simplifies to $O(|A|\times|B|\times|SN|\times Depth(SN))$, where $|A|$ and $|B|$ denote the cardinalities of the compared trees, $|SN|$ the cardinality of the semantic network exploited for semantic similarity assessment, and $Depth(SN)$ its maximum depth. It is computed as follows:

- *Struct_CBS* algorithm for the identification of the structural commonality between two sub-trees is of complexity: $O(|SbT_i|\times|SbT_j|)$ where $|SbT_i|$ and $|SbT_j|$ denote the cardinalities of the compared sub-trees.
- *Sem_RBS* for identifying the semantic resemblance between two sub-trees is of complexity: $O(|SbT_i|\times|SbT_j|\times|SN|\times Depth(SN))$. Note that $O(|SN|\times Depth(SN))$ underlines the time complexity of the semantic similarity measure itself [41].
- *TOC* algorithm for computing the costs of tree insert/delete operations, which makes use of *Struct_CBS* and *Sem_RBS* in identifying the structural commonalities and semantic resemblances between sub-trees in the source and destination trees, is of time complexity $\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(|SbT_i|\times|SbT_j|\times|SN|\times Depth(SN))$ and simplifies to $O(|A|\times|B|\times|SN|\times Depth(SN))$. The mathematical proof is provided in [75].
- The edit distance algorithm *TED* (an adaptation of the algorithm in [55]) which utilizes the results obtained by *TOC* (tree operations costs), is of complexity $O(|A|\times|B|\times|SN|\times Depth(SN))$.

When disregarding semantic similarity assessment, i.e., when input parameter $\alpha=1$ (thus disregarding algorithm *Sem_RBS*), our approach simplifies to $O(|A|\times|B|)$, similarly to existing XML-based tree edit distance comparison approaches, e.g., [12, 16, 55].

4.7.2. Space Complexity

As for memory usage, our approach requires RAM space to store the XML document trees being compared, as well as the distance matrixes and semantic vectors being computed. It simplifies to $O(|A|\times|B|)$ space (similarly to existing approaches, e.g., [12, 16, 55]) since:

- *Struct_CBS* requires $|SbT_i|\times|SbT_j|$ space for storing the distance matrix when identifying the structural commonalities between any two sub-trees SbT_i and SbT_j. Hence, space complexity is of $O(|SbT_i|\times|SbT_j|)$.
- *Sem_RBS* requires $2\times(|SbT_i|+|SbT_j|)$ space for handling corresponding sub-tree vectors, each vector being of maximal dimension $|SbT_i| + |SbT_j|$. Hence, *Sem_RBS* is of $O(|SbT_i|+|SbT_j|)$. Note that the semantic network is not stored in local memory, but is stored on disk (and is managed via a database system, cf. Section 6.6.1), and thus does not contribute to space complexity.
- *TOC* is of $\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(|SbT_i|\times|SbT_j|)$ space, for storing the various distance matrixes (*Struct_CBS*) and sub-tree vectors (*Sem_RBS*) between each pair of sub-trees in the source and destination XML trees. This simplifies to $O(|A|\times|B|)$ as shown in the previous section.
- The edit distance algorithm *TED* is of $O(|A|\times|B|)$ space complexity.

5. Comparison with Existing Approaches

In the following, we provide both theoretical and computational comparative analyses, evaluating our XML document similarity method against existing approaches.

[1] $Sim_{XDoc}(A, B) = 0$, means that computing the distance between A and B, consists of deleting all the nodes of the source tree, and then inserting all the nodes of the destination tree, i.e., $TED(A, B) = |A| + |B|$.

5.1. Formal Comparison

A formal mathematical comparison shows that some existing methods are lower bounds of our approach. This property conveys the fact that our method reduces edit operations costs following sub-tree similarities, and affects overall similarity values accordingly, whereas existing approaches usually exploit maximum edit operations costs regardless of the presence of sub-tree similarities, hence producing minimum similarity scores.

Theorem. Let A and B be XML trees, and $\mathrm{Sim}(A, B) = 1 - \frac{\mathrm{TED}(A, B)}{|A| + |B|}$, then:

- $\mathrm{Sim}_{\mathrm{Chawathe}}(A, B) \leq \mathrm{Sim}_{\mathrm{XDoc}}(A, B)$
- $\mathrm{Sim}_{\mathrm{Dalamagas\ et\ al.}}(A, B) \leq \mathrm{Sim}_{\mathrm{XDoc}}(A, B)$

Proof:

- *Proving that Chawathe's algorithm [12] is a lower bound of our XML comparison method is straight forward. When computing the distance between two trees using Chawathe's approach [12], all sub-trees are inserted/deleted via single node insertion/deletion operations regardless of the sub-tree similarities at hand. The costs of these insertions/deletions are equivalent to the maximum tree insertion/deletion operations costs following our TOC algorithm (Section 4.4), which yield a maximum edit distance, thus a minimum similarity value between the compared trees. In other words, Chawathe's algorithm [12] always yields similarity values lesser or equal to those computed via our approach.*
- *Proving that Dalamagas et al.'s algorithm [16] is a lower bound of our XML comparison method is also trivial. Indeed, the costs of tree insertion/deletion operations in [16] are computed as the sum of the costs of inserting/deleting all individual nodes in the considered sub-trees. These costs come down to the maximum tree operations costs computed following our method. Consequently, Dalamagas et al.'s algorithm [16] always yields similarity values that are lesser or equal to those computed via our method. Note that we do not consider the method's repetition/nesting reduction process in our analysis since it yields inaccurate comparison results in the general case (cf. Section 2.2.2)* □

As for Nierman and Jagadish's approach in [55], tree insertion/deletion operations costs are affected by the tree *contained-in* relation (cf. Section 2.2.2). Maximum costs (i.e., the costs of inserting/deleting all single nodes in the considered sub-trees) are attained when the *contained-in* relation is not verified. Otherwise, when the *contained-in* relation is verified, tree operations costs are minimal, and amount to the cost of inserting/deleting leaf nodes (normally unit costs, =1)[1]. Hence, we cannot mathematically conclude that the measure in [55] is a lower bound (or upper bound) of our XML comparison method since sub-tree costs are computed differently. In other words, the approach in [55] can yield similarity scores which are higher/lower than those produced by our method regardless of the similarities detected (since different mathematical cost schemes are utilized). However, it is clear that Nierman and Jagadish's approach only considers the *contained-in* relation between sub-trees while varying tree operations costs. On the other hand, our algorithm detects fined-grained structural and semantic similarities between sub-trees, among which the structural containment relation. Thus, our approach is able to detect a wider set or similarities w.r.t. the method in [55]. Thus, if we assume that sub-tree insertion/deletion costs in [55] are defined in accordance with our method (applying *TOC* while confining to the tree containment relation for instance, i.e., we only compute sub-tree insertion/deletion costs when the *contained-in* relation is verified), or vice-versa (assigning unit costs to tree operations used in our approach – instead of applying *TOC* – whenever sub-tree similarities are detected), then Nierman and Jagadish's algorithm would clearly yield similarity values that are lesser or equal to those obtained via our method.

Regarding the approach by Tekli *et al.* in [73], it focuses on the special case of semantic similarities between pairs of single node labels, particularly those having identical structural positions. Such similarities are covered in our current study, in the context of wider sub-tree semantic resemblances (an inner node would be treated as the root of its underlying sub-tree, whereas a leaf node would be simply viewed as a leaf node sub-tree). Yet, we cannot provide a formal mathematical comparison between both methods. In fact, node insertion/deletion operations costs are computed in a particular manner in [73], taking into account the semantic similarity between the node's label and that of its parent in the source/destination document tree. Thus, the approach in [73] yields similarity values that are not quantitatively comparable to those produced via our current method. Consider or instance XML document trees *X*, *Y* and *Z* in the example of Fig. 5:

- $\mathrm{Sim}_{\mathrm{Tekli\ et\ al.}}(X, Y) = 0.9432 > \mathrm{Sim}_{\mathrm{Tekli\ et\ al.}}(X, Z) = 0.8741$
- $\mathrm{Sim}_{\mathrm{XDoc}}(X, Y) = 0.9093 > \mathrm{Sim}_{\mathrm{XDoc}}(X, Z) = 0.8352$

Both methods detect that trees *X* and *Y* are semantically more similar than *X* and *Z*, w.r.t. the semantic network in Fig. 3. Yet, the similarity values are different, underlining that the methods are not quantitatively comparable.

5.2. Similarity Results for Motivating Examples

Hereunder, we present XML distance/similarity values obtained when applying our approach to treat the various XML comparison examples presented throughout the paper. Results in both Table 2 and Table 3 show that our XML similarity method is able to efficiently detect the various kinds of structural and semantic resemblances mentioned throughout the paper, which are left unaddressed by existing approaches (i.e., identical similarity values are obtained with existing approaches, despite the presence of structural and/or semantic similarities – values are omitted for ease of presentation), to the exception of a few cases detected by existing methods (discussed in Section 3).

[1] Please note that the minimum tree operation cost is not formally defined in [55]. We acquired this information from the authors.

Computational details are provided in the Appendix.

Table 2. Distance/similarity values obtained when comparing structurally similar documents, with parameter α set to 1 (*Struct_CBS* is exploited in computing tree operations costs).

	Our Approach		N. & J. [55]	Dalamagas *et al.* [16]	Chawathe [12]
	Distance	*Similarity*			
A/B	1.5	0.8636	*Detected*	*Not detected*	*Not detected*
A/C	3	0.7272			
A/D	3.2856	0.7473	*Not detected*		
A/E	5	0.6154			
F/G	5	0.5455			
F/H	7	0.3636			
F/I	4.2857	0.4643			
F/J	6	0.25			
K/L	0.5	0.9			
K/M	1	0.8			
K/N	1	0.8333			
K/P	2	0.6667			

Table 3. Distance/similarity values obtained when comparing semantically related documents with parameter α set to 0 (*Sem_RBS* is exploited in computing tree operations costs).

	Our Approach		N. & J. [55]	Dalamagas *et al.* [16]	Chawathe [12]	Tekli *et al.* [73]
	Distance	Similarity				
X/Y	0.8161	0.9093	*Not detected*	*Not detected*	*Not detected*	*Detected*
X/Z	1.4836	0.8352				*Not detected*
A'/B'	1.5189	0.8619				*Detected*
A'/C'	1.9604	0.8218				*Not detected*
A'/G'	2.3495	0.7389				
A'/H'	2.6641	0.7039				
A'/I'	3.0162	0.5691				
A'/J'	3.3308	0.5242				
K'/L'	0.5087	0.8983				
K'/M'	0.6103	0.8779				
K'/N'	1.0749	0.8208				
K'/P'	1.2205	0.7966				

6. Experimental Evaluation

6.1. Prototype

We have developed a prototype system, entitled *XS3* (*XML Structural and Semantic Similarity*)[1], to test, evaluate and validate our XML document comparison method, including implementations of its most recent alternatives in the literature. The *XS3* prototype, implemented using C#.Net, is made of four independent and interactive components, as well as various comparison and application modules:

- The *parser component* starts by verify the integrity of XML documents, undertaking lexical pre-processing and transforming documents into ordered labeled trees.
- The *similarity evaluation component* consists of several autonomous algorithms, including our approach and some of its most prominent alternatives which we refer to as *Chawathe* [12], *N & J* [55], *DCWS* [16], and *TCY* [73]. It is extensible to other approaches.
- The *Synthetic XML generator* produces sets of XML documents based on specific user requirements. It is an adaptation of the IBM XML documents generator[2] accepting as input: a DTD document, a *MaxRepeats*[3] value designating the maximum number of times a node will appear as child of its parent (when * or + options are encountered in the DTD), as well as a *NbDocs* value underscoring the number of documents to be produced.
- Furthermore, a *taxonomic analyzer component* was introduced to compute semantic similarity values between words (expressions) in a given semantic reference (e.g., WordNet [108]), to be subsequently exploited in evaluating XML element/attribute label similarity. It currently includes semantic measures developed in [97, 165] and is extensible to others.

In addition, *XS3* includes four XML document comparison modules, *One to One*, *One to Many*, *Many to Many* (consequently enabling XML document clustering), and *Set comparison* (for computing average inter-set and intra-set similarities, and evaluating clustering quality). The latter are thoroughly described in the following sections.

6.2. Evaluation Metrics

6.2.1. Background

How to experimentally evaluate the quality of an XML similarity method remains a debatable issue, especially in information retrieval. To our knowledge, the definition of standardized XML similarity evaluation metrics remains a hot topic in the INEX evaluation campaigns[4]. A few XML evaluation techniques have been proposed in the literature [16, 23, 55]. All of them use XML grammars (DTDs or XSDs) as reference for detecting structurally similar XML documents.

In [23], the authors compute inter-set and intra-set average similarities between documents corresponding to different DTDs and assess the attained scores to the *a priori* known DTDs. Results are depicted in a matrix where element (i, j) underscores the average similarity value, $Sim(S_i, S_j)$, corresponding to every pair of distinct documents such that the first belongs to set S_i (DTD_i) and the second to set S_j (DTD_j).

The authors in [16, 55] make use of clustering methods in order to group together structurally similar documents and subsequently evaluate how closely the obtained clusters correspond to the actual XML grammars. In addition, the authors in [16] adapt two metrics popular in information retrieval: *precision* and *recall* [62], in performing XML structural clustering evaluation. In the following, we report the definitions of those metrics and propose a method for extending their usage to obtain consistent experimental results.

6.2.2. Metrics Used

Owing to the proficient usage of their traditional predecessors in classic information retrieval evaluation, we make use of the *precision* (*PR*) and *recall* (*R*) metrics defined in [16], to evaluate the effectiveness of our approach and compare it to existing methods.

Following Dalamagas *et al.* [16], for an extracted cluster C_i that corresponds to a given XML grammar G_i (the cluster/grammar mapping issue is addressed subsequently):

- a_i is the number of XML documents in C_i that indeed correspond to G_i (correctly clustered documents).
- b_i is the number of documents in C_i that do not correspond to G_i (miss-clustered).

[1] Available at http://www.u-bourgogne.fr/Dbconf/XS3
[2] http://www.alphaworks.ibm.com
[3] A greater *MaxRepeats* increases the probability of attaining variability with optional and repeatable elements when generating XML documents.
[4] http://inex.is.informatik.uni-duisburg.de/

– c_i is the number of XML documents not in C_i, although they correspond to G_i (documents that should have been clustered in C_i).

Consequently, given n: the total number of generated clusters:

$$PR = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i} \in [0,1] \text{ and } R = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} c_i} \in [0,1] \quad \textbf{(13)}$$

High *precision* denotes that the clustering task achieved high accuracy, grouping together documents that actually correspond to the XML grammars mapped to the clusters. High *recall* means that very few documents are not in the appropriate cluster where they should have been. In addition to comparing one approach's *precision* improvement to another's *recall*, it is a common practice to consider the *F-value*, which represents the harmonic mean of *precision* and *recall*:

$$F\text{-}value = \frac{2 \times PR \times R}{PR + R} \in [0,1] \quad \textbf{(14)}$$

Therefore, as with traditional information retrieval evaluation, high *precision* and *recall*, and thus high *F-value* (indicating in our case excellent clustering quality) characterize a good similarity method.

6.3. Mapping Grammars to Clusters

Mapping XML grammars to XML document clusters comes down to mapping the groups of documents corresponding to each grammar (which we identify as *original grammar clusters*) to those created by the clustering process (which we identify as *extracted clusters*, or simply *clusters*). To get such a mapping, we compute the average intra-set similarity values between each *original grammar cluster* and *extracted cluster* and then identify the pairs of matching *grammars*/*extracted clusters* following the highest values. Note that in the following sections, the term *cluster* will always refer to *extracted cluster*.

6.4. Clustering XML Documents

In our experiments, we chose the well known single link hierarchical clustering method [27, 30] although any form of clustering could be utilized. Given n XML documents, we construct a fully connected graph G with n vertices (XML documents) and $(n \times (n\text{-}1))/2$ weighted edges. The weight of an edge corresponds to the similarity between the connected vertices. Consequently, the single link clusters for a similarity threshold s_i are identified by deleting all the edges with weights $< s_i$. Therefore, the single link clusters will group together XML documents that have pair-wise similarity values greater or equal than s_i.

However, unlike Dalamagas *et al.* in [16], we do not utilize a stopping rule to determine the most appropriate clustering level for the single link hierarchies, and thereafter obtain only one *PR*/*R* doublet for analysis with each clustering experiment. Instead, we compute a whole series of *PR*/*R* doublets. Those series correspond to the different clustering sets obtained by varying the clustering threshold in the *[0, 1]* interval. In other words, we construct a dendrogram (cf. Fig. 16) such as:

– For the initial clustering level, where the similarity threshold $s_1=0$ (or s_1 = minimum similarity attainable between any pair of documents), XML documents appear in one global cluster: the starting one.

– For the final clustering level, where the similarity threshold $s_n=1$ (with n the total number of levels, i.e., number of clustering sets in the dendrogram), each distinct document will appear in a different cluster.

– Intermediate clustering sets will be identified for thresholds s_i / $s_1 < s_i < s_n$.

Then, we compute *precision (PR)* and *recall (R)* for each clustering set identified in the dendrogram, thus constructing *PR* and *R* graphs that describe the system's evolution throughout the clustering process. We also compute average precision and recall values: *Ave(PR)* and *Ave(R)*, considering the whole dendrogram, on the basis of the obtained series, providing yet another indicator of clustering quality.

A sample dendrogram underlining the clustering evolution of 15 XML documents of the SIGMOD Record[1] (5 sampled from each of the *OrdinaryIssuePage.dtd*, *ProceedingsPage.dtd* and *SigmodRecord.dtd* grammars), is shown in Fig. 16.

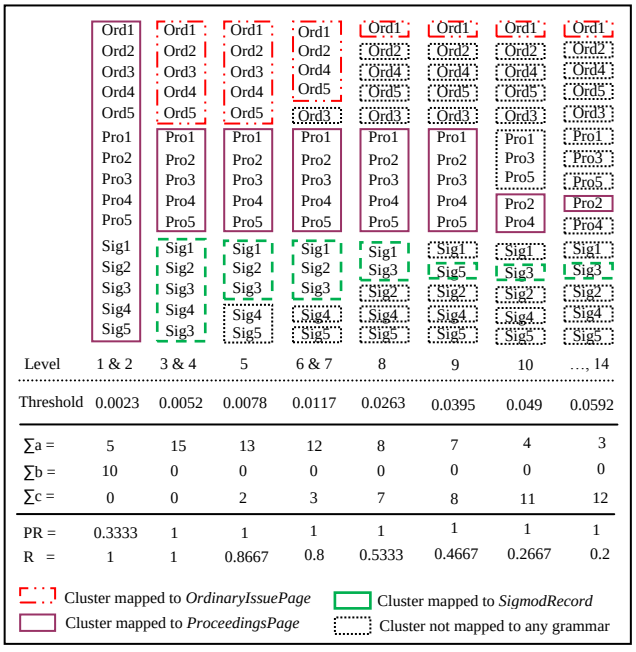


Fig. 16. Dendrogram and detailed *PR*/*R* computations when clustering 15 XML documents sampled from the SIGMOD record (here, clustering is based on structure, i.e., $\alpha = 1$).

6.5. Experimental Results

We conducted experiments on real and synthetic XML documents to test our XML comparison method. Results indicate that our approach yields improved clustering quality (i.e., comparison quality) than current approaches, w.r.t. both XML structural and semantic features. We detail each set of experiments in the following sub-sections.

[1] Available at http://www.sigmod.org/record/xml/

6.5.1. Structural Similarity Evaluation

To test our method's effectiveness in evaluating XML structural similarity, we conducted experiments on two sets of 750 documents, generated from 25 real-case[1] and synthetic XML grammars, using our adaptation of the IBM XML documents generator. We varied the *MaxRepeats* parameter to determine the number of times a node will appear as a child of its parent node. For a real dataset, we considered the online version of the ACM SIGMOD Record. We experimented on a set of 104 documents corresponding to *OrdinaryIssuePage.dtd* (30 documents), *ProceedingsPage.dtd* (47 documents) and *SigmodRecord.dtd* (27 documents)[2]. The characteristics of the document sets used are summarized in Table 4 and Table 5.

Table 4. Characteristics of the SIGMOD Record document set.

Grammars (DTDs)	Number of Documents	Average Node Depth (per doc)	Average nb of Elements (per doc)	Average nb of Attributes (per doc)
OrdinaryIssuePage	30	5.4997	179.9776	82.8333
ProceedingsPage	47	3.6739	264.5957	118.1277
SigmodRecord	27	5.7793	332.6667	210.2593

Table 5. Characteristics of synthetic XML document sets.

Document set	Number of Documents	Average Node Depth (per doc)	Average Number of Nodes (per doc)
MaxRepeats = 5	750	3.68	17.7067
MaxRepeats = 10	750	3.68	36.9133

Precision, *recall* and *F-value* graphs are presented in Figures 17, 18 and 19. Corresponding *Ave*(*PR*), *Ave*(*R*) and *Ave*(*F-value*) values are reported in Table 6.

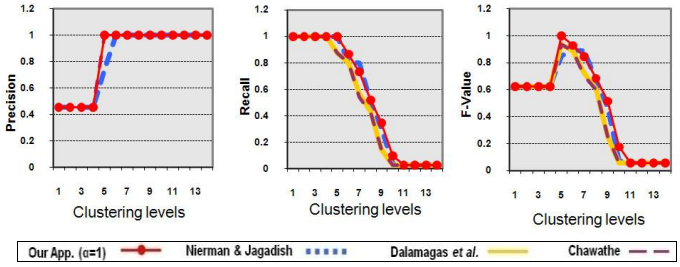


Fig. 17. *PR*, *R* and *F-Value* graphs for clustering real SIGMOD Record XML documents.

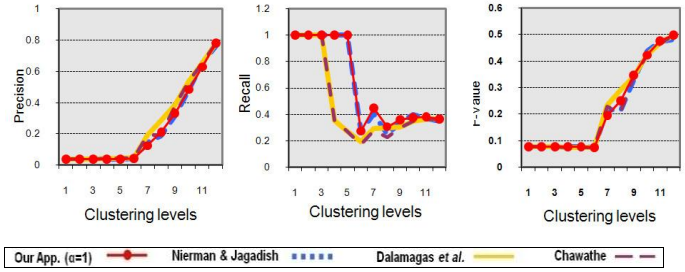


Fig. 18. *PR*, *R* and *F-Value* graphs for clustering documents of synthetic set 1 (MaxRepeats = 5).

[1] From http://www.xmlfiles.com and http://www.w3schools.com.

[2] We found only one XML file conforming to the *SigmodRecord.dtd* grammar: *SigmodRecord.xml*. However, due to its relatively large size (479KB) in comparison with the XML documents corresponding to the other two DTDs (10KB of average size per document), we carefully decomposed *SigmodRecord.xml* to several documents, creating a set of XML documents conforming to *SigmodRecord.dtd*.

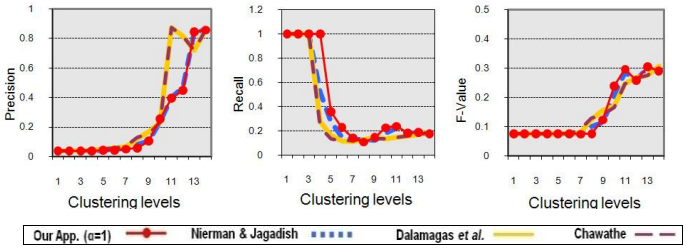


Fig. 19. *PR*, *R* and *F-value* graphs for clustering documents of synthetic set 2 (MaxRepeats = 10).

Table 6. Average *PR*, *R* and *F-values* obtained by varying the clustering threshold between *[0, 1]*.

	SIGMOD			*Set 1 (MaxRepeats=5)*			*Set 2 (MaxRepeats =10)*		
	PR	*R*	*F-value*	*PR*	*R*	*F-value*	*PR*	*R*	*F-value*
Chawathe	0.8782	0.3910	0.5411	0.2502	0.4737	0.3619	0.2783	0.3769	0.3276
DCWS	0.8782	0.3931	0.5432	0.2581	0.4838	0.3709	0.2779	0.3821	0.3300
N & J	0.8637	0.4268	0.5713	0.2334	0.6162	0.4248	0.2234	0.4177	0.3205
Our App. (α=1)	**0.8782**	**0.4326**	**0.5797**	**0.2341**	**0.6262**	**0.4302**	**0.2203**	**0.4656**	**0.3430**

Results, with respect to all three data sets, indicate that our approach yields improved global clustering quality (i.e., structural comparison quality) in comparison with current alternative approaches. For the SIGMOD Record document set, our method yields an average overall *precision* higher than that of *N & J* and identical to those achieved by *DCWS* and *Chawathe*'s algorithms. As for *recall*, our approach shows better results than *N & J*, *DCWS* as well as *Chawathe*. In fact, average *F-value* results underline our method's higher clustering efficiency (i.e., comparison quality). For the synthetic datasets, our method yields average *precision* levels lower than those achieved by its predecessors, to the exception of the first synthetic dataset (*MaxRepeats=5*) where our approach outranks *N&J*'s average *precision* level. However, our method consistently maintains *recall* levels higher than those of its alternatives. In cases where higher/lower *precision*/*recall* levels are obtained simultaneously, the *F-value* measure is fundamental in assessing the overall loss and gain in average *precision*/*recall*, and evaluating result quality. For both synthetic datasets, our method yields higher average *F-values* in comparison with *N&J*, *DCWS*, and *Chawathe*.

Note that the low *precision* levels obtained with the synthetic datasets are probably due to utilizing relatively similar grammars (we explicitly used grammars baring sub-tree similarities) in generating the document sets. Similar grammars would induce similar documents. Such documents could thus be easily miss-clustered if their structural similarities are detected, which is the case when using our approach (the clusters include the right documents as well as additional similar ones). Existing approaches disregard various kinds of similarities, e.g., sub-tree similarities, which is why they tend to distinguish documents that are in fact similar. Such undetected similarities might yield better *precision* levels (smaller clusters including only portions of correctly clustered documents). Nonetheless, they consistently yield lower *recall* values (lots of documents are not in the appropriate clusters where they should have been).

6.5.2. Evaluation of Structural and Semantic Similarity

Various experiments were conducted in order to validate our approach's ability of integrating semantic similarity evaluation

in XML document comparison. In addition to hierarchical clustering [30], we utilized the inter-set/intra-set average similarity technique introduced in [23] which seemed effective in evaluating the semantic relatedness between groups of XML documents. We exploited (extracts of) WordNet as the reference semantic network, weighted based on the Brown Corpus of American English [25][1]. Synthetic XML documents generated based on real and synthetic XML grammars[2] were considered (All test documents and grammars are published online[3] to facilitate future comparative evaluations). We selected general purpose XML grammars describing real world data, to allow relevant semantic evaluation using WordNet (which is a general purpose semantic reference describing every day English language [48]). Otherwise, it would be useless to evaluate the semantics of XML labels given a reference that does not encompass corresponding semantic concepts (for instance, it would be futile to compare XML documents describing protein sequences, using the general purpose WordNet, since most semantic concepts related to protein descriptions do not exist in WordNet, and require a dedicated semantic reference).

Note that the number of documents utilized in our combined semantic/structural similarity evaluation is reduced w.r.t. the structural similarity experiments, because of the complexity of the semantic similarity process due to traversing the reference semantic network (as shown in Section 6.6.1).

6.5.2.1. XML Document Clustering Experiments

Clustering experiments were conducted on six sets of 15 XML documents, generated based on 9 DTD grammars (some of which are shown in Fig. 26), using our XML documents generator. We varied the *MaxRepeats* parameter between 5 and 10. The characteristics of the produced document sets are summarized in Table 7. *PR*, *R* and *F-value* graphs are presented in Figures 20 - 25. Corresponding *Ave(PR)*, *Ave(R)* and *Ave(F-value)* values are reported in Tables 8 and 9.

Table 7. Characteristics of synthetic XML document sets.

Doc sets	Max Repeats	Grammars	N° of Docs	Avg Depth (per doc)	Avg Number of Nodes (per doc)
S1	5	Academy.dtd, College.dtd, Factory.dtd	15	2.1672	10.6667
S2	5	InstA.dtd, InstB.dtd, InstC.dtd	15	1.465	9.2
S3	5	InstK.dtd, InstL.dtd, InstM.dtd	15	0.7433	4.1333
S4	10	Academy.dtd, College.dtd, Factory.dtd	15	2.252	9.9333
S5	10	InstA.dtd, InstB.dtd, InstC.dtd	15	1.5459	19
S6	10	InstK.dtd, InstL.dtd, InstM.dtd	15	0.8361	6.5333

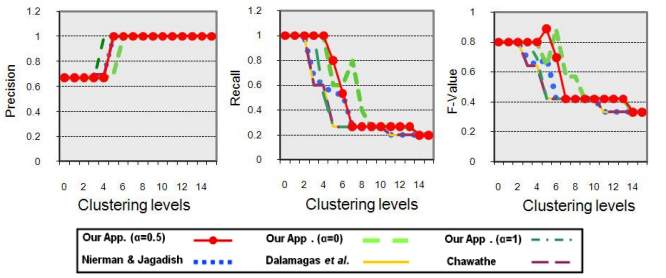


Fig. 20. *PR*, *R*, and *F-value* graphs for clustering documents of S_1.

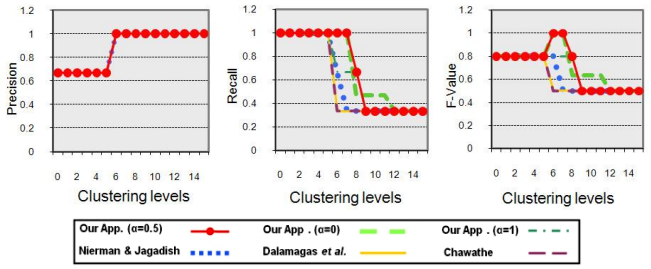


Fig. 21. *PR*, *R*, and *F-value* graphs for clustering documents of S_2.

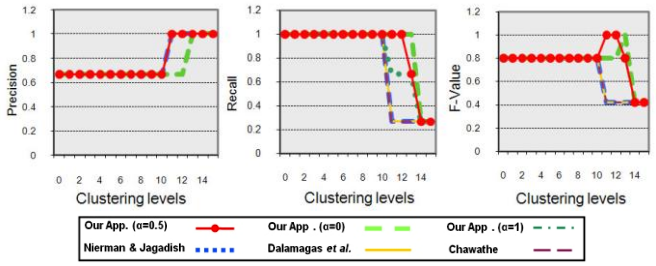


Fig. 22. *PR*, *R*, and *F-value* graphs for clustering documents of S_3.

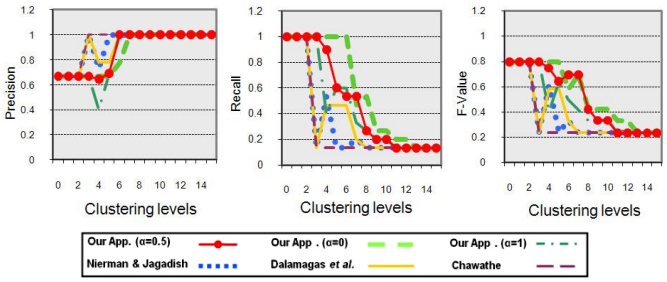


Fig. 23. *PR*, *R*, and *F-value* graphs for clustering documents of S_4.

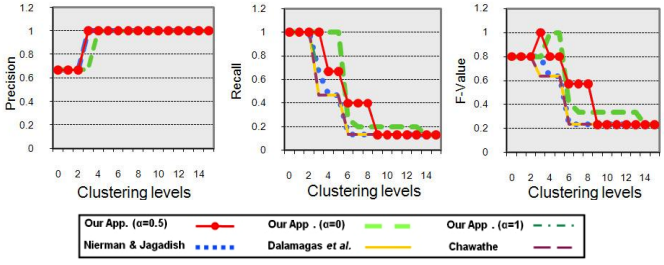
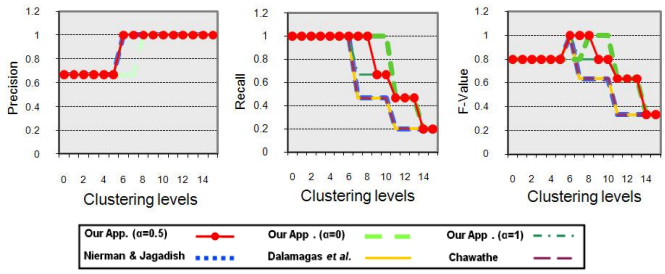


Fig. 24. *PR*, *R*, and *F-value* graphs for clustering documents of S_5.



[1] http://www.cogsi.princeton.edu/cgi-bin/webwn.
[2] From http://www.xmlfiles.com and http://www.w3schools.com.
[3] www.u-bourgogne.fr/DbConf/XS3

Fig. 25. *PR*, *R*, and *F-value* graphs for clustering documents of S_6.

Table 8. Average *PR*, *R* and *F-values* obtained by varying the clustering threshold between *[0, 1]*.

	S1			*S2*			*S3*		
	PR	*R*	*F-value*	*PR*	*R*	*F-value*	*PR*	*R*	*F-value*
Chawathe	0.9231	0.3714	0.5297	0.9048	0.5238	0.6635	0.8254	0.6508	0.7278
DCWS	0.9231	0.3714	0.5297	0.9048	0.5238	0.6635	0.8254	0.6508	0.7278
N & J	0.9241	0.3873	0.5458	0.9048	0.5397	0.6761	0.8254	0.6508	0.7278
Our App. (α=1)	0.9365	0.3968	0.5574	0.9048	0.5714	0.7005	0.8254	0.7079	0.7622
Our App. (α=0)	**0.9060**	**0.4730**	**0.6215**	**0.9048**	**0.6127**	**0.7306**	**0.7937**	**0.7556**	**0.7741**
Our App. (α=0.5)	**0.9206**	**0.4571**	**0.6109**	**0.9048**	**0.6032**	**0.7238**	**0.8254**	**0.7397**	**0.7802**

Table 9. Average *PR*, *R* and *F-values* obtained by varying the clustering threshold between *[0, 1]*.

	S4			*S5*			*S6*		
	PR	*R*	*F-value*	*PR*	*R*	*F-value*	*PR*	*R*	*F-value*
Chawathe	0.9524	0.2571	0.4049	0.9524	0.3048	0.4616	0.9048	0.5175	0.6584
DCWS	0.9312	0.2921	0.4447	0.9524	0.3048	0.4618	0.9048	0.5175	0.6584
N & J	0.9394	0.2794	0.4307	0.9524	0.3143	0.4726	0.9048	0.5175	0.6584
Our App. (α=1)	0.8933	0.3587	0.5119	0.9524	0.3714	0.5344	0.9048	0.5937	0.7169
Our App. (α=0)	**0.8942**	**0.4413**	**0.5909**	**0.9365**	**0.4095**	**0.5699**	**0.8730**	**0.6571**	**0.7499**
Our App. (α=0.5)	**0.9049**	**0.4079**	**0.5623**	**0.9524**	**0.3873**	**0.5507**	**0.9048**	**0.6254**	**0.7396**

Results, w.r.t. all six data sets, underline our approach's improved global clustering quality (i.e., XML comparison quality) in comparison with alternative approaches, when it is exploited as a purely structural comparison method (parameter *α=1*), and specifically when it is utilized as an integrated structural and semantic similarity method (*α=0* and *α=0.5*).

- When *α=1*, the system only considers sub-tree structural similarities (via *Struct_CBS*, cf. Fig. 9) in the comparison process.
- When *α=0*, the system only considers sub-tree semantic resemblances (via *Sem_RBS*, cf. Section 4.3), disregarding sub-tree structural similarities in the comparison process.
- When *α=0.5*, the system equally consider sub-tree structural and semantic features in the comparison process. In other words, all kinds of sub-tree similarities, structural and semantic (detailed in Sections 4.2 and 4.3), are detected, both *Struct_CBS* and *Sem_RBS* algorithms being executed.

In fact, our integrated structural and semantic similarity approach consistently maintains higher *recall* levels, in comparison with its structural version (*α=1*), *N & J*, *DCWS* as well as *Chawathe*. As for *precision*, our method tends to yield average levels that are identical to those attained using existing comparison methods, which is underlined in the results corresponding to sets S_2, S_3, S_5, and S_6. In a few cases, it achieved lower *precision*, i.e., with sets S_1 and S_4.

Nonetheless, in all six tests, average *F-value* results, characterizing both *precision* and *recall* levels simultaneously, underline our method's effectiveness w.r.t. its alternatives (with both *α=0* and *α=0.5*). Note that similarly to the structural evaluation results shown in the previous section, our integrated method's low *precision* levels are due to utilizing relatively similar grammars in generating the document sets: we explicitly used grammars baring semantic sub-tree similarities. On one hand, higher *recall* scores are sometimes obtained with *α=0* (*Sem_RBS* being considered with a maximum unit weight), in comparison with the case where *α=0.5* (where both *Struct_CBS* and *Sem_RBS* are considered with equal *0.5* weights, hence downscaling the impact of sub-tree semantic relatedness, and thus missing certain sub-tree semantic similarities when clustering documents). On the other hand, since existing approaches disregard semantic similarities, they tend to distinguish documents that are in fact similar, and place them in separated clusters. Such undetected similarities might yield better *precision* levels, i.e., smaller clusters including portions of correctly clustered documents. Nonetheless, they consistently yield lower *recall* values (and consequently low *F-values*) since lots of documents are not in the appropriate clusters where they should have been.

In the experiments above, we did not compare our method's effectiveness to *TCY* [73] due to the latter's asymmetric nature which is not suitable for applying our clustering algorithms. However, we considered *TCY* in our inter-set/intra-set evaluation experiments.

6.5.2.2. Inter-set & Intra-set Average Similarities Experiments

In the following, we present inter-set and intra-set average similarity results when comparing 5 sets of XML documents. Each set is made of 10 documents synthetically generated w.r.t. the DTD grammars shown in Fig. 26, varying the *MaxRepeats* factor between *5* and *10*.

```
<!DOCTYPE Academy [
<!ELEMENT Academy (Unit+)>
 <!ELEMENT Unit (Branch?)>
   <!ELEMENT Branch (Professor?,
                     PhDStudent+)>
   <!ELEMENT Professor (#PCDATA)>
   <!ELEMENT PhDStudent (#PCDATA)> ]
```

```
<!DOCTYPE College[
<!ELEMENT College (Unit+)>
  <!ELEMENT Unit (Branch?)>
    <!ELEMENT Branch (Lecturer?,
                      Scholar*)>
     <!ELEMENT Lecturer (#PCDATA)>
     <!ELEMENT Scholar (#PCDATA)> ]
```

```
<!DOCTYPE Factory [
<!ELEMENT Factory (Unit+)>
  <!ELEMENT Unit(Branch?)>
    <!ELEMENT Branch (Supervisor?,
                      Worker+)>
      <!ELEMENT Supervisor (#PCDATA)>
      <!ELEMENT Worker (#PCDATA)>  ]
```

```
<!DOCTYPE EduInst [
<!ELEMENT Institution(Academy+, College+)>
   <!ELEMENT Academy (Professor,
                      PhDStudent)>
   <!ELEMENT College (Lecturer, Scholar)>
   <!ELEMENT Professor (#PCDATA)>
   <!ELEMENT PhDStudent (#PCDATA)>
   <!ELEMENT Lecturer (#PCDATA)>
   <!ELEMENT Scholar (#PCDATA)>  ]
```

```
<!DOCTYPE Inst [
<!ELEMENT Institution (Academy+, Factory+)>
   <!ELEMENT Academy (Professor, PhDStudent)>
   <!ELEMENT Factory (Supervisor, Worker)>
        <!ELEMENT Professor (#PCDATA)>
        <!ELEMENT PhDStudent (#PCDATA)>
        <!ELEMENT Supervisor (#PCDATA)>
        <!ELEMENT Worker (#PCDATA)>  ]
```

Fig. 26. Sample DTD grammars inducing sets of XML document.

Recall that *a priori* known DTD grammars (inducing predefined document sets) serve as a reference for assessing the similarity results [23]. Intra-set average similarities are computed between documents of the same set S_i, reported as *(i, i)* values in the similarity matrix. Remaining *(i, j)* values correspond to intra-set average similarities, computed between documents belonging to sets S_i and S_j. Results are shown in Tables 10 and 11.

Note that we report our method's results when parameter *α = 0* (detecting sub-tree semantic resemblances) and *α = 1* (detecting sub-tree structural similarities), and omit those corresponding to *α=0.5* (considering both sub-tree structural and semantic features) since our aim here is to contrast our system's capability in detecting XML semantic resemblance w.r.t. structural similarity.

First of all, results show that our method, in both sub-tree structural and semantic facets, produces intra-set average

similarity values underlining a straight distinction between documents belonging to a given set (i.e., conforming to a given grammar) and others outside that set, similarly to existing XML comparison approaches.

Table 10. Average inter-set/intra-set similarities (tests conducted on 25 documents, 5 of each set, generated with *MaxRepeats*=5).

a. Our approach – semantic resemblance (α = 0).

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.3288	0.2783	0.1495	0.0947	0.0926
S2 (College.dtd)	0.2783	0.3621	0.1458	0.0904	0.0866
S3 (Factory.dtd)	0.1495	0.1458	0.3162	0.0826	0.0851
S4 (EduInst.dtd)	0.09468	0.0904	0.0826	0.3932	0.1802
S5 (Inst.dtd)	0.0926	0.0866	0.0851	0.1802	0.3932

b. Our approach - structural similarity (α = 1).

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0. 3222	0.1142	0.1059	0.0756	0.0756
S2 (College.dtd)	0.1142	0.3523	0.1012	0.0746	0.0558
S3 (Factory.dtd)	0.1059	0.1012	0.2988	0.0543	0.0730
S4 (EduInst.dtd)	0.0756	0.0746	0.0543	0.3932	0.1216
S5 (Inst.dtd)	0.0756	0.0558	0.0730	0.1216	0.3932

c. *TCY* [73].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.4568	0.4315	0.2673	0.1676	0.1602
S2 (College.dtd)	0.4319	0.4828	0.2709	0.1615	0.1518
S3 (Factory.dtd)	0.2626	0.2659	0.4011	0.1444	0.1523
S4 (EduInst.dtd)	0.1705	0.1723	0.1540	0.4429	0.3042
S5 (Inst.dtd)	0.1654	0.1617	0.1647	0.3042	0.4429

d. N & J [55].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.2321	0.1067	0.0975	0.0587	0.0587
S2 (College.dtd)	0.1067	0.2901	0.0938	0.0558	0.0558
S3 (Factory.dtd)	0.0975	0.0938	0.2160	0.0543	0.0549
S4 (EduInst.dtd)	0.0587	0.0558	0.0543	0.3213	0.1177
S5 (Inst.dtd)	0.0587	0.0558	0.0549	0.1177	0.3213

e. *DCWS* [16].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.2169	0.1067	0.0975	0.0587	0.0587
S2 (College.dtd)	0.1067	0.2644	0.0938	0.0558	0.0558
S3 (Factory.dtd)	0.0975	0.0938	0.2069	0.0543	0.0543
S4 (EduInst.dtd)	0.0587	0.0558	0.0543	0.2779	0.1143
S5 (Inst.dtd)	0.0587	0.0558	0.0543	0.1143	0.2779

f. Chawathe [12].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.2169	0.1067	0.0949	0.0587	0.0587
S2 (College.dtd)	0.1067	0.2644	0.0910	0.0558	0.0558
S3 (Factory.dtd)	0.0949	0.0910	0.1633	0.0539	0.0539
S4 (EduInst.dtd)	0.0587	0.0558	0.0539	0.2779	0.1143
S5 (Inst.dtd)	0.0587	0.0558	0.0539	0.1143	0.2779

Yet, when considering the semantics of XML sub-trees (e.g., with parameter α=0), our approach captures the semantic affinities between documents of different sets:

- Tables 10.a and 11.a show that document sets S_1 and S_2 share more semantic resemblances than sets S_1 and S_3, sets S_1/S_2 being structurally almost as similar as S_1/S_3 (cf. Tables 10, 11 - b, d, e, f).
- Tables 10.a and 11.a also show that document set S_1 shares more semantic meaning with set S_4 than with set S_5, sets S_1/S_4 and S_1/S_5 being structurally identical when factor MaxRepeats=5 (Tables 10, 11 - b, d, e, f).

Table 11. Average inter-set/intra-set similarities (tests conducted on 25 documents, 5 of each set, with *MaxRepeats* = 10).

a. Our approach – semantic resemblance (α = 0).

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.3991	0.2878	0.1755	0.0588	0.0628
S2 (College.dtd)	0.2878	0.3091	0.1695	0.0574	0.0609
S3 (Factory.dtd)	0.1755	0.1695	0.3089	0.0510	0.0599
S4 (EduInst.dtd)	0.0588	0.0574	0.0510	0.3864	0.0856
S5 (Inst.dtd)	0.0628	0.0609	0.0599	0.0856	0.3049

b. Our approach - structural similarity (α = 1).

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.3937	0.1233	0.1233	0.0416	0.0452
S2 (College.dtd)	0.1233	0.3027	0.1169	0.0413	0.0358
S3 (Factory.dtd)	0.1233	0.1169	0.3027	0.0326	0.0469
S4 (EduInst.dtd)	0.0416	0.0413	0.0326	0.3036	0.0587
S5 (Inst.dtd)	0.0452	0.0358	0.0469	0.0587	0.3049

c. *TCY* [73].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.4373	0.3906	0.2855	0.1096	0.1073
S2 (College.dtd)	0.3918	0.3740	0.2848	0.1074	0.1025
S3 (Factory.dtd)	0.2781	0.2770	0.3529	0.0898	0.1119
S4 (EduInst.dtd)	0.0939	0.0934	0.0883	0.458	0.1660
S5 (Inst.dtd)	0.1034	0.1013	0.1006	0.1661	0.3445

d. *N & J* [55].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.2935	0.1195	0.1195	0.0333	0.0367
S2 (College.dtd)	0.1195	0.2045	0.1125	0.0326	0.0358
S3 (Factory.dtd)	0.1195	0.1125	0.2045	0.0326	0.0358
S4 (EduInst.dtd)	0.0333	0.0326	0.0326	0.2627	0.0564
S5 (Inst.dtd)	0.0367	0.0358	0.0358	0.0564	0.2506

e. *DCWS* [16].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.2869	0.1195	0.1195	0.0333	0.0367
S2 (College.dtd)	0.1195	0.1942	0.1125	0.0326	0.0358
S3 (Factory.dtd)	0.1195	0.1125	0.1942	0.0326	0.0358
S4 (EduInst.dtd)	0.0333	0.0326	0.0326	0.2391	0.0545
S5 (Inst.dtd)	0.0367	0.0358	0.0358	0.0545	0.2192

f. *Chawathe* [12].

	S1	S2	S3	S4	S5
S1 (Academy.dtd)	0.1799	0.1051	0.1050	0.0333	0.0367
S2 (College.dtd)	0.1051	0.1363	0.1005	0.0326	0.0357
S3 (Factory.dtd)	0.1050	0.1005	0.1362	0.0326	0.0357
S4 (EduInst.dtd)	0.0333	0.0326	0.0326	0.2391	0.0545
S5 (Inst.dtd)	0.0367	0.0357	0.0357	0.0545	0.2192

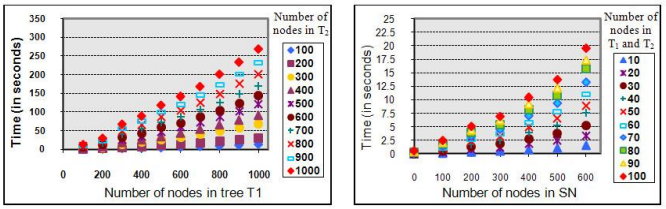
Thus, as shown in the inter/intra-set similarity values, semantic resemblances are left undetected using existing XML comparison methods, i.e., *N & J*, *DCWS* and *Chawathe*.

Note that *TCY* [73] is able to capture certain semantic similarities as shown in the results above. Yet, as discussed previously, it disregards various sub-tree semantic resemblances in comparing XML documents (cf. Section 3.2). In addition, it is asymmetric (e.g., $Sim(S_1, S_2) \neq Sim(S_2, S_1)$ as shown in the average inter-set similarity results), which is not in accordance with the formal definition of similarity (Section 4.6).

6.6. Performance Evaluation

6.6.1. Verifying Complexity Levels

As shown in Section 4.7, our XML comparison method is of $O(|T_1|\times|T_2|\times|SN|\times Depth(SN))$ time complexity. It simplifies to $O(|T_1|\times|T_2|)$ when semantic similarity evaluation is disregarded (*Sem_RBS* is disregarded). We start by verifying our approach's polynomial (quadratic) dependency on tree size, i.e., $O(|T_1|\times|T_2|)$. Timing experiments were carried out on a PC with an Intel Xeon 2.66 GHz processor with 2GB RAM. As predicted, results in Fig. 27.a reflect an almost perfect linear dependency on the size of each tree being compared.



a. Structural similarity evaluation (*Sem_RBS* deactivated).

b. Integrating semantic similarity evaluation.

Fig. 27. Timing results.

On the other hand, when evaluating both structural and semantic similarity (i.e, when both *Struct_CBS* and *Sem_RBS* algorithms are considered), the size of the reference semantic network, exploited while evaluating the semantic similarity measure (e.g., Lin's measure [41]) to compute pair-wise XML node label similarity, comes to play.

To our knowledge, timing analysis for Lin's measure [41] was not carried out previously. Theoretically, it can be estimated as $O(|SN|\times Depth(SN))$ [41] due to traversing the semantic network when searching for the lowest common ancestor between two taxonomic nodes (cf. Section 2.3.2). Thus, in order to reduce our method's overall complexity, we pre-compute semantic similarities for each pair of nodes in the taxonomy considered (which took about 20 seconds for the WordNet fragment depicted in Fig. 3, and more than 5 CPU hours for a 600 node semantic network) and store the results in a dedicated indexed table (Oracle 9i DB)[1]. As a result, *Sem_RBS* would access the indexed table to acquire semantic values instead of traversing the taxonomy to compute semantic similarity each time it is needed (pair-wise similarity values are computed once, prior to XML document comparison). Due to this process, we eliminated the impact of taxonomic depth

[1] Oracle uses the *B-Tree* indexing technique.

on overall timing complexity. Timing results in Fig. 27.b show that our approach becomes linearly dependent on the size on the taxonomy considered, complexity simplifying from $O(|T_1|\times|T_2|\times|SN|\times Depth(SN))$ to $O(|T_1|\times|T_2|\times|SN|)$.

As for space complexity, memory usage results confirm that our approach is quadratic in the combined size of the trees being compares, $O(|T_1|\times|T_2|)$, which underlines a linear dependency on the size of each tree (memory usage graphs are similar in overall shape to those depicted in Fig. 27, and thus are omitted for clearness of presentation, cf. [75] for details).

6.6.2. Comparison with Existing Approaches

In addition to verifying the complexity levels of our approach, we assess its overall efficiency w.r.t. its most prominent alternatives, e.g., *N & J* [55], *DCWS* [16] and *Chawathe* [12]. Results in Fig. 28 depict our method's time performance as a structural similarity method, disregarding semantic evaluation for fairness of comparison. Results demonstrate that our method's time performance is closely comparable to those of its alternatives, e.g., *N & J* [55], *DCWS* [16], and *Chawathe* [12] (which are also of $O(|T_1|\times|T_2|)$ time). Note that *Chawathe*'s superior performance was expected since the algorithm was originally conceived to provide higher efficiency levels [12] (in order to allow efficient external-memory computations, cf. Section 2.2.2), in comparison with our study (as well as *N & J* [55] and *DCWS* [16]), which targets result quality (e.g., clustering effectiveness [16, 55]) and higher comparison accuracy. Nonetheless, we are currently investigating several techniques related to XML similarity and performance enhancement, such as Prufer sequence encoding [4], B-Tree indexing [19] and Entropy [31], aiming to improve our method's performance level, without however affecting its effectiveness and result quality.

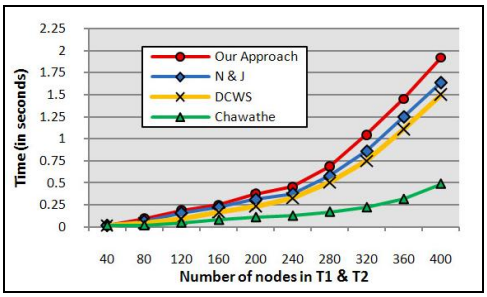


Fig. 28. Time comparison with existing approaches.

7. Conclusion

In this paper, we propose a fine-grained similarity approach for comparing rigorously structured XML documents. We particularly target document structure (i.e., *structure-only* XML, consisting of element/attribute tag names) and disregard content (i.e., element/attribute values), central in structural clustering/classification and structural querying applications. Our method combines tree edit distance computations and information retrieval semantic similarity assessment, so as to capture the structural and semantic resemblances between XML documents. We particularly focus on previously unaddressed sub-tree structural and semantic similarities, allowing the user to tune the comparison process according to her requirements and needs. Our theoretical study and experimental evaluation showed that our approach yields improved similarity results w.r.t. existing alternatives. Timing analysis underlined the impact of semantic similarity assessment, due to traversing the semantic network at hand.

We showed our approach's applicability in a generic Information Retrieval context (using fragments of WordNet). Apparently, adding semantic assessment to the edit distance computation process is a good thing, provided the semantic network (i.e., knowledge base) considered is relevant w.r.t. the documents at hand (WordNet is relevant for comparing generic XML documents representing real world data, such as those utilized in our experiments, but might not be useful when comparing XML documents describing gene and protein sequences [1], or multimedia MPEG-7 documents [50]…). Achieving improved XML similarity results would require an accurate, domain specific and complete semantic network, which up till now, rarely exist. Besides, the complexity of the semantic similarity process due to traversing the reference semantic network remains a major drag to performance, to be investigated in a dedicated future study.

In addition to improving the performance levels of our method, we are also currently investigating various optimization techniques, mostly based on machine learning such as Hopfield Neural Networks [32], Sigmoid [20], and Harmony [49], in order to enable a (semi-automatic) fine-tuning of our XML comparison process, giving more/less emphasis to XML structural and/or semantic properties (by calibrating parameter α) following the nature of the XML documents being compared. Other future directions include exploiting semantic similarity to compare, not only the structure of XML documents (element/attribute labels), but also their contents (values). Here, XML Schemas, underlining element/attribute data-types, come to play. In addition, we plan to extend our method toward XML document/grammar comparison. Few studies have addressed the latter issue, especially from a semantic perspective, which remains virtually uncharted territory.

Acknowledgements

We are grateful to the reviewers for their valuable comments and suggestions which have allowed us to further improve the manuscript's presentation, organization, and contents.

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

References

[1] Adak S.; Batra V.S.; Bhardwaj D.N.; Kamesam P.V.; Kankar P.; Kurhekar M.P. and Srivastrava B., *A System for Knowledge Management in Bioinformatics.* Inter. Conf. on Information and Knowledge Management (CIKM), 2002. pp. 638-641.

[2] Aho A.; Hirschberg D.; and Ullman J., *Bounds on the Complexity of the Longest Common Subsequence Problem.* J. of the ACM, 1976. 23(1):1-12.

[3] Algergawy A.; Nayak R. Saake G., *XML schema element similarity measures: A schema matching context.* Proc. of the Inter. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2009), 2009, 1246-1253.

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

[5] Amer-Yahia S.; Lakshmanan L.; and Pandit S., *FleXPath: Flexible Structure and Full-Text Querying for XML*. Inter. ACM SIGMOD Conf., 2004, 83-94.

[6] 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 Info. Systems, 2004. (29):23-46.

[7] Bille P., *A Survey on Tree Edit Distance and Related Problems.* Theoretical Computer Science, 2005. 337(1-3):217-239.

[8] Boughanem M., *Introduction to Information Retrieval.* Proc. of EARIA'06 (Ecole d'Automne en Recherche d'Information et Application), 2006. Ch 1.

[9] Buttler D., *A Short Survey of Document Structure Similarity Algorithms.* Proc. of the Inter. Conf. on Internet Computing (ICOMP), 2004. pp. 3-9.

[10] Candillier L.; Tellier I.; and Torre F., *Transforming XML Trees for Efficient Classification and Clustering.* Proc. of the Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), 2005, 469-480.

[11] Carmel D.; Efraty N.; Landau G.M.; Maarek Y.S. and Y. Mass, *An Extension of the Vector Space Model for Querying XML Documents via XML Fragments.* ACM SIGIR Workshop on XML and Information Retrieval, 2002. pp. 14-25.

[12] Chawathe S., *Comparing Hierarchical Data in External Memory.* Proc. of the Inter. VLDB Conf., 1999. pp. 90-101.

[13] Chawathe S. *et al.*, *Change Detection in Hierarchically Structured Information.* Proc. of the Inter. ACM SIGMOD Conf. , 1996, 26-37. Montreal.

[14] Cobéna G.; Abiteboul S.; and Marian A., *Detecting Changes in XML Documents.* IEEE Inter. Conf. on Data Engineering (ICDE), 2002, 41-52.

[15] D'Ulizia A.; Ferri F.; Formica A.; Grifoni P. and Rafanelli M., *Structural similarity in geographical queries to improve query answering.* Proc. of the 2007 ACM Symposium on Applied Computing (SAC), 2007. pp. 19-23.

[16] Dalamagas T. *et al.*, *A Methodology for Clustering XML Documents by Structure.* Information Systems, 2006. 31(3):187-228.

[17] Do H. and Rahm E., *COMA: A System for Flexible Combination of Schema Matching Approaches.* Inter. VLDB Conf. 2002. pp. 610-621.

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

[19] DuChateau F.; Bellahsene Z.; Hunt E.; Roantree M., a.R.M., *An Indexing Structure for Automatic Schema Matching.* Inter. Conf. on Data Engineering (ICDE) - Workshops, 2007. pp. 485-491.

[20] Ehrig M. and Staab S., *QOM - Quick Ontology Mapping.* In Proc. of the Inter. Semantic Web Conference (ISWC), 2004. pp. 683-697.

[21] Ehrig M. and Sure Y., *Ontology Mapping - an Integrated Approach.* Proc. of the European Semantic Web Conference (ESWC), 2004, 76-91.

[22] Ehrig M.; Staab S. and Sure Y., *Bootstrapping Ontology Alignment Methods with APFEL.* In Proc. of the Inter. WWW Conf., pp. 1148-1149.

[23] Flesca S.; Manco G.; Masciari E.; Pontieri L.; and Pugliese A., *Detecting Structural Similarities Between XML Documents.* Proc. of the Inter. ACM SIGMOD Workshop on The Web and Databases (WebDB), 2002, 55-60.

[24] Formica A. and Missikoff M., *Concept Similarity in SymOntos: An Enterprise Ontology Management Tool.* The Computer Journal, 2002. 45(6), pp. 583-594.

[25] Francis W. N. and Kucera H., *Frequency Analysis of English Usage.* Houghton Mifflin, Boston, 1982.

[26] Fuhr N. and Großjohann K., *XIRQL: A Query Language for Information Retrieval.* Proc. of the ACM-SIGIR Conference, 2001, pp. 172-180.

[27] Gower J. C. and Ross G. J. S., *Minimum Spanning Trees and Single Linkage Cluster Analysis.* Applied Statistics, 18, 1969, pp. 54-64.

[28] Grabs T. and Schek H.-J., *Generating Vector Spaces On-the-fly for Flexible XML Retrieval.* ACM SIGIR Workshop on XML and IR, 2002. pp.4-13.

[29] Guha S.; Jagadish H.V.; Koudas N.; Srivastava D.; and Yu T., *Approximate XML Joins.* Proc. of the Inter. ACM SIGMOD Conf., 2002. pp. 287-298.

[30] Halkidi M.;Batistakis Y. and Vazirgiannis M., *Clustering Algorithms and Validity Measures.* Inter. Conf. on Scientific and Statistical Database Management (SSDBM), 2001. pp 3-22.

[31] Helmer S., *Measuring the Structural Similarity of Semistructured Documents Using Entropy.* The Inter. VLDB Conf., 2007, 1022-1032.

[32] Hopfield J. and Tank D., *Neural Computation of Decisions in Optimization Problems.* Biological Cybernetics, 1985, 52(3):52–141.

[33] Jiang J. and Conrath D., *Semantic Similarity based on Corpus Statistics and Lexical Taxonomy.* Proc. of the Inter. Conf. on Research in Computational Linguistics, 1997.

[34] Leacock C. and Chodorow M., *Combining Local Context and WordNet Similarity for Word Sense Identification.* WordNet: An Electronic Lexical Database, Ch. 11, The MIT Press, Cambridge, 1998. pp. 265-263.

[35] 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.

[36] Leitao L.; Calado P. and Weis M., *Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection.* Proc. of the ACM Conf. on Information and Knowledge Management (CIKM), 2007. pp. 293-302.

[37] Levenshtein V., *Binary Codes Capable of Correcting Deletions, Insertions and Reversals.* Sov. Phys. Dokl., 1966. (6):707-710.

[38] Lian W. *et al.*, *An Efficient and Scalable Algorithm for Clustering XML Documents by Structure.* IEEE TKDE, 2004. 16(1):82-96.

[39] Liang W. and Yokota H., *SLAX: An Improved Leaf-Clustering Based Approximate XML Join Algorithm for Integrating XML Data at Subtree Classes.* Trans. of Information Processing Society of Japan, 2006. (47):47-57.

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

[41] Lin D., *An Information-Theoretic Definition of Similarity.* Proc. of the Inter. Conf. on Machine Learning (ICML), 1998. pp. 296-304. Morgan Kaufmann.

[42] Ma Y. and Chbeir R., *Content and Structure Based Approach for XML Similarity.* Inter. Conf. on Computer and Info. Tech. (ICCIT), 2005, 136-140.

[43] Madhavan J.; Bernstein P.; and Rahm E., *Generic Schema Matching With Cupid.* Proce. of the Inter. VLDB Conf., 2001. pp. 49-58.

[44] Maguitman A.; Menczer F.; Roinestad H.; and Vespignani A., *Algorithmic Detection of Semantic Similarity.* The Inter. WWW Conf., 2005. pp. 107-116.

[45] Marie A. and Gal A., *Boosting Schema Matchers.* In Proc. of the OTM 2008 Confederated Inter. Conferences, 2008, 283 – 300.

[46] Martino B. D., *Semantic web services discovery based on structural ontology matching.* International Journal of Web and Grid Services, 2009. 5(1):46–65.

[47] McGill M., *Introduction to Modern Information Retrieval.* 1983. McGraw-Hill, New York.

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

[49] Ming M.; Yefei P. and Michael S., *A Harmony Based Adaptive Ontology Mapping Approach.* In Proc. of the Inter. Conf. on Semantic Web and Web Services (SWWS'08), 2008, 336-342.

[50] Moving Pictures Experts Group. *MPEG-7.* [cited Jan 2011] http://www.chiariglione.org/mpeg/standards/mpeg-7/.

[51] Muthaiyah S. and L. Kerschberg, *A Hybrid Ontology Mediation Approach for the Semantic Web.* J. of E-Business Research (IJEBR), 2008. 4(4): 79-91.

[52] Muthaiyah S.; Barbulescu M. and L. Kerschberg, *A Hybrid Similarity Matching Algorithm for Mapping and Upgrading Ontologies via a Multi-Agent System.* Proc. of the WSEAS Inter. Conf. on Computers (CSCC), 2008.

[53] Myers E., *An O(ND) Difference Algorithm and Its Variations.* Algorithmica, 1986. 1(2):251-266.

[54] Navigli R. and Lapata M., *Graph Connectivity Measures for Unsupervised Word Sense Disambiguation.* In Proc. of the Inter. Joint Conf. on Artificial Intelligence (IJCAI), 2007, 1683-1688.

[55] Nierman A. and Jagadish H. V., *Evaluating structural similarity in XML documents.* Proc. of ACM SIGMOD WebDB, 2002, pp. 61-66.

[56] Patwardhan S.; Banerjee S. and Pedersen T., *SenseRelate:TargetWord – A Generalized Framework forWord Sense Disambiguation.* Proc. of the National Conf. on Artificial intelligence, 2005. V. 4 (AAAI'05), 1692-1693.

[57] 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.

[58] Rafiei D. *et al.*, *Finding Syntactic Similarities between XML Documents.* Inter. Conf. on Database and Expert Systems Applications (DEXA), 2006, 512-516.

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

[60] 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.

[61] Richardson R. and Smeaton A., *Using WordNet in a Knowledge-based approach to information retrieval.* The BCS-IRSG Colloquium on IR, 1995.

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

[63] Salton G. and Buckley C., *Term-weighting approaches in automatic text retrieval.* Information Processing and Management, 1988. 24(5):513-523.

[64] 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.

[65] Schenkel R. *et al.*, *Semantic Similarity Search on Semistructured Data with the XXL Search Engine* Information Retrieval, 2005. (8):521-545.

[66] Schlieder T., *Similarity Search in XML Data Using Cost-based Query Transformations.* Proc. of ACM SIGMOD WebDB, 2001, pp. 19-24.

[67] 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.

[68] Schöning H., *Tamoni – A DBMS Designed for XML.* IEEE Inter. Conf. on Data Engineering (ICDE), 2001. pp. 149-154.

[69] Sebti A. and Barfroush A.A., *A New Word Sense Similarity Measure in Wordnet.* Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT), 2008. pp. 369-373.

[70] Shasha D. and Zhang K., *Approximate Tree Pattern Matching.* Pattern Matching in Strings, Trees and Arrays, Oxford Univ. Press, 1995. Ch. 14.

[71] Slimani T.; Ben Yaghlane B. and Mellouli K., *A New Similarity Measure based on Edge Counting.* Proc. of World Academy of Science, Engineering and Technology, Vol. 17, 2006. pp. 34-38.

[72] Tai K., *The Tree-to-Tree correction problem.* Journal of the ACM, 1979. (26):422-433.

[73] Tekli J.; Chbeir R. and Yetongnon K., *Semantic and Structure Based XML Similarity: An Integrated Approach.* Proc. of the Inter. Conf. on Management of Data (COMAD), 2006, 32-43.

[74] Tekli J.; Chbeir R. and Yetongnon K., *Efficient XML Structural Similarity Detection using Sub-tree Commonalities.* Proc. of the Brazilian Symp. on Databases (SBBD) and SIGMOD DiSC, (Best paper), 2007, 116-130.

[75] Tekli J.; Chbeir R. and Yetongnon K., *XML Document Comparison: Appendix.* Technical Report - LE2I CNRS Laboratory, Univ. of Bourgogne, 2010. http://www.u-bourgogne.fr/Dbconf/XS3/XMLDocComparisonAppendix.pdf.

[76] Tekli J.; Chbeir R. and Yétongnon K., *A Fine-grained XML Structural Comparison Approach.* Proc. of the Inter. Conf. on Conceptual Modeling (ER), 2007. LNCS 4801, pp. 582-598.

[77] 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.

[78] Theobald A. and Weikum G., *Adding Relevance to XML.* Proc. of the Inter. ACM SIGMOD WebDB workshop, 2000. pp. 105-124.

[79] Theobald M.; Schenkel R. and Weikum G., *Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data.* In Proc. of the Inter. ACM SIGMOD WebDB workshop, 2003. pp. 1-6.

[80] Tversky, *Features of Similarity.* Psychological Review, 1977. 84(4):327-352.

[81] Wagner J. and Fisher M., *The String-to-String correction problem.* Journal of the ACM, 1974. 21(1):168-173.

[82] Web 3D. *X3D*. http://www.web3d.org/x3d/. [cited 27 May 2009].

[83] Weis M. and Naumann F., *Dogmatix Tracks down duplicates in XML.* ACM Inter. Conf. on Management of Data (SIGMOD), 2005, 431-442.

[84] Wong C. and Chandra A., *Bounds for the String Editing Problem.* Journal of the ACM, 1976. 23(1):13-16.

[85] World Wide Web Consortium. *The Document Object Model*. http://www.w3.org/DOM [cited 28 Jan 2011].

[86] World Wide Web Consortium. *Scalable Vector Graphics (SVG)*. http://www.w3.org/Graphics/SVG/. [cited 26 Jan 2011].

[87] Wu Z. and Palmer M., *Verb Semantics and Lexical Selection.* 32nd Annual Meeting of the Associations of Computational Linguistics, 1994. pp. 133-138.

[88] Yaworsky D., *Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora.* Proc. of the Inter. Conf.on Computational Linguistics (Coling), 1992. Vol 2, pp. 454-460. Nantes.

[89] 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.

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

Acknowledgement

The first author is partly funded by the Research Support Foundation of the State of Sao Paulo (FAPESP), post-doctoral fellowship n# 2010/00330-2.

Appendix - *Computation Examples*

In the following, we present two computation examples. The first shows how our approach considers *structural commonalities* in comparing XML trees. The second focuses on *semantic resemblances* between sub-trees. Similarity results for all XML motivation examples mentioned in Section 3 are reported and discussed subsequently.

I. Structural Similarity Evaluation

In this example, we consider the case of dummy XML document trees *A*, *D* and *E* in Fig. A. 1 (reported from Fig. 4 of the main paper). Recall that trees *D* and *E* are considered identical with respect to *A* following current approaches, i.e., [12, 16, 55], despite the fact that trees *A/D* share more structural similarities than *A/E* (as discussed in Section 3.1).

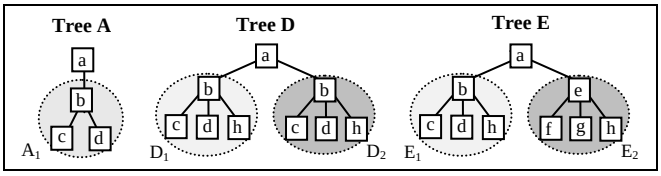


Fig. A. 1. XML trees *A*, *D* and *E* reported from Fig. 4.

In order to compare trees *A*/*D*, we start by executing algorithm *TOC* which computes operations costs. Note that in this example, parameter α is set to *1* since we only focus on XML structural commonalities. In fact, node labels in trees *A*, *D* and *E* are made of simple characters and have no semantic meanings. Thus, it would be useless to consider *Sem_RBS* in this case, which would obviously return null results.

$Cost_{Upd}(R(A), R(D)) = 0$, where $R(A).\ell = R(D).\ell =$ *'a'*

$$Cost_{DelTree}(A_1) = \sum_{\text{All nodes } x \text{ of } A_1} Cost_{Del}(x) \times \frac{1}{1 + Struct\_CBS(A_1, D_1)}$$

$$= 3 \times \frac{1}{1+0.75} = 1.7143$$

Likewise, $Cost_{InsTree}(D_1) = Cost_{InsTree}(D_2) = 4 \times \frac{1}{1 + 0.75} = 2.2856$

Related *Struct-CBS* computations are provided in Section 4.2 of the main paper.

Thus, when applied to XML trees *A* and *D*, with $\alpha=1$, our approach yields *TED*(*A*, *D*) = *3.2856* (cf. Table A.1).

Table A.1. Computing *TED* between XML trees *A* and *D*.

	R(D)	D_1	D_2
R(A)	0	2.2856	4.5712
A_1	1.7143	1	3.2856

- Dist[0][0] = $Cost_{Upd}(R(A), R(D)) = 0$, $R(A).\ell = R(D).\ell =$ *'a'*.
- Dist[1][1] = 1, cost of transforming sub-tree A_1 to D_1 (inserting node h).
- TED(A, D) = Dist[1][2] = 2.2856 + Dist[1][1] = 3.2856, inserting sub-tree D_2 into tree A.

When applied to XML trees *A* and *E*, with α=1, our approach yields *TED*(*A*, *E*) = *5*, which amounts to the costs of:

- Inserting node *h*, which is of maximum unit cost (=1) since *h* does share similarities with *A*,
- Inserting sub-tree E_2, which is of maximum cost (=4) since E_2 does not share any structural similarities with *A* (cf. Table A.2).

Table A.2. Computing TED between XML trees *A* and *E*.

	R(E)	E_1	E_2
R(A)	0	2.2856	6.2856
A_1	1.7143	1	5

- Dist[1][1] = 1, transforming A_1 into E_1 (inserting node h).
- Dist[1][2] = 4 + Dist[1][1] = 5, cost of inserting sub-tree E_2 into tree A.

Therefore, our approach is able to effectively compare XML document trees *A*, *D* and *E*, underlining that document trees *A/D* are more similar than *A/E* (pointing out structural similarities that are not detected via existing approaches):

- $Sim_{XDoc}(A, D) = 1 - \frac{TED(A, D)}{|A| + |D|} = 1 - \frac{3.2836}{13} = 0.7474$
- $Sim_{XDoc}(A, E) = 1 - \frac{TED(A, E)}{|A| + |E|} = 1 - \frac{5}{13} = 0.6154$

Similarly to the case of XML trees *A*, *D* and *E*, our approach detects the various kinds of XML tree structural similarities identified in our motivation examples in Section 3.1 (results are reported in Table 2 of the main paper).

II. Integrating Semantic Similarity Evaluation

In this computation example, we consider the case of XML trees *A'*, *B'* and *C'* in Fig. A. 2 (reported from Fig. 6 of the main paper). As discussed in motivation Section 3.2, trees *B'* and *C'* are structurally indistinguishable with respect to *A'* since they have different node labels. Yet, one can realize that *A'/B'* share more semantic similarities than *A'/C'* (similarities between sub-tree node labels *Academy*/*College*, *Professor*/*Lecturer*, and *PhD Student*/*Scholar*, as discussed previously).

Note that in this example, parameter α is set to *0* since we focus on sub-tree semantic resemblances. In fact, for the *A'*, *B'*, *C'* case, it is useless to consider *Struct_CBS* since the considered trees/sub-trees do not share structural similarities. In other words, *Struct_CBS* would yield zero values (recall that XML structure underlines the structural disposition and ordering of element/attribute tag labels. Hence, label disparities induce minimum structural similarity), which led us to maximize the weight of *Sem_RBS*.

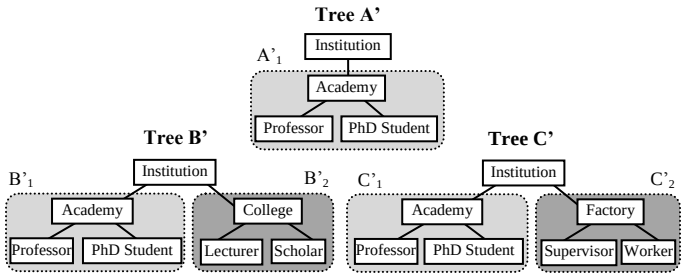


Fig. A. 2. XML trees *A'*, *B'* and *C'* reported from Fig. 6.

$Cost_{Upd}(R(A'), R(B')) = 0$, since $R(A').\ell = R(B').\ell =$ *'Institution'*

$$Cost_{DelTree}(A'_1) = \sum_{\text{All nodes } x \text{ of } A'_1} Cost_{Del}(x) \times \frac{1}{1+Sem\_RBS(A'_1, B'_1)} = \frac{3}{1+1} = 1.5$$

Likewise, $Cost_{InsTree}(B'_1) = 1.5$ (since A'_1 and B'_1 are identical).

$$Cost_{InsTree}(B'_2) = \sum_{\text{All nodes } x \text{ of } B'_2} Cost_{Del}(x) \times \frac{1}{1+Sem\_RBS(A'_1, B'_2)} = \frac{3}{1+0.9753} = 1.5188$$

Related *Sem_RBS* computations are provided in Section 4.3 of the main paper.

Table A.3. Computing TED between trees *A'* and *B'*.

	R(B')	B'_1	B'_2
R(A')	0	1.5	3.0188
A'_1	1.5	0	1.5188

Table A.4. Computing TED between trees *A'* and *C'*.

	R(C')	C'_1	C'_2
R(A')	0	1.5	3.4403
A'_1	1.5	0	1.9604

Yet, when applied to trees A' and C' (α = *0*), our approach yields *Dist(A', C')=1.9167*:

$Cost_{Upd}(R(A'), R(C')) = 0$, since $R(A').\ell = R(C').\ell =$ *'Institution'*

$Cost_{DelTree}(A'_1) = Cost_{InsTree}(C'_1) = 1.5$ (since sub-trees A'_1 and C'_1 are identical).

$$Cost_{InsTree}(C'_2) = \sum_{\text{All nodes } x \text{ of } C'_2} Cost_{Del}(x) \times \frac{1}{1+Sem\_RBS(A'_1, C'_2)} = \frac{3}{1+0.5303} = 1.9604$$

Related *Sem_RBS* computations are provided in Section 4.3 of the main paper.

Therefore, our approach is able to efficiently compare XML documents *A'*, *B'* and *C'* underlining that documents *A'/B'* are more similar than *A'/C'* (pointing out semantic similarities that are disregarded via existing approaches):

- $Sim_{XDoc}(A', B') = 1 - \frac{TED(A', B')}{|A'| + |B'|} = 1 - \frac{1.5189}{11} = 0.8619$
- $Sim_{XDoc}(A', C') = 1 - \frac{TED(A', C')}{|A'| + |C'|} = 1 - \frac{1.9604}{11} = 0.8218$

Results for all motivation examples discussed throughout the paper are reported in Table 3 of the main manuscript.