Building Semantic Trees from XML Documents*

Joe Tekli, Nathalie Charbel, and Richard Chbeir

Abstract. The distributed nature of the Web, as a decentralized system exchanging information between heterogeneous sources, has underlined the need to manage *interoperability*, i.e., the ability to automatically interpret information in Web documents exchanged between different sources, necessary for efficient information management and search applications. In this context, XML was introduced as a data representation standard that simplifies the tasks of interoperation and integration among heterogeneous data sources, allowing to represent data in (semi-) structured documents consisting of hierarchically nested elements and atomic attributes. However, while XML was shown most effective in exchanging data, i.e., in *syntactic interoperability*, it has been proven limited when it comes to handling semantics, i.e., *semantic interoperability*, since it only specifies the syntactic and structural properties of the data without any further semantic meaning. As a result, XML semantic-aware processing has become a motivating challenge in Web data management, requiring dedicated semantic analysis and disambiguation methods to assign well-defined meaning to XML elements and attributes. In this context, most existing approaches: i) ignore the problem of identifying ambiguous XML elements/nodes, ii) only partially consider their structural relationships/context, iii) use syntactic information in processing XML data regardless of the semantics involved, and iv) are static in adopting fixed disambiguation constraints thus limiting user involvement. In this paper, we provide a new XML Semantic Disambiguation Framework titled *XSDF* designed to address each of the above limitations, taking as input: an XML document, and then producing as output a semantically augmented XML tree made of unambiguous semantic concepts extracted from a reference machine-readable semantic network. *XSDF* consists of four main modules for: i) linguistic pre-processing of simple/compound XML node labels and values, ii) selecting ambiguous XML nodes as targets for disambiguation, iii) representing target nodes as special *sphere neighborhood* vectors including all XML structural relationships within a (user-chosen) range, and iv) running context vectors through a hybrid disambiguation process, combining two approaches: *concept-based* and *context-based* disambiguation, allowing the user to tune disambiguation parameters following her needs. Conducted experiments demonstrate the effectiveness and efficiency of our approach in comparison with alternative methods. We also discuss some practical applications of our method, ranging over semantic-aware query rewriting, semantic document clustering and classification, Mobile and Web services search and discovery, as well as blog analysis and event detection in social networks and tweets.

***Keywords*:** XML, Semi-structured data, Word sense disambiguation, Semantic-aware processing, Semantic ambiguity, Context representation, Semantic similarity, Knowledge bases.

1. Introduction

Over the past decade, publishing and processing data in XML has become increasingly attractive for organizations that want to easily inter-operate and provide their information in a well-defined, semi-structured, extensible, and machine-readable format to improve the quality of their Web-based information retrieval and data management applications [54]. Most approaches in this context use syntactic information in processing XML data, while ignoring the semantics involved [93]. In fact, even when the huge amount of raw information available to organizations exists in natural language form and needs to be automatically processed, it can be first distilled into a more structured form in which individual entities (pieces of data) are accessible. Here, information extraction and wrapping technologies can be used: extracting and structuring selected data from documents to make them easier to handle in enterprise applications, where the preferred output is largely XML (e.g., handling XML-based summaries of news headlines, legal decisions, research articles, medical reports, editorials, book reviews, etc.) [92]. However, attaining a higher degree of automated data processing capability and human-machine cooperation requires yet another breakthrough: extracting and processing the semantic features of XML data, whose impact has been highlighted in various XML-based applications, ranging over: semantic-aware query rewriting and expansion [21, 68] (expanding keyword queries by including semantically related terms from XML documents to obtain relevant results), XML document classification and clustering [94, 101] (grouping together documents based on their semantic similarities, rather than performing syntactic-only processing), XML schema matching and integration [24, 103] (considering the semantic meanings and relationships between schema elements and data-types), and more recently Web and mobile services' discovery, recommendation, and composition [46, 56, 115] (searching and mapping together semantically similar WSDL/SOAP descriptions when processing Web services, and semantic-aware mapping of XHTML/free-text descriptions when dealing with RESTful and/or mobile services), as well as XML-based semantic blog analysis and event detection in social networks and tweets [3, 9, 81]. Here, a major challenge remains unresolved: *XML semantic disambiguation*, i.e., how to resolve the semantic ambiguities and identify the meanings of terms in XML documents [43], which is central to improve the

* Work supported in part by the National Council for Scientific Research (CNRS), Lebanon, project: *NCSR00695_01/09/15*, and by LAU grant: *SOERC1415T004.*

- *Joe Tekli is with the Electrical and Computer Engineering Dept., Lebanese American University, 36 Byblos, Lebanon. Email: joe.tekli@lau.edu.lb*
- *Nathalie Charbel is with the LIUPPA Lab., University of Pau and Adour Countries, 64600 Anglet, France. Email: nathalie.charbel@univ-pau.fr*
- *Richard Chbeir is with the LIUPPA Lab., University of Pau and Adour Countries, 64600 Anglet, France. Email: richard.chbeir@univ-pau.fr*

performance of XML-based applications. The problem is made harder with the volume and diversity of XML data on the Web.

Usually, heterogeneous XML data sources exhibit different ways to annotate similar (or identical) data, where the same real world entity could be described in XML using different structures and/or tagging, depending on the data source at hand (as shown in Fig. 1, where two different XML documents describe the same *Hitchcock movie*). The core problem here is lexical ambiguity: a term (e.g., an XML element/attribute tag name or data value) may have multiple meanings (polysemy), it may be implied by other related terms (metonymy), and/or several terms can have the same meaning (synonymy) [43]. For instance (according to a general purpose knowledge base such as WordNet [28]), the term *"Kelly"* in XML document 1 of Fig. 1 may refer to *Emmet Kelly: the circus clown*, *Grace Kelly: Princess of Monaco*, or *Gene Kelly: the dancer*. However, looking at its context in the document, a human user can tell that *"Kelly"* here refers to *Grace Kelly*. Yet while seemingly obvious for humans, such semantic ambiguities remain extremely complex to resolve with automated processes.

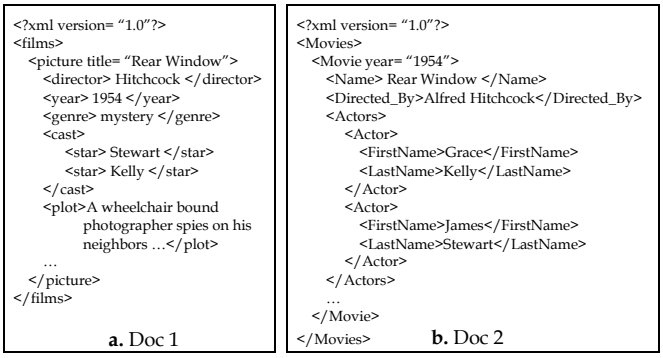


Fig. 1. Sample documents with different structures and tagging, yet describing the same information.

In this context, *word sense disambiguation* (WSD), i.e., the computational identification of the meaning of words in *context* [67], could be central to automatically resolve the semantic ambiguities and identify the meanings of XML element/attribute tag names and data values, in order to effectively process XML documents. While WSD has been widely studied for flat textual data [40, 67], yet, the disambiguation of structured XML data remains largely untouched. The few existing approaches to XML semantic-aware analysis (Section 2) have been extended from traditional flat text WSD, and thus show several limitations, motivating this work:

- **Motivation 1:** They completely ignore the problem of *semantic ambiguity*, which in turn leads to questions of disambiguation *exhaustivity* versus *selectivity*. To our knowledge, none of the existing XML-based approaches addresses the problem of selecting target nodes to be processed for disambiguation, which should inherently be the most *ambiguous* nodes in the XML document (similarly to selecting the most ambiguous words as targets in flat text WSD [34, 75]). Rather, they perform semantic disambiguation on all nodes of the XML document which is exhaustively time consuming and sometimes needless (e.g., no need to disambiguate unambiguous nodes). Here, the main difficulty resides in identifying/selecting *ambiguous* nodes (words) which need to be processed for disambiguation, since there is no formal method for evaluating XML node (word) *ambiguity.*
- **Motivation 2:** They only partially consider the structural relationships/context of XML nodes (e.g., solely focusing on parent-node relationships [98], or ancestor-descendent relationships [95]). For instance, in Fig. 1, processing XML node *"cast"* for disambiguation: considering (exclusively) its parent node label (i.e., *"picture"*), its root node path labels (i.e., *"films"* and *"picture"*), or its node sub-tree labels (i.e., *"star"*), remains insufficient for effective disambiguation.
- **Motivation 3:** They make use of syntactic processing techniques such as the *bag-of-words* paradigm [94, 98] (commonly used with flat text) in representing XML data as a plain set of words/nodes, thus neglecting XML structural and/or semantic features as well as compound node labels.
- **Motivation 4:** They are mostly static in adopting a fixed context size (e.g., parent node [98], or root path [95]) or using preselected semantic similarity measures (e.g., edge-based measure [57], or gloss-based measure [95]), such that the user's ability in tuning disambiguation parameters to allow system adaptability (following her needs) is minimal.

The main goal of our study is to provide an effective method to XML semantic analysis and disambiguation, overcoming the limitations mentioned above. We aim to transform traditional *syntactic XML trees* into *semantic XML trees* (or graphs, when hyperlinks come to play), i.e., XML trees made of concept nodes with explicit semantic meanings. Each concept will represent a unique lexical sense, assigned to one or more XML element/attribute labels and/or data values in the XML document, following the latter's structural context. To do so, we introduce a novel XML Semantic Disambiguation Framework titled *XSDF*, a fully automated solution to semantically augment XML documents using a machine-readable semantic network (e.g., WordNet [28], Roget's thesaurus [117], Yago [37], an adaptation of FOAF [3][1], etc.), identifying the semantic definitions and relationships among concepts in the underlying XML structure. Different from existing approaches, *XSDF* consists of four main modules for: i) linguistic pre-processing of XML node labels and values to handle compound words (neglected in most existing solutions), ii) selecting ambiguous XML nodes as primary targets for disambiguation using a dedicated *ambiguity degree* measure (unaddressed in existing solutions), iii) representing target nodes as special *sphere neighborhood* vectors considering a comprehensive XML structure context including all XML structural relationships within a (user-chosen) range (in contrast with partial context representations using the *bag-of-words* paradigm), and iv) running *sphere neighborhood* vectors through a hybrid disambiguation process, combining two approaches: *concept-based* and *context-based* disambiguation, allowing the user to tune disambiguation parameters following her needs (in contrast with static methods). We have implemented *XSDF* to test and evaluate our approach. Experimental results reflect our approach's effectiveness in comparison with existing solutions.

The overall architecture of *XSDF* has been introduced in [18]. This paper adds: i) an extended presentation of *XSDF*'s mathematical model for ambiguity degree and disambiguation

[1] A semantic network structure can be automatically generated based on the FOAF social network, where person profiles represent concepts, and links between profiles represent concept relationships. Such a semantic network can be used for person recognition and/or identification (as a special case of *named entity recognition* or *disambiguation* [67]).

computations, with corresponding proofs, ii) an adapted gloss-based semantic similarity measure, expanded and normalized to be used in concept-based disambiguation, iii) dedicated algorithms to perform concept-based and context-based semantic disambiguation, along with detailed complexity analysis, iv) a detailed presentation of experimental results, v) an extended discussion of the state of the art solutions, as well as vi) a discussion of the main applications scenarios which can benefit, in one way or another, from XML semantic analysis and disambiguation.

The remainder of this paper is organized as follows. Section 2 reviews the background and related works. Section 3 develops our XML disambiguation framework. Section 4 presents experimental results. Section 5 discusses potential applications, before concluding the paper in Section 6 with current directions.

2. Background and Related Works

First, Section 2.1 briefly describes traditional word sense disambiguation developed for flat text. Then, Section 2.2 covers XML (semi-structured) semantic disambiguation methods.

2.1. Word Sense Disambiguation

WSD underlines the process of computationally identifying the senses (meanings) of words in context, to discover the author's intended meaning [40]. The general WSD task consists of the following main elements: i) selecting words for disambiguation, ii) identifying and representing word contexts, iii) using reference knowledge sources, iv) associating senses with words, and v) evaluating semantic similarity between senses.

2.1.1. Selecting Words for Disambiguation

There are two possible methods to select target words for disambiguation: i) *all-words*, or ii) *lexical-sample*. In all-words WSD, e.g., [17, 75], the system is expected to disambiguate all words in a (flat) textual document Although considered as a complete and exhaustive disambiguation approach, yet it remains extremely time-consuming and labor intensive. In addition, the high (time and processing) costs might not match performance expectations after all [67]. In lexical-sample WSD, e.g., [34, 75], specific target words are selected for disambiguation (usually one word per sentence). These words are often the most ambiguous, and are usually chosen using supervised learning methods trained to recognize salient words in sentences [67]. Experimental results reported in [67] show high disambiguation accuracy using the lexical-sample approach, in comparison with the all-words approach. However, a major difficulty in adopting the lexical-sample approach is in selecting ambiguous (target) words, due to the lack of formal methods to quantify *semantic ambiguity*, given that current supervised learning approaches are time-consuming including a training phase requiring training data which are not always available.

2.1.2. Identifying and Representing Context

Once words have been selected for disambiguation, their contexts have to be identified, since sense disambiguation relies on the notion of context, such as words that appear together in the same context usually have related meanings [49]. The *context* of a word in traditional flat textual data usually consists of the set of terms in the word's vicinity, i.e., terms occurring to the left and right of the considered word, within a certain predefined window size [49]. Other features can also be used to describe context, such as information resulting from linguistic pre-processing including part-of-speech tags (e.g., verb, subject, etc.), grammatical relations, etc. [67]. Once the context has been identified, it has to be effectively represented to perform disambiguation computations. Here, the traditional *bag-of-words* paradigm is broadly adopted with flat textual data [40, 67], where the context is processed as a set of terms surrounding the word to disambiguate. A vector representation considering the number of occurrences of words in context can also be used [67]. More structured context representations have been investigated in [2, 110], using co-occurrence graphs. Yet, they require substantial additional processing than the *bag-of-words* model.

2.1.3. Using Reference Knowledge Sources

In addition to the contexts of target words, external knowledge is essential to perform WSD, providing reference data which are needed to associate senses with words. In this context, WSD methods can be distinguished as *corpus-based* or *knowledge-based*, depending on the kind of external knowledge sources they rely on. The *corpus-based* approach, e.g., [5, 6, 21], is data-driven, as it involves information about words previously disambiguated, and requires supervised learning from sense-tagged corpora (e.g., SemCor [65], where each word/expression is associated an explicit semantic meaning) in order to enable predictions for new words. *Knowledge-based* methods, e.g., [64, 67, 94], are knowledge-driven, as they handle a structured sense inventory and/or a repository of information about words that can be automatically exploited to distinguish their meanings in the text. Machine-readable knowledge bases (dictionaries, thesauri, and/or lexical ontologies, such as WordNet [28], Roget's thesaurus [117], ODP [54], etc.) provide ready-made sources of information about word senses to be exploited in knowledge-based WSD. While *corpus-based* methods have been popular in recent years, e.g., [5, 6, 21], they are generally data hungry and require extensive training, huge textual corpora, and/or a considerable amount of manual effort to produce a relevant sense-annotated corpus, which are not always available and/or feasible in practice. Therefore, knowledge-based methods have been receiving more attention lately, e.g., [64, 67, 94], and include most solutions targeting XML data.

2.1.4. Associating Senses with Words

The final step in WSD is to associate senses with words, taking into account the target words' contexts as well as reference external knowledge about word senses. This is usually viewed as a word-sense classification task. In this regard, WSD approaches can be roughly categorized as *supervised* or *unsupervised*. On one hand, supervised methods, e.g., [61, 67, 107], involve the use of machine-learning techniques, using samples (a human expert manually annotates examples of a word with the intended sense in context) provided as training data for a learning algorithm that induces rules to be used for assigning meanings to other occurrences of the word. External knowledge (mainly corpus-based) is used and combined with the human expert's own knowledge of word senses when manually tagging the training examples. While effective, yet supervised methods include a learning phase which is highly time-consuming, and requires a reliable training set with a wide coverage which is not always available.

On the other hand, unsupervised methods, e.g., [57, 71, 93], are usually fully automated and do not require any human intervention or training phase. Most recent (and XML-related) approaches, e.g., [58, 71, 93], make use of a machine-readable knowledge base (e.g., such as WordNet [28]) represented and processed as a semantic network made of a set of concepts representing word senses, and a set of links connecting the concepts, representing semantic relations (synonymy, hyponymy, etc., [28, 79], cf. Fig. 2). Given a target word to be disambiguated, WSD consists in identifying the semantic concept (word sense), in the reference semantic network that best matches the semantic concepts (word senses) of terms appearing in the context of the target word. Semantic concept matching is usually performed using a measure of semantic similarity between concepts in the reference semantic network, also known as *knowledge-based* semantic similarity [14, 70].

2.1.5. Semantic similarity measures in a semantic network

Several methods have been proposed to determine semantic similarity between concepts (and consequently related terms) in a knowledge base (semantic network). These can be organized in three main categories [14]: *edge-based*, *node-based*, and *gloss-based* [14]. Edge-based methods [47, 114] estimate similarity as the shortest path (in edges, weights, or number of nodes) between the concepts being compared. A central edge-based measure (adopted in our study) is proposed by Wu and Palmer in [114]:

$$\mathrm{Sim}_{\mathrm{Edge}}(c_1, c_2, SN) = \frac{2N_0}{N_1 + N_2 + 2N_0} \in [0,1] \quad \textbf{(1)}$$

where c_1 and c_2 designate two semantic concepts (word senses), *SN* designates the reference semantic network, N_1 and N_2 are respectively the lengths of the paths separating concepts c_1 and c_2 from their lowest common ancestor concept c_0 in *SN*, and N_0 is the length of the path separating concept c_0 from the root of *SN*.

Node-based approaches [50, 77] estimate similarity as the maximum amount of information content that concepts share in common, based on the statistical distribution of concept (term) occurrences in a text corpus (e.g., the Brown corpus [29]). A central node-based measure (adopted in our study) is proposed by Lin in [50]:

$$\mathrm{Sim}_{\mathrm{Node}}(c_1, c_2, SN) = \frac{2 \log p(c_0)}{\log p(c_1) + \log p(c_2)} \in [0,1] \quad \text{having} \quad p(c_i) = \frac{Freq(c_i)}{N} \quad \textbf{(2)}$$

where c_0, c_1, and c_2 are the same as in formula (1), $p(c_i)$ denotes the occurrence probability of concept c_i designating the normalized frequency of occurrence of c_i in a reference corpus such as the Brown text corpus [29] (adopted in our study), and *N* designates the size (total number of words) in the reference corpus.

Gloss-based methods [7, 8] evaluate word overlap between the glosses of concepts being compared, a gloss underlining the textual definition describing a concept (e.g., the gloss of the 1st sense of word *Actor* in WordNet is *"A theatrical performer"*, cf. Fig. 1). A central gloss-based measure (extended in our study, cf. Section 3.5.1) is proposed by Banerjee and Pedersen in [8]:

$$\mathrm{Sim}_{\mathrm{Gloss}}(c_1, c_2, SN) = \left(\mathrm{gloss}(c_1) \cup \mathrm{gloss}(rel(c_1))\right) \cap \left(\mathrm{gloss}(c_2) \cup \mathrm{gloss}(rel(c_2))\right) \quad \textbf{(3)}$$

where $gloss(c_i)$ is the bag of words in the textual definition of concept (word sense) c_i, and $rel(c_i)$ is the set of concepts related to c_i through a semantic relation in *SN*.

Discussion: It has been shown that *gloss-based* measures evaluate, not only *semantic similarity*, but also *semantic relatedness* [71], which is a more general notion: including similarity as well as any kind of functional relationship between terms (e.g., *"penguin"* and *"Antarctica"* are not similar, but they are semantically related due to their *natural habitat* connection), namely antonymy (e.g., *"hot"* and *"cold"* are dissimilar: having opposite meanings, yet they are semantically related), which makes *gloss-based* measures specifically effective in WSD [67]: matching not only similar concepts, but also semantic related ones[2].

Note that *unsupervised/knowledge-based* WSD has been largely investigated recently (including most methods targeting XML data), in comparison with *supervised* and *corpus-based* methods, which usually require extensive training and large test corpora [67], and thus do not seem practical for the Web. The reader can refer to [40, 67] for reviews on traditional WSD.

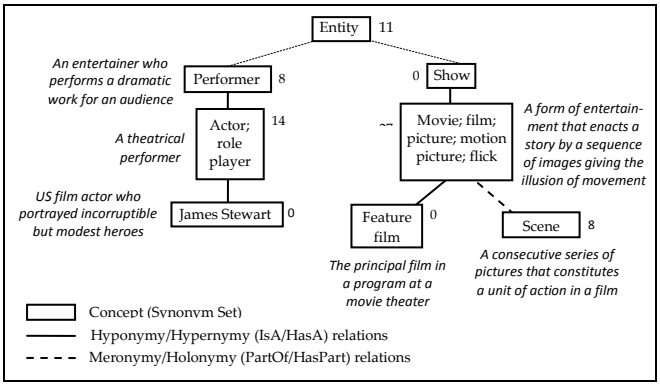


Fig. 2. Extract of the WordNet semantic network. Numbers next to concepts represent concept frequencies (based on the Brown corpus [29]). Sentences next to concepts represent concept glosses.

2.2. XML Semantic Disambiguation

Few approaches have been developed for semantic disambiguation of XML and semi-structured data. The main challenges reside in the notion of XML (structural) contextualization and how it is processed, as described below.

2.2.1. XML Context Identification

While the context of a word in traditional flat textual data consists of the set of terms in the word's vicinity [49], yet there is no unified definition of the context of a node in an XML document tree. Various approaches have been investigated: i) *parent node*, ii) *root path*, iii) *sub-tree*, and iv) *versatile structural context*.

2.2.1.1. Parent Node Context

The authors in [97, 98] consider the context of an XML data element to be efficiently determined by its parent element, and thus process a parent node and its children data elements as one unified (canonical) entity. The approach is based on the observation that an XML data node (i.e., element/attribute value) constitutes by itself a semantically meaningless entity. They introduce the notion of canonical tree as a structure grouping together a leaf (data) node with its parent node, which is deemed as the simplest

[2] The reader can refer to [14, 84, 118] for comprehensive reviews and evaluations of semantic similarity measures.

semantically meaningful structural entity. The authors utilize dedicated context-driven search heuristics (e.g., structure pruning, identifying immediate relatives, etc.) to determine the relationships between the different canonical trees. These are exploited to assign semantic node labels using a manually built ontology [96], generalizing/specializing node concepts following ontology labels and the nodes' structural positions in the XML tree hierarchy.

2.2.1.2. Root Path Context

In [94, 95], the authors extend the notion of XML node context to include the whole XML root path, i.e., the path consisting of the sequence of nodes connecting a given node with the root of the XML document (or document collection). They perform per-path sense disambiguation, comparing every node label in each path with all possible senses of node labels occurring in the same path. Each XML path is transformed into a weighted graph, with nodes underlining the senses of each path element, and edges connecting node senses following path direction and node sense semantic similarities (Section 2.1.5). The authors utilize an existing gloss-based WordNet similarity measure [8] and introduce an edge-based measure (similar to [114]) in comparing label semantic senses to compute graph edge weights. Then, selecting the appropriate sense for a given node label consists in identifying the set of connected node senses, in the corresponding weighted graph, such as the sum of the weights over their edges is maximum.

2.2.1.3. Sub-tree Context

Different from the notions of parent context and path context, the authors in [106] consider the set of XML nodes contained in the sub-tree rooted at a given element node, i.e., the set of labels corresponding to the node at hand and all its subordinates, to describe the node's XML context. The authors apply a similar paradigm to identify to contexts of all possible node label senses in WordNet. As a result, both the target XML node (to be disambiguated), and each of its possible node label senses in the WordNet taxonomy are represented as sets of lexical words/ expressions. Consequently, XML node label sense disambiguation is performed by comparing the XML node context set to each of the candidate sense context sets, using a classic similarity measure between bags-of-words (the authors utilize Cosine, taking into account TF-IDF word frequencies in each of the set representations). The target XML node is finally mapped to the semantic sense such that their context sets have the highest similarity.

2.2.1.4. Versatile structural context

In [57, 58], the authors combine the notions of parent context and descendent (sub-tree) context in disambiguating generic structured data (e.g., XML, web directories, and ontologies). The authors consider that a node's context definition depends on the nature of the data and the application domain at hand. They propose various edge-weighting measures (namely a Gaussian decay function, cf. formula (4)) to identify *crossable* edges, such as nodes reachable from a given node through any *crossable* edge (following a user-specified direction, e.g., ancestor, descendent, or both) belong to the target node's context:

$$weight(n_c, n) = 2 \times \frac{e^{-\frac{d^2}{8}}}{\sqrt{2 \times \pi}} + 1 - \frac{2}{\sqrt{2 \times \pi}} \quad \textbf{(4)}$$

where n_c is a context node, n is the target node, and d is the distance (in number of edges) separating n_c form n in the XML tree. An extension of the traditional bag-of-words model is introduced to consider edge weights in representing the XML context. Hence, structure disambiguation is undertaken by comparing the target node label with each candidate sense (semantic concept) corresponding to the labels in the target node's context, taking into account corresponding XML edge weights. The authors assign confidence scores to each semantic sense following its order of appearance in the corresponding WordNet term definition. They consequently utilize an edge-based semantic similarity measure [45], exploiting the hypernymy/hyponymy relations (excluding remaining relations such as meronymy and holonymy, etc.), to identify the semantic sense best matching the target node label.

2.2.2. XML Context Representation and Processing

Another concern in XML-based WSD is how to effectively process the context of an XML node (once it has been identified) considering the structural positions of XML data. Most existing WSD methods - developed for flat textual data (Section 2.1) and/or XML-based data [94, 95, 97, 98] - adopt the *bag-of-words* model where the context is processed as a set of words surrounding the term/label (node) to be disambiguated. Hence, all context nodes are treated the same, despite their structural positions in the XML tree. One approach in [57, 58], identified as *relational information model*, extends the *bag-or-words* paradigm toward a vector-based representation with confidence scores combining: i) distance weights separating the context and target nodes, and ii) semantic weights highlighting the importance of each sense candidate. The authors utilize a specially tailored distance Gaussian decay function (cf. formula (4)) estimating edge weights such that the closer a node (following a user-specified direction), the more it influences the target node's disambiguation [57, 58]. The distance decay function is not only utilized in identifying the context of a target node, but also produces weight scores which are assigned to each context node in the context vector representation, highlighting the context node's impact on the target node's disambiguation process [57].

2.2.3. Associating Senses with XML Nodes

Once the contexts of XML nodes have been determined, they can be handled in different ways. Two approaches, both *unsupervised* and *knowledge-based*, have been adopted in this context, which we identify as: i) *concept-based* and ii) *context-based*. On one hand, the concept-based approach adopted in [94, 95] consists in evaluating the semantic similarity between target node senses (concepts) and those of its context nodes, using measures of semantic similarity between concepts in a semantic network. The target node label is matched with the candidate sense corresponding to the candidate combination having the maximum score. On the other hand, the context-based approach introduced in [106] consists in building a context vector for the target node in the XML document tree, and context vectors for each target node sense (concept) in the semantic network (SN). Then, the XML context set is compared with each of the SN context sets, using a typical set comparison measure (e.g., *Jaccard* similarity). After, the target node label is matched with the candidate sense having the SN context set with maximum score w.r.t. the XML target node context set. A hybrid method in [57, 58] combines variants of the two preceding approaches to disambiguate generic struc-

tured data (including XML). Yet, the authors do not compare their solution with existing XML disambiguation methods.

Wrapping up: we identify four major limitations motivating our work (which were highlighted in Section 1): most existing methods i) completely ignore the problem of *semantic ambiguity*, ii) only partially consider the structural relationships/context of XML nodes (e.g., parent-node [98] or ancestor-descendent relations [95]), ii) neglect XML structural/semantic features by using syntactic processing techniques such as the *bag-of-words* paradigm [94, 98], and iv) are static in choosing a fixed context (e.g., parent node [98], or root path [95]) or preselected semantic similarity measures, thus minimizing user involvement.

3. XML Disambiguation Framework

To address all motivations above and provide a more complete and dynamic XML disambiguation approach, we introduce *XSDF* (XML Semantic Disambiguation Framework) as an unsupervised and knowledge-based solution to resolve semantic ambiguities in XML documents. *XSDF*'s overall architecture is depicted in Fig. 3. It is made of four modules: i) linguistic pre-processing, ii) nodes selection for disambiguation, iii) context definition and representation, and iv) XML semantic disambiguation. The system receives as input: i) an XML document tree, ii) a semantic network (noted *SN*), and iii) user parameters (to tune the disambiguation process following her needs), and produces as output a semantic XML tree.

We develop *XSDF*'s modules in the following, starting with the XML and semantic data models adopted in our study.

3.1. XML and Semantic Data Models

XML documents represent hierarchically structured information and can be modeled as *rooted ordered labeled trees* (Fig. 4.a and b), based on the Document Object Model (DOM) [111].

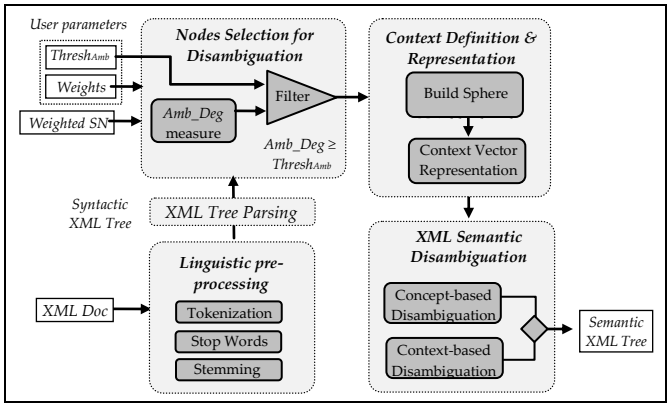


Fig. 3. Overall *XSDF* architecture.

Definition 1 – Rooted 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, $T[i].\ell$ its label, $T[i].d$ its depth (in number of edges), and $T[i].f$ its out-degree (i.e., the node's fan-out). $R(T)$= [0] designates the root node of tree T [3] ●

Definition 2 – XML Document Tree: It is a *rooted ordered labeled tree* where nodes represent XML elements/attributes, labeled using element/attribute tag names. Element nodes are ordered following their order of appearance in the XML document. Attribute nodes appear as children of their containing element nodes, sorted[4] by attribute name, and appearing before all sub-elements [69, 119]. Element/attribute text values are stemmed and decomposed into tokens (using our linguistic pre-processing component), mapping each token to a leaf node labeled with the respective token, appearing as a child of its container element/attribute node, and ordered following their appearance order in the element/attribute text value (Fig. 4.a) ●

Note that element/attribute values can be disregarded (*structure-only*) or considered (*structure-and-content*) in the XML document tree following the application scenario at hand. Here, we believe integrating XML structure and content is beneficiary in resolving ambiguities in both element/attribute tag names (structure) and data values (content). For instance, in Fig. 1.a, considering data values *"Kelly"* and *"Stewart"* would be beneficial to disambiguate tag label *"cast"*. The same applies the other way: *"cast"* can help disambiguate *"Kelly"* and *"Stewart"*.

Also, we formally define a semantic network, as the semantic (knowledge base) data model adopted in our study[5].

Definition 3 – Semantic network: It is made of concepts representing groups of words/expressions designating word senses, and links connecting the concepts designating semantic relationships. It can be represented as $SN=(C, L, G, E, R, f, g, h)$:

- C: set of nodes representing concepts in SN (e.g., *synsets* as in WordNet [28]),
- L: set of words (expressions) describing concept labels and synonyms,
- G: set of glosses describing concept definitions,
- E: set of edges connecting concept nodes, $E \subseteq C \times C$,
- R: set of semantic relationship names describing edge labels, i.e., R = {*Is-A*, *Has-A*, *Part-Of*, *Has-Part*, etc.}. Note that synonymous words/expressions are integrated in the concepts themselves (as concept labels), and thus the relationship (edge label) *synonym* does not exist in R.
- f: function designating the labels, sets of synonyms, and glosses associated to concept nodes, f: $C \rightarrow (L, L^n, G)$ where n designates the number of synonyms per concept,
- g: function designating the labels of edges, g: $E \rightarrow R$,
- h: function associating to concepts their occurrence frequencies (cf. Fig. 2) statistically quantified from a given text corpus (e.g., the Brown corpus [29])[6].

Note that $c \in C$ denotes a concept with $c.\ell$ its label, $c.syn$ its set of synonymous labels, $c.gloss$ its gloss, and $c.freq$ its frequency. Similarly, $e \in E$ ($\in SN$) designates an edge linking a pair of concepts, with $e.rel$ its edge label (semantic relationship name) ●

After disambiguating target XML node labels in the input XML document tree, the latter are replaced with the identifiers of corresponding semantic concepts from the reference semantic network, thus producing an output XML tree augmented with semantic meaning. Formally:

Definition 4 – Semantic XML Tree: It is an augmented *XML document tree* in which the labels of nodes which have been

[3] *Tree* and *rooted ordered labeled tree* are used interchangeably hereafter.

[4] While the order of attributes (unlike elements) is irrelevant in XML [1], yet we adopt an ordered tree model to simplify processing [69, 119].

[5] *Knowledge base* & *semantic network* are used interchangeably hereafter.

[6] Concept frequencies allow to estimate the information content that concepts share in common, used in *node-based* semantic similarity (Section 2.1.5).

targeted for disambiguation are replaced by semantic concepts (identifiers) extracted from a reference semantic network, whereas non-target XML nodes remain untouched, such that the original XML tree structure remains intact (Fig 4.b) ●

3.2. Linguistic Pre-Processing

Linguistic pre-processing consists of three main phases: i) tokenization, ii) stop word removal, and iii) stemming, applied on each of the input XML document's element/attribute tag names and text values, to produce corresponding XML tree node labels. Here, we consider three possible inputs:

– Element/attribute tag names consisting of individual words,
– Element/attribute tag names consisting of compound words, usually made of two individual terms (t_1 and t_2)[7] separated by special delimiters (namely the underscore character, e.g., "*Directed_By"*), or the use of upper/lower case to distinguish the individual terms (e.g., "*FirstName"*),
– Element/attribute text values consisting of sequences of words separated by the space character.

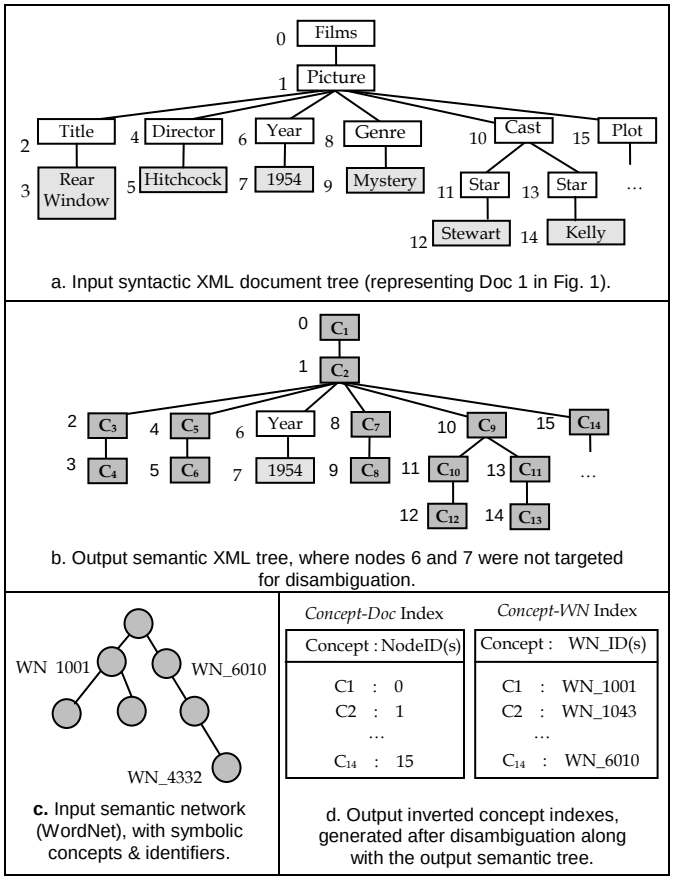


Fig. 4. Sample input (syntactic) XML and output (semantic) XML trees.

Considering the first case, no significant pre-processing is required, except for stemming (when the word is not found in the reference semantic network). In the second case (i.e., compound words, usually disregarded in existing methods), if t_1 and t_2 match a single concept in the semantic network (i.e., a synset in WordNet, e.g., *first name*), they are considered as a single token. Otherwise, they are considered as two separate terms, and are processed for stop word removal and stemming. Yet, they are kept within a single XML node (x) as its label ($x.\ell$) in order to be treated together afterward, i.e., one sense will be finally associated to $x.\ell$, which is formed by the best combination of t_1 and t_2's senses (in contrast with studies in [57, 58, 106] which process token senses separately as distinct labels). As for the third case, we apply traditional tokenization, i.e., the text value sentence is broken up into a set of word tokens t_i, processed for stop word removal and stemming, and then represented each as an individual node (x_i) labeled with the corresponding token ($x_i.\ell = t_i$), and appearing as a child of the containing element/attribute node.

3.3. Node Selection for Disambiguation

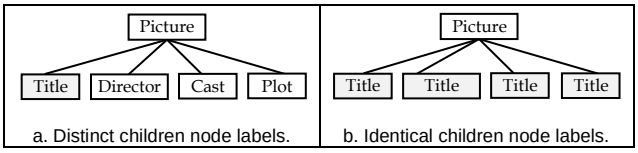
Given an input XML tree, the first step is to select target nodes to disambiguate, which (we naturally assume) are the most ambiguous nodes in the document tree. Thus, we aim to provide a mathematical definition to quantify an XML node ambiguity degree which can be used to select target nodes for disambiguation (answering *Motivation 1*). To do so, we start by clarifying our intuitions about XML node ambiguity:

- **Assumption 0:** An XML node whose label has only one possible sense is considered to be unambiguous, i.e., its semantic ambiguity is minimal.
- **Assumption 1:** The semantic ambiguity of an XML node is related to the number of senses of the node's label: i) the more senses it has, the more ambiguous the node is, ii) the fewer senses it has, the less ambiguous the node is.
- **Assumption 2:** The semantic ambiguity of an XML node is related to its distance to the root node of the document tree: i) the closer it is to the root, the more ambiguous it is, ii) the farther it is from the root, the less ambiguous it is.

Assumption 2 follows the nature of XML and semi-structured data, where nodes closer to the root of the document tend to be more descriptive of the whole document, i.e., having a broader meaning, than information further down the XML hierarchy [11, 119]. In other words, as one descends in the XML tree hierarchy, information becomes increasingly specific, consisting of finer details [102], and thus tends to be less ambiguous.

- **Assumption 3:** The semantic ambiguity of an XML node is related to its number of children nodes having distinct labels: i) the lesser the number of distinct children labels, the more ambiguous the node is, ii) the more the number of distinct children labels, the less ambiguous the node is.

Assumption 3 is highlighted in the sample XML trees in Fig. 5. One can clearly identify the meaning of root node label *"Picture"* (i.e., *"motion picture"*) in Fig. 5.a., by simply looking at the node's distinct children labels. Yet the meaning of *"Picture"* remains ambiguous in the XML tree of Fig. 5.b (having several children nodes but with identical labels). Hence, we believe that distinct children node labels can provide more hints about the meaning of a given XML node, making it less ambiguous.



[7] Having more than two terms per XML tag name is unlikely in practice [106]. This is also true with WordNet concept names, usually made of 1-to-2 terms.

Fig. 5. Sample XML document trees.

While our goal is to quantify XML semantic ambiguity, yet this can be done in many alternative ways that would be consistent with the above assumptions. Hence, we first provide some basic definitions that introduce a set of *ambiguity degree factors* mapping to the above assumptions, which we will then utilize to build our integrated ambiguity degree measure.

Definition 5 – XML Node Polysemy factor: The polysemy ambiguity degree factor of an XML node x in tree T, noted $Amb_{Polysemy}$, increases when the number of senses of $x.\ell$ is high in the reference semantic network SN, or else it decreases:

$$\mathrm{Amb}_{Polysemy}(x.\ell, SN) = \frac{senses(x.\ell) - 1}{\mathrm{Max}(senses(SN)) - 1} \in [0,1] \quad \textbf{(5)}$$

where $Max(senses(SN))$ is the maximum number of senses of a word/expression in SN (e.g., in WordNet 3.0 [28], $Max(senses(SN)) = 33$, for the word *"head"*) ●

Lemma 1: The XML node polysemy factor $Amb_{Polysemy}$ in Definition 5 varies in accordance with Assumptions 0 and 1 ●

Proof of Lemma 1: Given formula (5), $Amb_{Polysemy}$ varies as follows:

- The minimum value $Amb_{Polysemy} = 0$ is obtained when $x.\ell = 1$, i.e., ℓ has only one sense (e.g., *"first name"* in WordNet), i.e., x is without ambiguity: it always refers to the same meaning.
- The maximum value $Amb_{Polysemy} = 1$ is obtained when $x.\ell = Max(senses(SN))$.
- When $senses(x.\ell)$ increases/decreases, $Amb_{Polysemy}$ follows accordingly such that $Amb_{Polysemy} \in [0, 1]$.

Definition 6 – XML Node Depth factor: The depth ambiguity degree factor of an XML node x in tree T, noted Amb_{Depth}, increases when the distance in number of edges between x and $R(T)$ is low, or else it decreases such that:

$$\mathrm{Amb}_{Depth}(x, T) = 1 - \frac{x.d}{\mathrm{Max}(depth(T))} \in [0,1] \quad \textbf{(6)}$$

where $Max(depth(T))$ is the maximum depth in T ●

Lemma 2: The XML node depth factor Amb_{Depth} in Definition 6 varies in accordance with Assumption 2 ●

Proof of Lemma 2: Given formula (6), Amb_{Depth} varies as follows:

- The maximum value $Amb_{Depth}=1$ is obtained when $x.d = 0$, i.e., when $x = R(T)$.
- The minimum value $Amb_{Depth} = 0$ is obtained when $x.d = Max(depth(T))$, i.e., when x is one of the farthest nodes from $R(T)$: one of the deepest (most specific) leaf nodes in T's hierarchy.
- When $x.d$ increases/decreases, Amb_{Depth} follows inversely such that $Amb_{Depth} \in [0, 1]$.

Definition 7 – XML Node Density factor: The density ambiguity degree factor of an XML node x in tree T, noted $Amb_{Density}$, increases when the number of children nodes of x having distinct labels, designated as $\overline{x.f}$, is low, or else it decreases such that:

$$\mathrm{Amb}_{Density}(x, T) = 1 - \frac{\overline{x.f}}{\mathrm{Max}(\overline{fan\text{-}out(T)})} \in [0,1] \quad \textbf{(7)}$$

where $Max(\overline{fan\text{-}out(T)})$ is the maximum number of children nodes with distinct labels in T. We identify this factor as the *node density factor* to distinguish it from traditional *node fan-out*: number of children nodes (regardless of label, Definition 1) ●

Lemma 3: The XML node density factor $Amb_{Density}$ in Definition 7 varies in accordance with Assumption 2 ●

Proof of Lemma 3: Given formula (7), $Amb_{Density}$ varies as follows:

- The maximum value $Amb_{Density} = 1$ is obtained when $\overline{x.f} = 0$, i.e., when x is a leaf node and does not have any children nodes (to provide hints concerning x's meaning).
- The maximum value $Amb_{Density} = 0$ is obtained when $\overline{x.f} = Max(fan\text{-}out(T))$, i.e., when x has the largest number of children nodes with distinct labels in T. In other words, it has the highest possible number of hints about its meaning in T.
- When $\overline{x.f}$ increases/decreases, $Amb_{Density}$ follows inversely such that $Amb_{Density} \in [0, 1]$.

From the above ambiguity factor definitions, we can provide an integrated definition for XML ambiguity degree specifically tailored to satisfy all the stated assumptions:

Definition 8 – XML Node Ambiguity Degree: Given an XML tree T, a node $x \in T$, and a reference semantic network SN, we define the ambiguity degree of x, $Amb_Deg(x, T, SN)$, as the ratio between $Amb_{Polysemy}(x.\ell, SN)$ on one hand, and the sum of $1\text{-}Amb_{Depth}(x, T)$ and $1\text{-}Amb_{Density}(x, T)$ on the other hand:

$$\mathrm{Amb_Deg}(x, T, SN) = \frac{w_{Polysemy} \times \mathrm{Amb}_{Polysemy}(x.\ell, SN)}{w_{Depth} \times (1 - \mathrm{Amb}_{Depth}(x, T)) + w_{Density} \times (1 - \mathrm{Amb}_{Density}(x, T)) + 1} \in [0,1] \quad \textbf{(8)}$$

where $w_{Polysemy}$, w_{Depth}, $w_{Density} \in [0, 1]$ are independent weight parameters allowing the user to fine-tune the contributions of polysemy, depth, and density factors respectively ●

In fact, we chose to define Amb_Deg as a ratio (rather than a typical weighted sum) with $Amb_{Polysemy}$ as the only numerator factor: in order to guarantee that Amb_Deg is minimal (= 0) when the $Amb_{Polysemy}$ is minimal (=0), regardless of other factors which were allocated to the denominator. The latter Amb_{Depth} and $Amb_{Density}$ factors were combined (within the denominator) using weighted sum aggregation, with their values reversed ($1\text{-}Amb_{Depth}$ and $1\text{-}Amb_{Density}$) to counter their usage as denominator factors: varying proportionally with $Amb\text{-}Deg$. As a result:

Lemma 4: The ambiguity degree measure Amb_Deg in Definition 8 varies in accordance with Assumptions 0-3 ●

Proof of Lemma 4: Given formula 8, Amb_Deg varies as follows:

- The minimum value *Amb_Deg* = 0 is obtained when the label of node *x* has only one possible sense, i.e., when $Amb_{Polysem}(x.\ell, SN) = 0$, regardless of its depth and density factors, and regardless of parameter weights (Assumption 0 and Lemma 1).
- The maximum value *Amb_Deg* = 1 is obtained when: i) $x.\ell$ has the maximum number of senses in *SN* (e.g., 33 senses in WordNet [28]), ii) *x* is the root node of *T*, *x.d* = 0, and iii) *x* has the minimum number of children nodes with distinct labels in *T*, $\overline{x.f} = 0$, regardless of w_{Depth} and $w_{Density}$ parameter values (Assumptions 1-3, and Lemmas 1-3).
- The value of *Amb_Deg* increases with: i) the increase in $x.\ell$'s polysemy in *SN*, ii) the decrease in *x*'s depth, and iii) the decrease in *x*'s density in the XML document tree. Inversely, the value of *Amb_Deg* decreases with: i) the decrease in $x.\ell$'s polysemy, ii) the increase in *x*'s depth, and iii) the increase in *x*'s density (Assumptions 1-3, and Lemmas 1-3).

Special case: When the label of node *x* consists of a compound word made of tokens t_1 and t_2, we compute *Amb_Deg*(*x*) as the average of the ambiguity degrees of t_1 and t_2.

Amb_Deg is computed for all nodes in the input XML tree. Then, only the most ambiguous nodes are selected as targets for disambiguation following an ambiguity threshold $Thresh_{Amb}$ automatically estimated or set by the user, i.e., nodes having $Amb_Deg(x, T, SN) \geq Thresh_{Amb}$, whereas remaining nodes are left untouched. Note that the user can disregard the ambiguity degree measure: i) by setting $w_{Polysemy} = 0$ so that all nodes end up having *Amb_Deg* = 0 regardless of polysemy, depth, and density factors, or ii) by setting $Thresh_{Amb} = 0$ so that all nodes are selected for disambiguation regardless of their ambiguity degrees.

Note that the fine-tuning of parameters is an optimization problem such that parameters should be chosen to maximize disambiguation quality (through some cost function such as *f-value*, cf. Section 4). This can be solved using a number of known techniques that apply linear programming and/or machine learning in order to identify the best weights for a given problem class, e.g., [38, 60, 66]. Providing such a capability, in addition to manual tuning, would enable the user to start from a sensible choice of values (e.g., identical weight parameters to consider all ambiguity features equally, i.e., $w_{Polysemy} = w_{Depth} = w_{Fan\text{-}out} = 1$, with a minimal threshold $Thresh_{Amb} = 0$ to consider all results initially) and then optimize and adapt the disambiguation process following the scenario and optimization (cost) function at hand. We do not further address the fine-tuning of parameters here since it is out of the scope of this paper (to be addressed in a future study).

3.4. Context Definition and Representation

3.4.1. Sphere Neighborhood

For each target node selected from the previous phase, node contexts have to be defined and processed for disambiguation. While current approaches only partly consider the semi-structured nature of XML in defining disambiguation contexts (*Motivation 2*), we introduce the *sphere neighborhood* context model, inspired from the sphere-search paradigm in XML IR [33][8], taking into account the whole structural surrounding of an XML target node (including its ancestors, descendants, and siblings) in defining its context. We define the notion of context ring as the set of nodes situated at a specific distance from the target node. An XML sphere encompasses all rings included at distances less or equal to the size (radius) of the sphere.

Definition 8 – Context Ring: Given an XML tree *T* and a target node $x \in T$, we define an XML context ring with center *x* and radius *d* as the set of nodes located at distance *d* from *x* with respect to the XML structure containment relationship, i.e., $R_d(x) = \{x_i \in T \mid Dist(x, x_i) = d\}$. $R_d(x)$ is more generally noted $R_{d,rel}(x)$ when a different relationship *rel* (other than XML structure containment) is considered in building the ring (e.g., using hyperlinks, or semantic relationships when available) ●

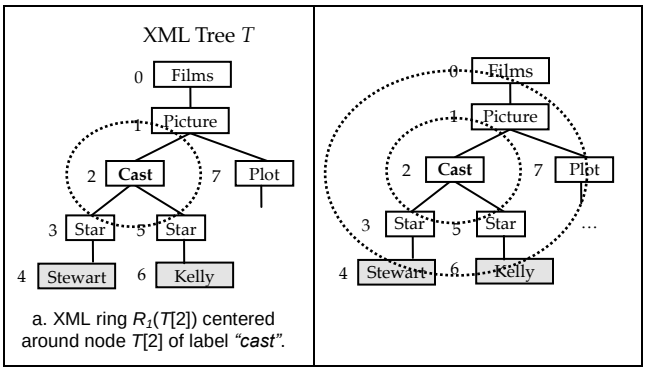


Fig. 6. Sample XML context (ring and) sphere neighborhoods.

The distance between two XML nodes in an XML tree, $Dist(x_i, x_j)$, is evaluated as the number of edges separating the nodes. For instance, in tree *T* of Fig. 6.a, the distance between nodes *T*[2] and *T*[6] of labels "*cast*" and *"Kelly"* respectively is equal to 2. Hence, the XML ring $R_1(T[2])$ centered around node *T*[2] (*"cast"*) at distance 1 consists of nodes: *T*[1] (*"Picture"*), *T*[3] (*"star"*) and *T*[5] (*"star"*).

Definition 9 – Context Sphere: Given an XML tree *T*, a target node $x \in T$, and a set of XML context rings $R_{d_j}(x) \subseteq T$, we define an XML context sphere with center *x* and radius *d* as the set of nodes in the rings centered around *x* at distances less or equal to *d*, i.e., $S_d(x) = \{x_i \in T \mid x_i \in R_{d_j}(x) \wedge d_j \leq d\}$ ●

In Fig. 6.b, the XML sphere $S_2(T[2])$ centered around node *T*[2] of label *"cast"* with radius 2 consists of: ring $R_1(T[2])$ of radius 1 comprising nodes *T*[1] (*"picture"*), *T*[3] (*"star"*) and *T*[5] (*"star"*), and ring $R_2(T[2])$ of radius 2 comprising nodes *T*[0] (*"Films"*), T[4] (*"Stewart"*), T[6] (*"Kelly"*), and *T*[7] (*"Plot"*). The size (radius) of the XML sphere context is tuned following user preferences and/or the nature of the XML data at hand (e.g., XML trees might contain specialized and domain-specific data, and thus would only require small contexts to achieve good disambiguation, whereas more generic XML data might require larger contexts to better describe the intended meaning of node labels and values, cf. experiments in Section 4).

While the notion of context *sphere* with XML seems limited to that of a *disk* in 2D space (cf. Fig. 6), nonetheless, our sphere neighborhood model is general and can be straightforwardly extended to consider different kinds of rings having different tree

[8] While comparable with the concept of *XML sphere* exploited in [33] the latter consists of an XML retrieval paradigm for computing TF-IDF scores to rank XML query answers, which is orthogonally different, in its use and objectives, from our disambiguation proposal.

node distance functions (including edge weights, density, or direction, etc. [31, 41]), and different kinds of node relationships (including hyperlinks, or semantic relationships between nodes when available). For instance, it can be applied to describe the contexts of concepts in a semantic network *SN* (which we adopt in our context-based disambiguation algorithm, cf. Section 3.5.2), where $R_{d,rel}(c)$ designates the context ring centered around concept *c* and containing all concepts at distance *d* from *c* w.r.t. semantic relationship *rel*. The context sphere $S_d(c)$ would thus contain all rings at distance *d* from *c* defined w.r.t. the different kinds of semantic relationships considered in defining its context rings (cf. Fig. 7).

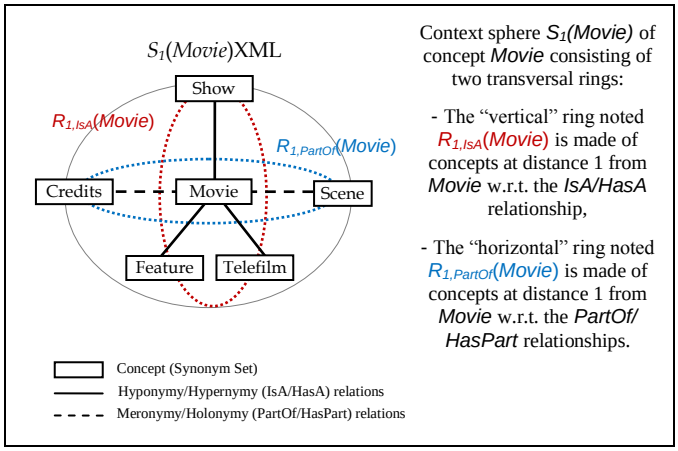


Fig. 7. Sample context (ring and) sphere neighborhoods of concept *Movie* based on the *SN* extract in Fig. 2.

Note that in this study, we restrict ourselves to the most intuitive notion of distance in building our ring and sphere contexts, where distance is evaluated as the number of XML structural containment edges (semantic relationship edges when applied on a *SN*), and report the investigation of more sophisticated XML (semantic network) distance functions to a dedicated study.

3.4.2. Context Vector Representation

Having identified the context of a given XML target node, we need to evaluate the impact of each of the corresponding context nodes in performing semantic disambiguation (in contrast with existing methods using the *bag-of-words* paradigm where context is processed as a set of words/nodes disregarding XML structure: *Motivation 3*). Here, we introduce a *relational information* solution based on the general vector space model in information retrieval [62] (in comparison with the specific decay function used in [57, 58]), designed to consider the structural proximity/relationships among XML nodes in computing disambiguation scores following our sphere neighborhood model. Our mathematical formulation follows two basic assumptions:

- **Assumption 4:** XML context nodes closer to the target node should better influence the latter's disambiguation, whereas those farther away from the target node should have a smaller impact on the disambiguation process.

This is based on the structured nature of XML, such as nodes closer together in the XML hierarchy are typically more related than more separated ones.

- **Assumption 5:** Nodes with identical labels, occurring multiple times in the context of a target node, should better influence the latter's disambiguation in contrast with nodes with identical labels occurring a lesser number of times.

This is based on the notion of context in WSD, where words occurring multiple times in the context of a target word have a higher impact on the target's meaning[9]. Therefore, we represent the context of a target XML node *x* as a weighed vector, whose dimensions correspond to all distinct node labels in its sphere neighborhood context, weighted following their structural distances from the target node.

Definition 10 – XML Context Vector: Given a target node $x \in$ XML tree *T*, and its sphere neighborhood $S_d(x) \in T$, the corresponding context vector $\overrightarrow{V_d(x)}$ is defined in a space whose dimensions represent, each, a single node label $\ell_r \in S_d(x)$, such as $1 < r < n$ where *n* is the number of distinct node labels in $S_d(x)$. The coordinate of a context vector $\overrightarrow{V_d(x)}$ on dimension ℓ_r, $w_{\overrightarrow{V_d(x)}}(\ell_r)$, stands for the weight of label ℓ_r in sphere $S_d(x)$ •

Definition 11 – XML Node Label Weight: The weight $w_{\overrightarrow{V_d(x)}}(\ell_r)$ of node label ℓ_r in context vector $\overrightarrow{V_d(x)}$ corresponding to the sphere neighborhood $S_d(x)$ of target node *x* and radius *d*, consists of the structural frequency of nodes $x_i \in S_d(x)$ having label $x_i.\ell = \ell_r$. It is composed of an occurrence frequency factor $Freq(\ell_r, S_d(x))$ (based on Assumption 5) defined using a structural proximity factor $Struct(x_i, S_d(x))$ (based on Assumption 4). Formally, given $|S_d(x)|$ the cardinality (in number of nodes) of $S_d(x)$, $Freq(\ell_r, S_d(x))$ underlines the total number of occurrences of nodes $x_i \in S_d(x)$ having label $x_i.\ell = \ell_r$, weighted w.r.t. structural proximity:

$$\mathrm{Freq}(\ell_r, S_d(x)) = \sum_{\substack{x_i \in S_d(x) \,/ \\ x_i.\ell = \ell_r}} \mathrm{Struct}(x_i, S_d(x)) \in \left[\frac{1}{d+1}, \frac{|S_d(x)|+1}{2}\right] \quad \textbf{(9)}$$

$Struct(x_i, S_d(x))$ underlines the structural proximity between each context node $x_i \in S_d(x)$ having $x_i.\ell = \ell_r$, and the target (sphere center) node *x*, formally:

$$\mathrm{Struct}(x_i, S_d(x)) = 1 - \frac{\mathrm{Dist}(x, x_i)}{d+1} \in \left[\frac{1}{d+1}, 1\right] \quad \textbf{(10)}$$

The denominator in $Struct(x_i, S_d(x))$ is incremented by 1 (i.e., *d*+1) to allow context nodes occurring in the farthest ring of the sphere context $S_d(x)$, i.e., the ring $R_d(x)$ of radius *d*, to have a non-null weight in $\overrightarrow{V_d(x)}$, and thus a non-null impact on the disambiguation of target node *x*.

Consequently, the combined weight factor is defined as the normalized occurrence frequency factor:

$$w_{\overrightarrow{V_d(x)}}(\ell_r) = \frac{\mathrm{Freq}(\ell_r, S_d(x))}{\mathrm{Max}_{\mathrm{Freq}}} = \frac{2 \times \mathrm{Freq}(\ell_r, S_d(x))}{|S_d(x)|+1} \in [0,1] \quad \textbf{(11)}$$

[9] This is also similar to the notion of *term frequency* (or TF) in information retrieval, where the importance of a given term *t* in describing a document *D* increases with the frequent occurrence of *t* in *D* [43, 62].

where $\text{Max}_{\text{Freq}} = \frac{|S_d(x)|+1}{2}$ following formula (9) ●

For instance, given the XML tree in Fig. 6, Fig. 8 shows context vectors of sphere neighborhoods $S_1(T[2])$ and $S_2(T[2])$ centered around node $T[2]$ of label *"Cast"*. Considering $\overrightarrow{V_1(T[2])}$:

- $w_{\overrightarrow{V_1(\text{"Cast"})}}(\text{"Cast"}) = \frac{2\times(1)}{(4)+1} = 0.4$ given that: i) *"Cast"* is the label of $S_1(T[2])$'s target (center) node $T[2]$, i.e., $Struct(T[2], S_1(T[2]))=1$, ii) $T[2]$ is the only node occurrence of label *"cast"* in $S_1(T[2])$, i.e., $Freq(\text{"Cast"}, S_1(T[2]))=1$, and iii) $|S_1(T[2])| = 4$.
- $w_{\overrightarrow{V_1(\text{"Cast"})}}(\text{"Star"}) = \frac{2\times(0.5+0.5)}{(4)+1} = 0.4$ given that: i) $S_1(T[2])$ contains two nodes $T[3]$ and $T[5]$ of label *"Star"* at distance = 1 from the target node $T[2]$, i.e., $Struct(T[3], S_1(T[2]))$ = 1-$\frac{1}{2}$ = 0.5, ii) $T[3]$ and $T[5]$ are the only node occurrences of label *"Star"* in $S_1(T[2])$, i.e., $Freq(\text{"Star"}, S_1(T[2]))$ = 0.5 + 0.5 = 1, and iii) $|S_1(T[2])| = 4$.

Similarly for remaining context vector weight computations.

	Cast	*Picture*	*Star*
$\overrightarrow{V_1(T[2])}$	0.4	0.2	0.4

	Cast	*Picture*	*Star*	*Films*	*Stewart*	*Kelly*	*Plot*
$\overrightarrow{V_2(T[2])}$	0.2223	0.1482	0.2964	0.0741	0.0741	0.0741	0.0741

Fig. 8. Sample sphere context vectors based on the sphere neighborhoods in Fig. 6.

Here, one can realize that label weights in Fig. 8 increase as nodes occur closer to the target node (Assumption 4), and as the number of node label occurrences increases in the sphere context (Assumption 5, e.g., in $\overrightarrow{V_1(T[2])}$, $w_{\overrightarrow{V_1(\text{"Cast"})}}(\text{"Star"}) = 2 \times w_{\overrightarrow{V_1(\text{"Cast"})}}(\text{"Picture"})$ since node label "*Star*" occurs twice in $S_1(T[2])$ while "*Picture*" occurs once; similarly in $\overrightarrow{V_2(T[2])}$). Formally:

Lemma 5: The context vector weight measure $w_{\overrightarrow{V_d(x)}}(\ell_r)$ in Definition 11 varies in accordance with Assumptions 5 and 6 ●

Proof of Lemma 5: Given formula (11), $w_{\overrightarrow{V_d(x)}}(\ell_r)$ varies as:

- $w_{\overrightarrow{V_d(x)}}(\ell_r)$ increases with $Freq(\ell_r, S_d(x))$, which increases with $Struct(x_i, S_d(x))$:
  - The value of $Struct(x_i, S_d(x))$ is inversely proportional to the distance between the target node x and context node x_i (following Assumption 1).
    - The minimum value $Struct(x_i, S_d(x)) = \frac{1}{d+1}$ is reached when $x_i \in R_d(x)$ where $R_d(x)$ is the outer-most ring in sphere $S_d(x)$.
    - The maximum value $Struct(x_i, S_d(x))=1$ is reached when processing the target node itself, i.e., $x_i = x$.
  - The value of $Freq(\ell_r, S_d(x))$ is proportional to the number of occurrences of nodes having the same label $x_i.\ell = \ell_r$ (following Assumption 2).
    - The minimum value $Freq(\ell_r, S_d(x)) = \frac{1}{d+1}$ is reached when there is only one context node having label ℓ_r and occurring on the outer-most ring $R_d(x)$ of the sphere neighborhood, more formally: $\exists\ x_i \in S_d(x)\ /\ (x_i.\ell = \ell_r) \wedge (x_i \in R_d(x)) \wedge (\forall\ x_j \in S_d(x) / x_j.\ell \neq \ell_r)$
    - The maximum value of $Freq(\ell_r, S_d(x)) = \frac{|S_d(x)|+1}{2}$ is reached when all context nodes have the same label ℓ_r and appear on the inner-most ring $R_1(x)$ of the sphere neighborhood, more formally: $\forall\ x_i \in S_d(x) / (x_i.\ell = \ell_r) \wedge (x_i \in R_1(x))$. Here, $Freq(\ell_r, S_d(x)) = 1 + \frac{1}{2} \times (|S_d(x)|-1) = \frac{|S_d(x)|+1}{2}$

Consequently:

- The minimum value $w_{\overrightarrow{V_d(x)}}(\ell_r) = 0$ is obtained when no nodes of label ℓ_r occur in the sphere context of target node x.
- The maximum value $w_{\overrightarrow{V_d(x)}}(\ell_r) = 1$ is obtained when maximum frequency is obtained, since the weight score is normalized using maximum frequency, i.e., $\frac{|S_d(x)|+1}{2}$

In short, context nodes are weighted based on their labels' occurrences as well as the sizes (radiuses) of the sphere contexts to which they correspond, varying their weights and thus their impact on the target node's disambiguation accordingly.

3.5. XML Semantic Disambiguation

Once the contexts of XML nodes have been determined, we process each target node and its context nodes for semantic disambiguation. Here, we propose to combine two strategies: the *concept-based* approach and the *context-based* approach. The former is based on semantic concept comparison between target node senses (concepts) and those of its sphere neighborhood context nodes, whereas the latter is based on context vector comparison between the target node's sphere context vector in the XML tree and context vectors corresponding to each of its senses in the reference semantic network. The user will be able to combine and fine-tune both approaches (*Motivation 4*).

3.5.1. Concept-based Semantic Disambiguation

It consists in comparing the target node with its context nodes, using a combination of semantic similarity measures (*edge-based*, *node-based*, and *gloss-based*, cf. Section 2.1) in order to compare corresponding semantic concepts in the reference semantic network. Then, the target node sense with the maximum similarity (relatedness) score, w.r.t. context node senses, is chosen as the proper target node sense. To do so, we propose an extension of context-based WSD techniques (cf. Section 2.2.3) where we:

- Build upon the sphere neighborhood context model, to consider XML structural proximity in evaluating the semantic meanings of context nodes (in comparison with the traditional *bag-of-words* context model),
- Allow an extensible combination of several semantic similarity measures, in order to capture semantic relatedness from different perspectives (in comparison with most existing methods which exploit pre-selected measures).

Definition 12 – Concept-based Semantic Score: Given a target node $x \in$ XML tree T and its sphere neighborhood $S_d(x) \in T$, and given s_p as one possible sense for $x.\ell$ in a semantic network SN, we define $Concept_Score(s_p, S_d(x), SN)$ to quantify the semantic impact of s_p as the potential candidate for the intended sense (meaning) of $x.\ell$ within context $S_d(x)$ in T w.r.t. SN, computed as the average of the weighted maximum similarities between s_p and context node senses:

$$\text{Concept_Score}(s_p, S_d(x), SN) = \sum_{x_i \in S_d(x)} \frac{\underset{s_j^i \rightarrow x_i.\ell}{\text{Max}} \left(\text{Sim}(s_p, s_j^i, \overline{SN}) \right) \times w_{\overline{V_d(x)}}(x_i.\ell)}{|S_d(x)|} \in [0,1] \quad \textbf{(12)}$$

where s_j^i designates the jth sense of context node $x_i.\ell \in S_d(x)$, and $Sim(s_p, s_j^i, SN)$ designates the semantic similarity measure between senses s_p and s_j^i w.r.t. SN ●

Definition 13 – Semantic Similarity Measure: It quantifies the semantic similarity (relatedness[10]) between two concepts (i.e., word senses) c_1 and c_2 in a reference semantic network SN, computed as the weighted sum of several semantic similarity measures[11]. Formally:

$$\begin{aligned} \text{Sim}(c_1, c_2, SN) = {} & w_{Edge} \times \text{Sim}_{Edge}(c_1, c_2, SN) + \\ & w_{Node} \times \text{Sim}_{Node}(c_1, c_2, SN)) + \\ & w_{Gloss} \times \text{Sim}_{Gloss}(c_1, c_2, SN)) \quad \in [0, 1] \end{aligned} \quad \textbf{(13)}$$

where:

- $w_{Edge} + w_{Node} + w_{Gloss} = 1$ and $(w_{Edge}, w_{Node}, w_{Gloss}) \geq 0$
- Sim_{Edge} is a typical edge-based measure from [114], cf. formula (1)
- Sim_{Node} is a typical node-based measure from [50], cf. formula (2)
- Sim_{Gloss} is a typical gloss-based measure from [8], cf. formula (3), expanded and normalized following Definition 14 ●

Definition 14 – Expanded/Normalized Gloss Similarity Measure: Given concepts c_1 and c_2 in a semantic network SN, we define an expanded version of the basic gloss-based measure from [8] (cf. formula (3)) to include (in addition to the concept's original gloss $gloss(c_i)$ and the glosses if its surrounding concepts in SN connected using different semantic relations $gloss(rel(c_i))$ the concept's set of synonymous words/expressions $syns(c_i)$ within its extended gloss, thus enriching the latter with more semantically related terms. Formally:

$$\begin{aligned} \text{Sim}_{Gloss}(c_1, c_2, SN) &= \text{ExtGloss}(c_1) \cap \text{ExtGloss}(c_1) \\ &= \left(\text{gloss}(c_1) \cup \text{gloss}(rel(c_1) \cup syns(c_1))\right) \cap \\ &\quad \left(\text{gloss}(c_2) \cup \text{gloss}(rel(c_2) \cup syns(c_1))\right) \in [0, 1] \end{aligned} \quad \textbf{(14)}$$

where the *set overlapping* (intersection scoring) mechanism, noted $\cap$ (which was originally designed to assign a score of n^2 to an n-word overlap sequence, thus producing non-normalized scores following [8][12]) is normalized as follows:

$$\text{ExtGloss}(c_1) \cap \text{ExtGloss}(c_2) = \frac{\sum_m n^2}{Max(|\text{ExtGloss}(c_1)|, |\text{ExtGloss}(c_1)|)^2} \in [0,1] \quad \textbf{(15)}$$

having m the number of sequences of n consecutive overlapping words between the two extended glosses of c_1 and c_2, where m and $n \in [1, \text{Max}((|ExtGloss(c_1)|, |ExtGloss(c_2)|)]$ ●

Lemma 6. The gloss similarity measure in Definition 14 produces normalized similarity scores $\in [0, 1]$ ●

Proof of Lemma 6: Given formulas (14) and (15), Sim_{Gloss} varies as follows:

- The minimum value $Sim_{Gloss} = 0$ is obtained when the two extended glosses of concepts c_1 and c_2 being compared are completely different, i.e., the number of n-word overlap sequences between the latter is = 0.
- The maximum value $Sim_{Gloss} = 1$ is obtained when c_1 and c_2 have the exact same extended glosses, i.e., having one single n-word overlap sequence (m=1) where $n = |ExtGloss(c_1)| = |ExtGloss(c_2)|$.
- The value of Sim_{Gloss} increases with the increase of the number (m) and size (n) of n-word overlap sequences, and decreases otherwise.

Algorithm $XSD_{Concept}$ for concept-based XML Semantic Disambiguation, following Definition 12, is provided in Fig. 9. It accepts as input a target XML node x and a user specified sphere neighborhood radius d, along with the source XML tree T and a reference semantic network SN. After generating x' sphere neighborhood $S_d(x)$ and context vector $\overrightarrow{V_d(x)}$ (line 1), each of the target label senses s_p in SN is compared with the senses s_j^i of each of the context node labels $x_i.\ell$ (lines 3-9) weighted using corresponding sphere neighborhood vector weights $w_{\overline{V_d(x)}}(x_i.\ell)$ (line 10). The target node sense with the maximum combined similarity score is

[10] Semantic *relatedness* is more general than *similarity*: dissimilar concets may be semantically related by lexical relationships such as meronymy (*car–wheel*), antonymy (*dark–light*), or any kind of functional relationship (e.g., *pencil–paper, rain–flood*). Most semantic similarity measures typically capture relatedness rather than just similarity, which is required in WSD [14].

[11] Here, we use three typical semantic similarity measures, yet any other semantic similarity measure can be used, or combined with the latter.

[12] The authors in [8] consider that a phrasal n-word overlap (i.e., overlap of a sequence of n consecutive words) is a much rarer occurrence than a single word overlap, and thus assign an n-word overlap the score of n^2, which is greater than the sum of the scores assigned to those n individual word overlaps which is equal to n ($=1^2 + 1^2 + \ldots$ n times).

then chosen as the output concept *c* which best describes the sense (meaning) of the target node's label *x*. ℓ w.r.t. *SN* (lines 12-13).

Algorithm $XSD_{Concept}$

Input: x, d // Target XML node *x* and sphere neighborhood radius *d*
T // Source XML tree
SN // Reference semantic network
Output: c // Semantic concept representing the sense (meaning) of *x*.ℓ

Begin 1
Generate $S_d(x) \in T$ and $\overrightarrow{V_d(x)}$ // Context vector of *x* in *T* 2
For each $s_p \in SN \rightarrow x.\ell$ // For each sense of the target node label 3
{ 4
For each $x_i \in S_d(x)$ // Processing senses of context nodes 5
{ 6
For each $s_j^i \in SN \rightarrow x_i.\ell$ { $Sim_Score(s_j^i) = Sim(s_p, s_j^i)$ } 7
$Max_Score(x_i.\ell) = \max_{s_j^i \rightarrow x_i.\ell} \left(Sim_Score(s_j^i) \right)$ 8
} 9
$Concept_Score(s_p,) = \sum_{x_i \in Sd(x)} \frac{Max_Score(x_i.\ell) \times w_{\overrightarrow{V_d(x)}}(x_i.\ell)}{|S_d(x)|}$ 10
} 11
$c = s_k$ / $Concept_Score(s_k) = \max_{s_r \rightarrow x.\ell} \left(Concept_Score(s_p) \right)$ 12
Return c // Sense (concept) with maximum score 13
End 14

Fig. 9. Concept-based XML Semantic Disambiguation algorithm.

Special case: If the target node label *x*.ℓ is a compound word consisting of two tokens t_1 and t_2 for which no single match was found in the reference semantic network *SN* (Section 3.2), an average score for each possible combination of senses (s_p, s_q) corresponding to each of the individual token senses (s_p for token t_1, and s_q for t_2) is computed to identify the sense combination which is most suitable for the compound node label:

$$Concept_Score((s_p, s_q), S_d(x), SN) = \sum_{x_i \in S_d(x)} \frac{\max_{s_j^i \rightarrow x_i.\ell} \left(Sim((s_p, s_p), s_j^i, SN) \right) \times w_{\overrightarrow{V_d(x)}}(x_i.\ell)}{|S_d(x)|} \in [0,1] \quad \textbf{(16)}$$

where

$$Sim((s_p, s_p), s_j^i, SN) = \frac{Sim(s_p, s_j^i, SN) + Sim(s_q, s_j^i, SN)}{2} \in [0,1]$$

Note that a compound context node label $x_i.\ell$ which tokens t_1^i and t_2^i do not match any single concept in *SN*, is processed similarly to a compound target node label.

3.5.2. Context-based Semantic Disambiguation

It consists in comparing the target node sphere neighborhood in the XML tree with each of its possible sense (concept) sphere neighborhoods in the reference semantic network. To do so, we adopt the same notions of sphere neighborhood and context vector (Definitions 8-11) defined for XML nodes in an XML tree, to build the sphere neighborhood and context vector of a semantic concept in the semantic network. The only difference here is that sphere rings in the semantic network are built using the different kinds of semantic relationships connecting semantic concepts (e.g., hypernyms, hyponyms, meronyms, etc., cf. Definition 3 and Fig. 7), in contrast with sphere rings in an XML tree which are built using XML structural containment relationships (Definition 2). Here, given a reference semantic network *SN*, a semantic concept $c \in SN$, and a radius *d'*, we designate by $R_{d',rel}(c)$, $S_{d'}(c)$, and $\overrightarrow{V_{d'}(c)}$ the ring, sphere, and context vector of radius *d'* corresponding to concept *c* in *SN* respectively considering the different kinds of semantic relationships *rel* connecting *c* within *SN*. Note that linguistic pre-processing (cf. Section 3.2) can be applied to concept labels (when needed[13]) before building context vectors and computing vector weights. Formally:

Definition 15 – Context-based Semantic Score: Given a target node $x \in$ XML tree *T*, its sphere neighborhood $S_d(x) \in T$ and context vector $\overrightarrow{V_d(x)}$, and given s_p as one possible sense for x.ℓ in a reference semantic network *SN*, with its sphere neighborhood $S_{d'}(s_p) \in SN$ and context vector $\overrightarrow{V_{d'}(s_p)}$, we define $Context_Score(s_p, S_d(x), d', SN)$ to quantify the semantic impact of s_p as the potential candidate designating the intended sense (meaning) of *x*.ℓ within context $S_d(x)$ in *T* w.r.t. *SN*, computed using a vector similarity measure between $\overrightarrow{V_d(x)}$ and $\overrightarrow{V_{d'}(s_p)}$:

$$Context_Score(s_p, S_d(x), d', SN) = \cos(\overrightarrow{V_d(x)}, \overrightarrow{V_{d'}(s_p)}) \in [0, 1] \quad \textbf{(17)}$$

where *cos* designates the *cosine* vector similarity measure[14] ●

Algorithm $XSD_{Context}$ for context-based XML Semantic Disambiguation, following Definition 15, is provided in Fig. 10.

Algorithm $XSD_{Context}$

Input: x, d, d' // Target XML node *x*, sphere neighborhood radiuses *d* and *d'*
T // Source XML tree
SN // Reference semantic network

Output: c // Semantic concept representing the sense (meaning) of *x*.ℓ

Begin 1
Generate $S_d(x) \in T$ and $\overrightarrow{V_d(x)}$ // Context vector of *x* in *T* 2
For each $s_p \in SN \rightarrow x.\ell$ // For each sense of the target node label 3
{ 4
Generate $S_{d'}(s_p) \in SN$ and $\overrightarrow{V_{d'}(s_p)}$ // Context vector of s_p in *SN* 5
$Context_Score(s_p) = \frac{\overrightarrow{V_d(x)} \cdot \overrightarrow{V_{d'}(s_p)}}{\| \overrightarrow{V_d(x)} \| \times \| \overrightarrow{V_{d'}(s_p)} \|}$ // Context similarity 6
} 7
$c = s_k$ / $Context_Score(s_k) = \max_{s_r \rightarrow x.\ell} \left(Context_Score(s_p) \right)$ 8
Return c // Sense (concept) with maximum score 9
End 10

Fig. 10. Context-based XML Semantic Disambiguation algorithm.

$XSD_{Context}$ accepts as input a target XML node *x* and user specified sphere neighborhood radiuses *d* and *d'*, along with the source XML tree and a reference semantic network *SN*. After generating *x*' sphere neighborhood $S_d(x)$ and context vector $\overrightarrow{V_d(x)}$ in *T* (line 1), the algorithm generates the sphere neighborhood $S_{d'}(s_p)$ and

[13] This depends on the semantic network (not required with WordNet).
[14] We adopt *cosine* since it is widely used in IR [62]. Yet, other vector similarity measures can be used, e.g., Jaccard, Pearson corr. coeff., etc.

context vector $\overrightarrow{V_{d'}(s_p)}$ for each of the target label senses s_p in *SN* (lines 3-5). It then compares the target node context vector $\overrightarrow{V_d(x)}$ with each of the label sense context vectors $\overrightarrow{V_{d'}(s_p)}$ using (cosine) vector comparison (line 6). Consequently, the target node sense with the maximum vector similarity score is chosen as the output concept *c* which best describes the sense (meaning) of the target node's label $x.\ell$ w.r.t. *SN* (lines 8-9).

Note that XML and semantic network sphere neighborhood sizes *d* and *d'* are not strictly tied (and can be chosen differently by the user) since the semantic network's structure can be quite different from the XML document structure (producing smaller/larger contexts accordingly).

Special case: If the target node label $x.\ell$ is a compound word consisting of tokens t_1 and t_2 for which no single match was found in the reference semantic network *SN*, an integrated score for each possible combination of senses (s_p, s_q) corresponding to the individual token senses (s_p for token t_1, and s_q for token t_2) is computed. Here, the sphere neighborhoods and context vectors of individual senses s_p and s_q are combined together to represent the context sphere of the combination of senses (s_p, s_q) in *SN*:

$$\text{Concept_Score}((s_p, s_q), S_d(x), SN) = \cos\left(\overrightarrow{V_d(x)}, \overrightarrow{V_{d'}(s_p, s_q)}\right) \in [0,1] \quad \textbf{(18)}$$

where $\overrightarrow{V_{d'}(s_p, s_q)}$ is a compound context vector generated based on compound sphere neighborhood $S_{d'}(s_p, s_q) = S_{d'}(s_p) \bigcup S_{d'}(s_q)$.

3.5.3. Combined Semantic Disambiguation

While *concept-based* and *context-based* disambiguation can be applied separately as described in the above sections, yet we allow the user to combine and fine-tune both approaches (answering *Motivation 4*), producing a combined score as the weighted sum of concept-based and context-based scores:

$$\begin{aligned} &\text{Combined_Score}(s_p, S_d(x), SN) = \\ &w_{Concept} \times \text{Concept_Score}(s_p, S_d(x), SN) + \\ &w_{Context} \times \text{Context_Score}(s_p, S_d(x), SN) \quad \in [0, 1] \end{aligned} \quad \textbf{(19)}$$

where $w_{Concept} + w_{Context} = 1$ and $(w_{Concept}, w_{Context}) \geq 0$

Algorithm $XSD_{Combined}$ is an integration of algorithms $XSD_{Concept}$ and $XSD_{Context}$ and is thus omitted here for clarity.

3.6. Complexity Analysis

The overall time complexity of our XML disambiguation approach following *XSDF* simplifies to: $O\left(|T| + \left(|X| \times |senses(x.\ell)| \times \left(|S_d(x)| \times |senses(x_i.\ell)| + \left(|S_d(x)| + |S_d(s_p)|\right)\right)\right)\right)$ where $|T|$ designates the cardinality (in number of nodes) of the XML tree, $|X|$ the total number of target XML nodes selected for disambiguation, *x* a target node, $|senses(x.\ell)|$ the total number of possible senses (concepts) of the target node's label $x.\ell$ in the reference semantic network *SN*, (e.g., WordNet), $|S_d(x)|$ the cardinality of the sphere neighborhood context of *x* in the XML tree, $x_i \in S_d(x)$ a context node, $|Senses(x_i.\ell)|$ the total number of possible senses of context node label $x_i.\ell$, $s_p \in senses(x.\ell)$ a possible sense for target node label $x.\ell$, and $|S_d(s_p)|$ the cardinality of the sphere neighborhood of target label sense s_p in *SN*. Complexity is evaluated as the sum of the complexities of the four main phases of *XSDF*, and mainly amounts to the complexities of our *concept-base* and *context-based* disambiguation algorithms:

- The complexity of the three initial phases of *XSDF:* i) *linguistic pre-processing* (tokenization, stop word removal and stemming), ii) *node selection for disambiguation* (ambiguity degree computation), and iii) *context definition and representation* (sphere neighborhood and context vector generation) respectively amounts to: $O(|T| \times n \times |\ell|) + O(|T|) + O(|T|)$, which simplifies to $O(|T|)$.
- The complexity of our *XML semantic disambiguation* phase simplifies to: $O\left(|X| \times |senses(x.\ell)| \times \left(|S_d(x)| \times |senses(x_i.\ell)|\right) + \left(|S_d(x)| + |S_d(s_p)|\right)\right)$
  - The complexity of algorithm $XSD_{Concept}$ (cf. Fig. 9) amounts to $O(|senses(x.\ell)|^n \times |S_d(x)| \times |senses(x_i.\ell)|^n \times (|SN| \times \text{depth}(SN)) + O(|gloss|^2))$, where: i) *n* designates the total number of tokens within the target (and context) node label (usually $n=1$, i.e., the label consists of a single word, and sometimes $n=2$ when the label consists of a compound word made of two terms[15] without a single match in the *SN*), ii) $O(|SN| \times \text{depth}(SN))$ underlines the complexity of the edge-based and node-based semantic similarity measures adopted in our study; and iii) $O(|gloss|^2)$ is the time complexity of our gloss-based measure where *gloss* designates our extended gloss (including synonyms). Yet, given that *SN* designates a fixed reference throughout the disambiguation process (i.e., $|SN|$, $\text{depth}(SN)$, and $|gloss|$ remain unchanged), and given that the number of tokens per XML node label is usually $n=1$, $XSD_{Concept}$'s complexity simplifies to $O(|senses(x.\ell)| \times |S_d(x)| \times |senses(x_i.\ell)|)$.
  - The complexity of algorithm $XSD_{Context}$ (Fig. 10) amounts to $O(|senses(x.\ell)|^n \times (|S_d(x)| + n \times |S_d(s_p)|)$ where: i) $O(|S_d(x)| \times |S_d(s_p)|)$ designates the complexity of computing the cosine measure between context vectors $\overrightarrow{V_d(x)}$ and $\overrightarrow{V_d(s_p)}$, and ii) $n \times |S_d(s_p)|$ designates the size of vector $\overrightarrow{V_d(s_p)}$ extended to cover the sphere neighborhoods of *n* context node label tokens when context node labels consists of compound words. Yet, given that the number of tokens per XML node label is usually $n=1$, $XSD_{Context}$'s complexity simplifies to $O(|senses(x.\ell)| \times (|S_d(x)| + |S_d(s_p)|)$.
  - The complexity of algorithm $XSD_{Combined}$ consists of the sum of the complexities of $XSD_{Concept}$ and $XSD_{Context}$ and thus simplifies to: $O(|senses(x.\ell)| \times |S_d(x)| \times |senses(x_i.\ell)|) + O(|senses(x.\ell)| \times (|S_d(x)| + |S_d(s_p)|) = O\left(|senses(x.\ell)| \times \left(|S_d(x)| \times |senses(x_i.\ell)|\right) + \left(|S_d(x)| + |S_d(s_p)|\right)\right)$
  - When applied to all target nodes |X|, the complexity of our *XML semantic disambiguation* phase becomes: $\sum_{x \in X} O\left(|senses(x.\ell)| \times \left(|S_d(x)| \times |senses(x_i.\ell)|\right) + \left(|S_d(x)| + |S_d(s_p)|\right)\right) = O\left(|X| \times |senses(x.\ell)| \times \left(|S_d(x)| \times |senses(x_i.\ell)|\right) + \left(|S_d(x)| + |S_d(s_p)|\right)\right)$

[15] Recall that having more than two terms per XML tag name is unlikely [106].

Space complexity is similar to time complexity and simplifies (in worst case) to the same time complexity factor.

3.7. Parallel versus Incremental Disambiguation

On one hand, XSDF's architecture can be straightforwardly implemented on a parallel (multi-core and/or multi-thread) processing platform, since XML nodes targeted for disambiguation can be processed independent, and thus simultaneously (in parallel). With *parallel disambiguation*, complexity simplifies to $O\left(|senses(x.\ell)| \times \left(|S_d(x)| \times |senses(x_i.\ell)|\right) + \left(|S_d(x)| + |S_d(s_p)|\right)\right)$ time.

On the other hand, XSDF can also be straightforwardly extended toward *incremental disambiguation*, where: i) nodes targeted for disambiguation are processed one at a time, ii) ordered following their ambiguity degree, starting from the least (or most) ambiguous nodes, such that iii) the senses of initially disambiguated nodes are utilized in processing subsequent nodes. While time complexity remains unaffected here (i.e., time is linear w.r.t. the number of target nodes, cf. Section 3.6), yet, incremental disambiguation might affect disambiguation quality: previously disambiguated nodes (senses) might affect the result quality (senses selected) for later disambiguated ones.

In this study, we adopt the most basic XSDF design: i) non-parallel with ii) no incremental processing, and defer the evaluation of parallel and incremental disambiguation architectures to a dedicated upcoming study.

4. Experimental Evaluation

We have developed a prototype titled *XSDF*[16] to test and compare our approach with its most recent alternatives. We have evaluated three criteria: i) *semantic ambiguity*, ii) *disambiguation quality*, and iii) *time performance*.

4.1. Experimental Test Data

We used a collection of 80 test documents gathered from several data sources having different properties (cf. Table 4), which we describe and organize based on two features: i) *node ambiguity*, and ii) *node structure* (cf. Table 1). The former feature highlights the average amount of ambiguity of XML nodes in the XML tree, estimated using our *ambiguity degree* measure, *Amb_Deg* ∈ [0, 1]. The latter feature describes the average amount of structural richness of XML nodes, in terms of node *depth*, *fan-out*, and *density* in the XML tree, estimated as the sum of normalized node depth (1-Amb_{Depth}), fan-out, and density (1-$Amb_{Density}$) factors, averaged over all nodes in the XML tree, formally:

$$\text{Struct_Deg}(x, T) = \frac{w_{Depth} \times x.d}{\text{Max}(depth(T))} + \frac{w_{Fan\text{-}out} \times x.f}{\text{Max}(fan\text{-}out(T))} + \frac{w_{Density} \times \overline{x.f}}{\text{Max}(\overline{fan\text{-}out}(T))} \in [0,1] \quad \textbf{(20)}$$

where $w_{Depth} + w_{Fan\text{-}out} + w_{Density} = 1$ and $(w_{Depth}, w_{Fan\text{-}out}, w_{Density}) \geq 0$.

High node depth, fan-out, and/or density here indicate a highly structured XML tree, whereas low node depth, fan-out, and/or density indicate a poorly structured tree (relatively flat document).

In our experiments, we set equal weights $w_{Depth} = w_{Fan\text{-}out} = w_{Density} = 1/3$ when measuring *Struct_Deg* (Table 1). As mentioned previously, we do not address the issue of assigning weights, which could be performed using optimization techniques (e.g., linear programming and/or machine learning [38, 60, 66]) to help fine-tune input parameters and obtain optimal results (Section 3.3). Such a study would require a thorough analysis of the relative effect of each parameter on disambiguation quality, which we report to a dedicated study.

Table 1. XML test documents organized based on average node *ambiguity* and *structure*[17].

	Structure +	**Structure –**
Ambiguity +	**Group 1** *Amb_Deg* = 0.1127 & *Struct_Deg* = 0.6803	**Group 2** *Amb_Deg* = 0.1378 & *Struct_Deg* = 0.6621
Ambiguity –	**Group 3** *Amb_Deg* = 0.0625 & *Struct_Deg* = 0.612	**Group 4** *Amb_Deg* = 0.0447 & *Struct_Deg* = 0.5515

4.2. XML Ambiguity Degree Correlation

We compared XML node ambiguity ratings produced by human users with those produced by our system (i.e., via our *ambiguity degree* measure, *Amb_Deg*, cf. Section 3.3), using *Pearson's correlation coefficient*, $pcc = \delta_{XY}/(\sigma_X \times \sigma_Y)$ where: *x* and *y* designate user and system generated ambiguity degree ratings respectively, σ_X and σ_Y denote the standard deviations of *x* and *y* respectively, and δ_{XY} denotes the covariance between the *x* and *y* variables. The values of $pcc \in [-1, 1]$ such that: -1 designates that one of the variables is a decreasing function of the other variable (i.e., values deemed ambiguous by human users are deemed unambiguous by the system, and viceversa), 1 designates that one of the variables is an increasing function of the other variable (i.e., values are deemed ambiguous/unambiguous by human users and the system alike), and 0 means that the variables are not correlated. Five test subjects (two master students and three doctoral students, who were not part of the system development team) were involved in the experiment. For each (of the 80) test document(s) in our datasets, manual ambiguity ratings (in the form of integers ∈ [0, 4], i.e., ∈ [*min*, *max*] ambiguity) where acquired from each tester for 12-to-13 randomly pre-selected nodes[18], i.e., a total of 1000 nodes (during an average 10 hours rating time per tester). We first quantified cross-(human) tester agreement, by computing pair-wise correlation scores among testers for each of the rated datasets (cf. Table 2, Fig. 11, Fig. 12, and Fig. 13). One can realize that average cross-tester correlation levels are relatively low when rating datasets with *high ambiguity* (cf. Groups 1 and 2 in Table 2) and increase when rating datasets with *low ambiguity* (cf. Groups 3 and 4 in Table 2). On one hand, we realized that testers tend to agree less when rating ambiguous documents: certain testers provide high ambiguity scores for some XML node labels (considering them to be highly ambiguous) whereas other testers

[16] Available online at http://sigappfr.acm.org/Projects/XSDF/

[17] We organized documents based on *ambiguity* first, and *structure* second. This explains why *Group 2* has approximately the same *Struct_Deg* as *Group 3* (whose value is slightly smaller than that of *Group 2*): since *Group 2* has a higher *Amb_Deg* than *Group 3*. We adopted this organization since ambiguity is the foremost feature targeted in this study, which also allowed us to improve and facilitate the discussion of experimental results later.

[18] The human testers were provided: i) the source XML documents where the nodes targeted for ambiguity rating were highlighted, and ii) a document with instructions on how to rate, including a table containing all target nodes to which they should assign their ratings. Testers were instructed to check each node's context in the XML document while providing ratings, without however informing them of the nature/size/span of the context. This was done on purpose since we wanted our human testers to rely each on her/his own basic understanding/intuition about the "context" of a node (rather than follow any predefined mathematical concept) in providing her/his ratings. Sample test documents and ratings are provided online.

provide low ambiguity scores when rating the same node labels (considering them to be less ambiguous), hence yielding low tester correlation (cf. scores of Groups 1 and 2 in Table 2). One can also realize this discrepancy in human ratings among node labels of the same dataset/document: nodes of higher average ambiguity show higher discrepancy in individual tester ratings (e.g., labels *"line"* in Fig. 11.a and *"header"* in Fig. 12.a where tester ratings are significantly different) in comparison with labels of lower average ambiguity (e.g., labels *"scene description"* Fig. 11.a and *"item type"* in Fig. 12.a where tester ratings are almost identical). This was expected since it seemed normal that different human testers perceive the meanings of ambiguous nodes/documents differently. On the other hand, testers tend to agree more when rating documents which are less ambiguous: most testers provide similar scores for the same XML node labels (usually agreeing when labels are of low ambiguity, and sometimes agreeing when labels are of high ambiguity, cf. Fig. 13.a), hence yielding high tester correlation (cf. scores of Groups 3 and 4 in Table 2). On the whole, considering all inter-tester correlations over all datasets highlights an overall average correlation score of 0.411, which seemed a "reasonably" high inter-tester correlation given the diversity of the documents being rated (hence considering the obtained average tester ratings as a "reasonably" acceptable reference for empirical evaluation). Subsequently, we produced average human tester ratings per target XML node label in each dataset, which then were correlated with system ratings, computed with variations of *Amb_Deg*'s parameters to stress the impact of its factors ($Amb_{Polysemy}$, Amb_{Depth}, and $Amb_{Density}$): i) Test #1 considers all three factors equally ($w_{Polysemy} = w_{Depth} = w_{Density} = 1$), ii) Test #2 focuses on the polysemy factor ($w_{Polysemy} = 1$ while $w_{Depth} = w_{Density} = 0$), iii) Test #3 focuses on the depth factor ($w_{Depth} = 1$ while $w_{Polysemy} = 0.2$ and $w_{Density} = 0$), iv) Test #4 focuses on the density factor ($w_{Density} = 1$, $w_{Polysemy} = 0.2$ and $w_{Depth} = 0$).

Table 2. Correlation between human tester ratings.

		Testers										Avg.
		1-2	1-3	1-4	1-5	2-3	2-4	2-5	3-4	3-5	4-5	
Group 1	Dataset 1	-0.254	0.681	0.239	0.051	-0.283	-0.184	0.185	-0.104	0.498	-0.394	**0.043**
Group 2	Dataset 2	-0.028	0.580	-0.024	0.405	-0.160	0.333	0.051	-0.215	0.498	0.157	**0.16**
Group 3	Dataset 3	0.061	0.522	0.284	0.284	0.194	-0.287	-0.287	0.486	0.486	1	**0.274**
	Dataset 4	0.128	0.127	0.269	0.269	0.223	0.471	0.471	0.467	0.467	0.986	**0.388**
	Dataset 5	0.527	0.818	0.818	0.818	0.365	0.365	0.365	1	1	1	**0.707**
Group 4	Dataset 6	0.552	1	1	0.114	0.553	0.553	0.373	1	0.114	0.114	**0.537**
	Dataset 7	0.944	1	1	1	0.944	0.944	0.944	1	1	1	**0.977**
	Dataset 8	-0.219	0.277	0.178	-0.026	0.436	0.311	0.536	0.742	0.225	0.423	**0.288**
	Dataset 9	0.2282	-0.08	-0.083	-0.08	0.219	0.228	0.219	0.961	0.923	0.961	**0.35**
	Dataset 10	0.401	1	1	-0.123	0.401	0.401	-0.024	1	-0.123	-0.123	**0.381**

Table 3. Correlation between human ratings and system generated ambiguity degrees.

		Test #1 *All factors*	**Test #2** *Polysemy*	**Test #3** *Depth*	**Test #4** *Density*
Group 1	Dataset 1	0.394	0.411	0.335	**0.439**
Group 2	Dataset 2	**0.017**	0.181	0.243	0.139
Group 3	Dataset 3	-0.087	-0.139	-0.071	-0.138
	Dataset 4	0.408	0.438	0.390	0.398
	Dataset 5	-0.184	-0.185	-0.131	-0.235
Group 4	Dataset 6	-0.284	-0.291	-0.243	-0.316
	Dataset 7	-0.177	-0.190	-0.254	-0.143
	Dataset 8	-0.119	-0.025	0.033	-0.156
	Dataset 9	-0.452	-0.301	-0.251	**-0.456**
	Dataset 10	-0.258	0.180	0.412	0.276

Results are compiled in Table 3. Detailed manual and system ambiguity ratings concerning the three extreme correlations scores (*maximum*, closest to *null*, and *minimum* scores, highlighted in bold in Table 3) are shown in Figures 11-13.

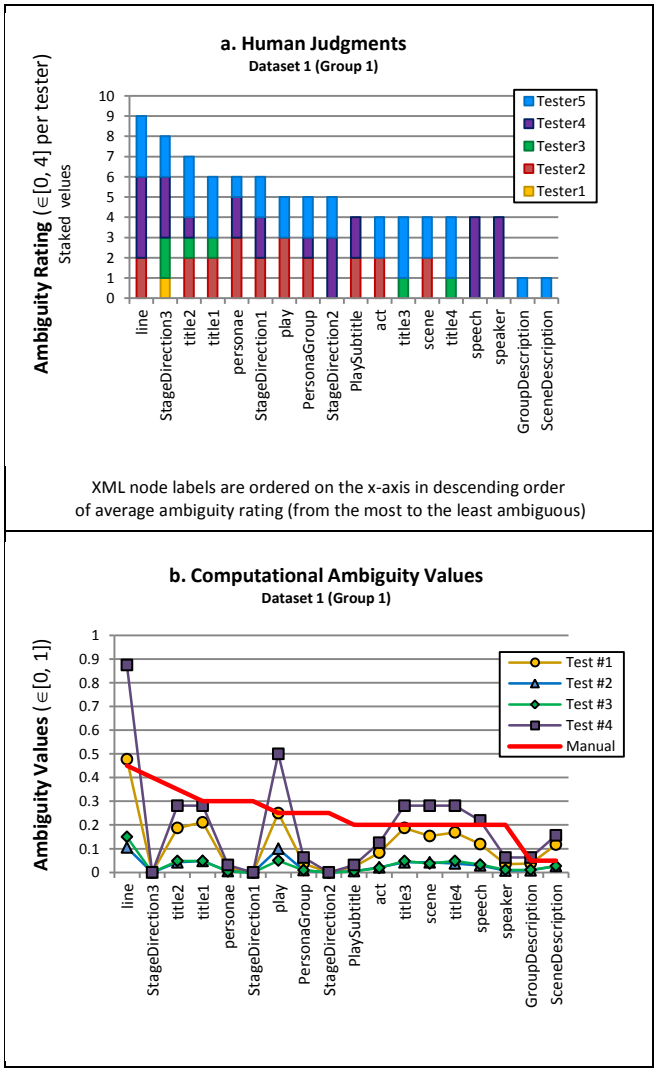


Fig. 11. Manual and system generated average ambiguity degrees highlighting maximum correlation with documents of Dataset 1 of Group 1. The *x* axis represents sample node labels (tag names/values) statistically selected to describe Dataset 1[19].

Results in Table 3 and Figures 11-13 show that the highest correlation scores are obtained when evaluating nodes of Dataset 1 in Group 1 (*high ambiguity* and *rich structure*), with a maximum *corr* = 0.439 with Test #4 (using the *density* factor of *Amb_Deg*). The lowest scores with all four tests are obtained with Dataset 9 in Group 4 (*low ambiguity* and *poor structure*), with a minimum *corr* = -0.456 with Test #4. Close to null scores are

[19] Node labels shown in the graphs of Figures 11, 12, and 13 are snapshots of those in the corresponding datasets, statistically sampled to represent correlation score variation between extremes (*maximum*, closest to *null*, and *minimum* scores) describing the node label distribution in each dataset.

obtained with Dataset 2 of Group 2 (*high ambiguity* and *poor structure*), Dataset 3 of Group 3 (*low ambiguity* and *rich structure*), and Dataset 8 of Group 4 (*low ambiguity* and *poor structure*), such that the closest to null score *corr* = 0.017 is obtained with Dataset 2 of Group 2 in Test #1 (combining all factors of *Amb_Deg*). The above results highlight several observations.

1) When ***highly ambiguous*** and ***highly structured*** XML nodes are involved (e.g., Group 1), XML ambiguity seems to be perceived and evaluated similarly by human users and our system, obtaining maximum positive correlation between human and *Amb_Deg* scores (cf. visualized sample in , cf. Fig. 11.b).

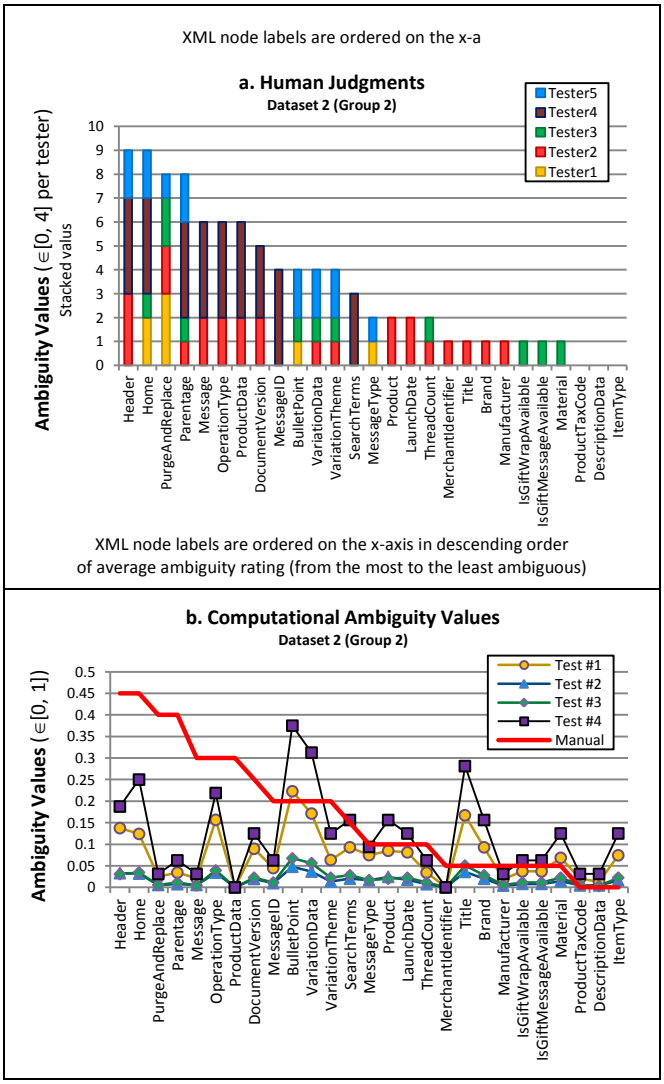


Fig. 12. Manual and system generated average ambiguity degrees highlighting closest to zero (null) correlation with Dataset 2 of Group 2.

2) When ***less ambiguous*** and/or ***poorly structured*** XML nodes are involved, ambiguity seems to be evaluated differently by users and our system, attaining: negative or close to null correlation when low ambiguity and/or poorly structured XML nodes are evaluated: i) negative correlation (opposite ambiguity scores) when low ambiguity and poorly structured XML nodes are evaluated (e.g., Group 4, cf. visualized sample in Fig. 13.b), ii) close to null correlation (broadly related or unrelated ambiguity scores) with either low ambiguity and/or poorly structured nodes (e.g., Groups 2, 3, and 4), and iii) varying correlation (ranging from positive to negative scores) when less ambiguous nodes are evaluated (e.g., Dataset 4 in Group 3 yields a positive correlation score, whereas Datasets 3 and 5 yield close to null and negative scores).

These contradictory and/or unrelated correlation scores are mainly due to the intuitive understanding of semantic meaning by humans, in contrast with the intricate processing done by our automated system. For instance, regarding documents of Dataset 9 of Group 4 (conforming to the *personnel.dtd* grammar of the Niagara XML document collection, cf. Fig. 14), the meaning of child node label "*state*" under node label "*address*" was obvious for our human testers (providing an ambiguity score of 0/4). Yet, the interpretation of the meaning of "*state*" is not so obvious for a machine, especially using a rich lexical dictionary such as WordNet where word "*state*" has 8 different meanings. Here, a label considered relatively unambiguous from the user's point of view was assigned a higher ambiguity score by the system based on the expressiveness of the lexical reference.

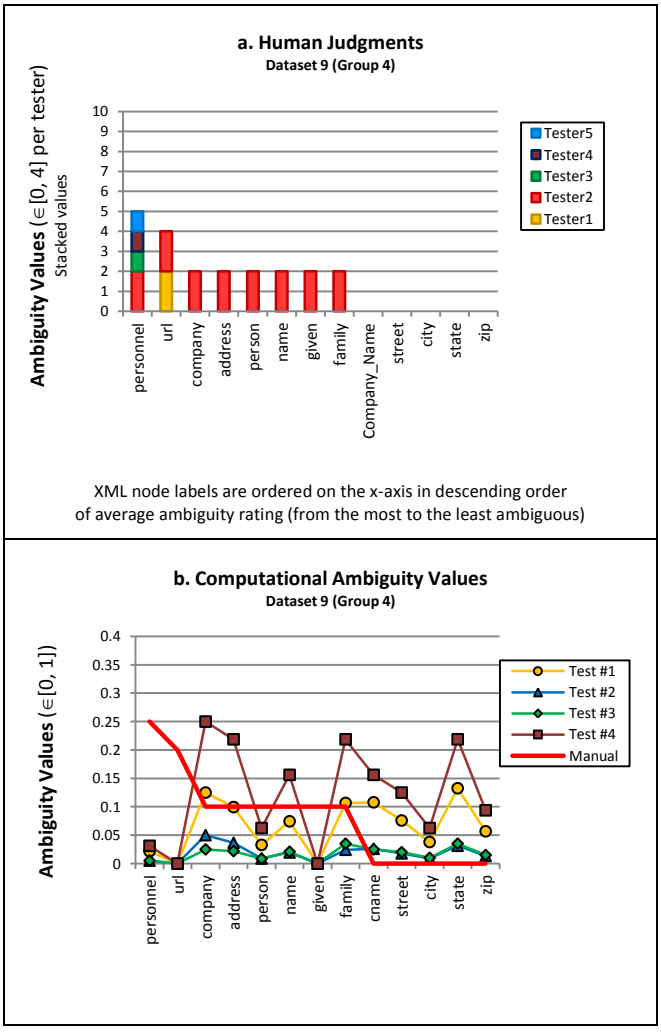


Fig. 13. Manual and system generated average ambiguity degrees highlighting minimum correlation with Dataset 9 of Group 4 (Fig. 14).

Table 4. Characteristics of test documents.

Groups	Datasets	Source dataset	Grammar	N# of docs	Avg. N# of nodes per doc	Node label polysemy (N# of senses)		Node Depth		Node Fan-out		Node Density	
						Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
Group 1	1	Shakespeare collection[20]	shakespeare.dtd	10	192.054	7.052	30	3.687	6	0.604	20	0.38	6
Group 2	2	Amazon product files[21]	amazon_product.dtd	10	113.333	8.407	72	4.309	7	0.539	13	0.38	6
Group 3	3	SIGMOD Record[22]	ProceedingsPage.dtd	6	39.375	4.615	16	2.743	6	0.692	9	0.692	9
	4	IMDB database[23]	movies.dtd	6	15.475	4	10	2.666	5	1.066	5	1	5
	5	Niagara collection[24]	bib.dtd	8	26.5	4.384	13	2.961	5	0.884	5	0.884	5
Group 4	6	W3Schools[25]	cd_catalog.dtd	4	16.5	3.937	10	2.312	3	0.812	6	0.812	6
	7	W3Schools	food_menu.dtd	4	16	2.375	7	2.437	3	0.562	4	0.562	4
	8	W3Schools	plant_catalog.dtd	4	11.675	3.454	15	2	3	1.181	6	1.181	6
	9	Niagara collection	personnel.dtd	4	19	3.947	9	2.368	5	1.157	4	1.157	4
	10	Niagara collection	club.dtd	4	15.5	4.533	10	2.266	4	1.4	5	1.4	5

3) Concerning *Amb_Deg*'s ***parameter weight variations*** (for $w_{Polysemy}$, w_{Depth}, and $w_{Density}$) with tests 2-4, all three parameters seem to have comparable impacts on ambiguity evaluation. This can be clearly seen in Table 3 where similar correlation scores are obtained for each dataset when running Tests #2-4.

```
<?xml version="1.0"?>
<company>
    <cname id="id3_4">Informix</cname>
    <address>
        <street>123 6th Ave W</street>
        <city>Portland</city>
        <state>OR</state>
        <zip>54678</zip>
    </address>
    <personnel>
        <person>
            <name>
                <given>Fran</given>
                <family>Car</family>
            </name>
            <url>http://null</url>
        </person>
    </personnel>
</company>
```

```
<?xml encoding="ISO-8859-1"?>
<!ELEMENT company (address, cname,
                   personnel)>
<!ATTLIST comapny id ID #REQUIRED>
<!ELEMENT address (street, city, state, zip)>
<!ELEMENT personnel (person)+>
<!ELEMENT person (name, email?, url?)>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT name (family?|given?)>
<!ELEMENT cname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

Fig. 14. Sample XML document from Dataset 9 of Group 4, with corresponding grammar.

Note that certain parameters naturally have more impact in certain cases, based on the nature of the XML data involved. For instance, XML files with uneven distributions of depth, such as those in Dataset 2 of Group 2 (Amazon products file), containing nodes of depth 0 or 1 and others of depth 9, produce the highest correlation score with Test #3 (0.243) which evaluates the *depth* factor of *Amb_Deg*.

Likewise, XML files with uneven distributions of density, such as those in Dataset 1 of Group 1 (Shakespeare plays), in which there are nodes with 0 or 1 distinct children and others with 6 distinct children, produce the highest correlation score with Test #4 (0.439) which evaluates the *density* factor of *Amb_Deg*. Note that when XML documents do not contain serious disparities in XML structure (i.e., *depth* and *density* factors are almost homogeneous across all nodes), the *polysemy* factor would naturally have the highest impact in evaluating ambiguity, such as with Dataset 4 of Group 3 (IMDB movies file) where maximum correlation is reached with Test #2 evaluating the *polysemy* factor of *Amb_Deg*.

Note that evaluating XML node ambiguity is not addressed in existing methods (they do not select target nodes, but rather disambiguate all of them, which can be complex and needless).

4.3. XML Semantic Disambiguation Quality

In addition to evaluating our XML ambiguity degree measure, we ran a series of experiments to evaluate the effectiveness of our XML disambiguation approach. We used the same test datasets described previously. Target XML nodes were first subject to manual disambiguation (12-to-13 nodes were randomly pre-selected per document yielding a total of 1000 target nodes, allowing the same human testers to manually annotate each node by choosing appropriate senses from WordNet, which required on average 22 hours per tester) followed by automatic disambiguation. We then compared user and system generated senses to compute *precision*, *recall* and *f-value* scores.

4.3.1. Testing with Different Configurations

We first tested the effectiveness of our approach considering its different features and possible configurations, considering: i) the properties of XML data (w.r.t. ambiguity and structure), ii) context size (sphere neighborhood radius), and iii) the disambiguation process used (concept-based, context-based, and the combined approaches). Note that when applying the concept-based ($XSD_{Concept}$) and the combined ($XSD_{Combined}$) disambiguation algorithms, semantic similarity measures were considered with identical parameter weights ($w_{Edge} = w_{Node} = w_{Gloss} = 1/3 = 0.3334$). Note that in this study, we do not address the issue of assigning semantic similarity weights, which could help the user fine-tune the latter algorithms to obtain optimal results. Such a study would require a dedicated analysis of the relative effect of each individual measure on concept-based disambiguation, which is out of the scope of this paper. In the following, we show average *f-value* results in Fig. 15 (*precision* and *recall* levels follow similar patterns and are omitted for clarity). Several interesting observations can be made here.

1) Considering ***XML data properties***, our approach produced consistent *f-value* levels $\in$ [0.55, 0.69] over all the tested configurations. The highest levels were reached with XML nodes of Dataset 1 of Group1 having *high ambiguity* and *rich structure*, which resonates with the node ambiguity results discussed in the previous section: highly ambiguous and structurally rich XML nodes seem to be most effectively processed by our approach, in comparison with less ambiguous/structurally poor nodes.

[20] Available at http://metalab.unc.edu/bosak/xml/eg/shaks200.zip
[21] Available at simply-amazon.com/content/XML.html
[22] Available at http://www.acm.org/sigmod/xml
[23] Data extracted from http://www.imdb.com/ using a wrapper generator.
[24] Available at http://www.cs.wisc.edu/niagara/
[25] Available from http://www.w3schools.com

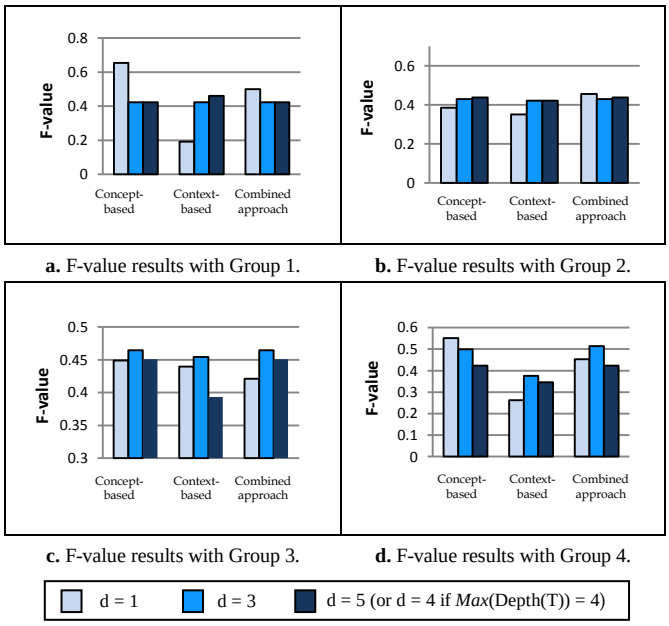


Fig. 15. Average *f-value* scores considering different features and configurations of our approach[26].

2) Considering ***context size,*** optimal *f-value* levels are obtained with the smallest sphere neighborhood radius d=1 with Group 1 (*high ambiguity* and *rich structure* XML nodes), whereas optimal levels are obtained with larger contexts having d=3 with Groups 2, 3, and 4 (*low ambiguity* and/or *poor structure*). This is expected since increasing context size with highly ambiguous/structure rich XML would increase the chances of including noise (e.g., unrelated/heterogeneous XML nodes) in the disambiguation context and thus disrupt the process. Yet, increasing context size with less ambiguous/poorly structured XML could actually help in creating a large-enough and/or rich-enough context to perform effective disambiguation.

In both cases, the above results emphasize the need for a flexible approach (such as ours), allowing the user to fine-tune context size based on the nature and properties of the data in order to optimize disambiguation results.

3) Considering the ***disambiguation process***, one can realize that the *concept-based* approach ($XSD_{Concept}$) generally produces higher *f-value* levels in comparison with the *context-based* approach ($XSD_{Context}$), the latter appearing to be more sensitive to context size (especially with Groups 2, 3, and 4). This is expected since algorithm $XSD_{Context}$ primarily depends on the notion of context and context nodes, in both the XML document and semantic network, and thus increasing/decreasing context size would disturb its effectiveness. The effect of context size here could be aggravated when using a rich semantic network (such as WordNet) where a small increase in sphere neighborhood radius could include a huge number of concepts (synsets) in the semantic network context vector, thus adding considerable noise. Yet, algorithm $XSD_{Concept}$ seems less sensitive to varying context sizes since it largely focuses on individual context nodes and their possible senses: i) even with a small number of context nodes (small context size), the number of combination of possible senses would be enough to provide good quality disambiguation, and ii) increasing the number of context nodes (i.e., by increasing context size) increases the number of possible sense combinations, which does not necessarily (or drastically) reduce quality since the best sense combination (i.e., the right sense) can still be identified with reduced noise effect.

Results also show that the combined approach, using $XSD_{Combined}$, seems to provide a good compromise, emphasizing (once more) the usefulness of having a tunable approach allowing the user to adapt the process following her needs.

4.3.2. Comparative Study

In order to further evaluate our approach, we have conducted a comparative study to assess its effectiveness w.r.t. related XML disambiguation methods. A qualitative comparison is shown in Table 6. In short, XSDF: i) considers both XML structure (tag names) and content (element/attribute values), ii) identifies and selects ambiguous XML nodes as targets for disambiguation, iii) considers *sphere neighborhood* as comprehensive XML context model including all XML structural relationships within a certain radius from the target node, iv) allows the user to choose context size (radius) following her needs, v) represents sphere neighborhoods as context vectors following a relational information model considering structural proximity between XML nodes, vi) introduces a hybrid approach combining two disambiguation algorithms ($XSD_{Concept}$ and $XSD_{Context}$), allowing the user to fine-tune disambiguation parameters following her needs. On the other hand, existing approaches: i) only consider XML structure (disregarding element/attribute values) [57, 58, 95, 106], ii) do not address the issue of automatically selecting target nodes for disambiguation [57, 58, 95, 106], iii) consider XML contexts with partial coverage of XML data such as with parent node [97, 98], sub-tree [106] or root path models [95], iv) are static in that context size is pre-defined [95, 97, 98, 106] and cannot be adapted by the user, v) represent contexts as sets of nodes using the *bag-of-words* paradigm [95, 106], disregarding structural proximity among nodes, vi) use static disambiguation algorithms which cannot be easily tuned by the user [57, 58, 95, 106].

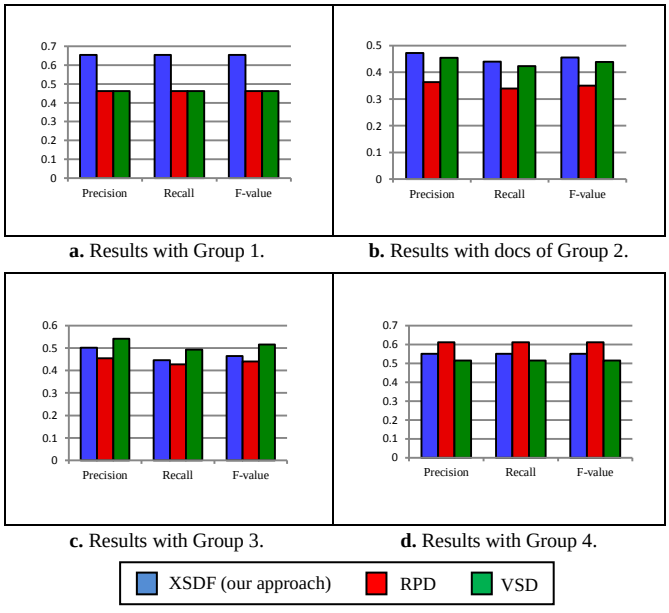
We have also experimentally compared our method's effectiveness with two of its most recent alternatives, i.e., *RPD* (Root Path Disambiguation) [95], and *VSD* (Versatile Structure Disambiguation) [57][27]. We ran a battery of experiments on the same experimental data groups (described in Section 4.1), considering the same target XML nodes (considered in Section 4.3.1). Here, we provide a compiled presentation considering optimal input parameters for our approach (i.e., context size d=1 when processing Groups 1 and 4 using the *concept-based* disambiguation process, d=1 when processing Group 2 using the *combined* approach, and d=3 when processing Group 3 using the *combined*

[26] When applying the context-based ($XSD_{Context}$) and combined ($XSD_{Combined}$) disambiguation algorithms, XML and semantic network sphere neighborhood radiuses d and d' were tied such that $d' = d$, since our objective here was to evaluate the effect of increasing/decreasing context size (mainly in the XML document), regardless of the nature of the context itself.

[27] *RPD* and *VSP* were developed within standalone disambiguation approaches (including the whole pipeline from linguistic pre-processing to sense mapping) which we could compare with our approach. In contrast, we could not compare our method with the *parent node context* [97] and *sub-tree context* [106] approaches since they were not developed as standalone disambiguation processes, but were integrated within specific applications (i.e., XML semantic search and document classification respectively).

approach)[28], as well as optimal input parameters for each of the alternative disambiguation approaches, as indicated in their corresponding studies. Results in Fig. 16 show that our method yields *precision*, *recall*, and *f-value* levels higher than those achieved by its predecessors, with almost all test cases except for two: *VSD* produces better results with Group 3 (Fig. 16.c), and *RPD* produces better results with Group 4 (Fig. 16.d).

Here, one can realize that XML nodes in Groups 3 and 4 are less ambiguous and poorly structured in comparison with the first two test groups (cf. Table 1). As a result, choosing a reduced context made of root path nodes (with *VSD*) or select/reachable context nodes (with *RPD*) has proven to be less noisy (including less context nodes) in these cases, in comparison with the more comprehensive context models used with our approach.



a. Results with Group 1. **b.** Results with docs of Group 2.

c. Results with Group 3. **d.** Results with Group 4.

Fig. 16. Average *precision*, *recall* and *f-value* scores comparing our approach with RPD [95] and VSD [57].

One can also realize that our method produces highest *precision*, *recall*, and *f-value* levels with Group 1 (*high ambiguity* and *rich structure* XML nodes), with average 44% improvement over *RDP* and *VSD* (Fig. 16.a), in comparison with average 15% with Group 2, and almost 0% improvements with Groups 3 and 4[29]. This concurs with our results of the previous section: our method is more effective when dealing with highly ambiguous nodes within a rich XML structure, in comparison with less ambiguous/poorly structured XML. Overall *precision*, *recall*, and *f-value* results in Table 5 averaged over all four groups of XML test data, confirm our method's improvement in comparison with *RPD* and *VSD*.

Table 5. *Precision*, *recall*, and *f-value* results averaged over all groups of XML documents (Fig. 16), comparing our approach with its alternatives.

	Precision	***Recall***	***F-value***
XSDF	0.5447	0.5226	0.5312
RPD [95]	0.3946	0.4737	0.4828
VSD [57]	0.4728	0.4598	0.4659

[28] Manually identified from repeated tests with different parameter values.

[29] We compute the average improvement (or deterioration) of *XSDF* over both *RPD* and *VSD*, for each of the test groups.

4.4. Performance Evaluation

In addition to testing the effectiveness of our approach in identifying correct mappings, we evaluated its efficiency in terms of execution time.

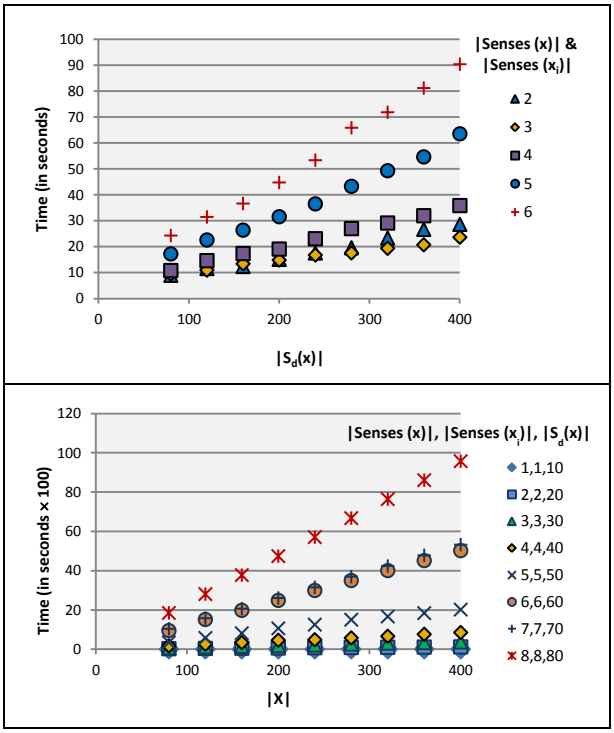


Fig. 17. Timing results regarding our *concept-based* approach (when executing algorithm $XSD_{Concept}$, cf. Fig. 9).

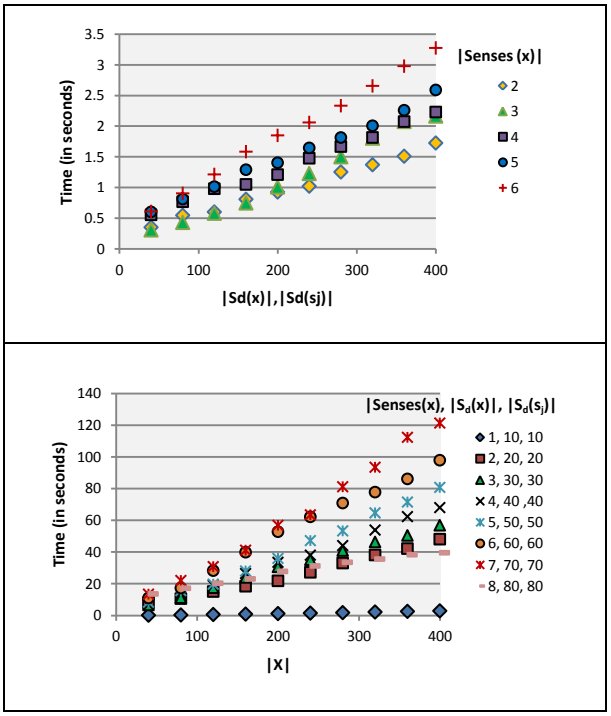


Fig. 18. Timing results regarding our *context-based* approach (when executing algorithm $XSD_{Context}$, cf. Fig. 10).

Table 6. Comparing our method with existing approaches.

Approaches	Considers linguistic pre-processing	Considers tag tokenization (compound terms)	Addresses XML node ambiguity	Integrates an inclusive XML structure context	Flexible w.r.t. context size	Adopts *relational information* approach	Combines the results of various semantic similarity measures	Straightforward mathematical functions	Disambiguates XML structure and content
RPD [95]	√	x	x	x	x	x	x	x	x
VSD [57]	√	√	x	√	√	√	x	x	x
XSDF (our approach)	√	√	√	√	√	√	√	√	√

Results highlight the linear complexities of both our concept-based approach (using algorithm $XSD_{Concept}$) i.e., $O(|senses(x.\ell)| \times |S_d(x)| \times |senses(x_i.\ell)|)$ (Fig. 17) and our context-based approach (using algorithm $XSD_{Context}$), i.e., $O(|senses(x.\ell)| \times (|S_d(x)| + |S_d(s_p)|)$ (Fig. 18). Time is also linear w.r.t. the number of target nodes being disambiguated, designated as $|X|$.

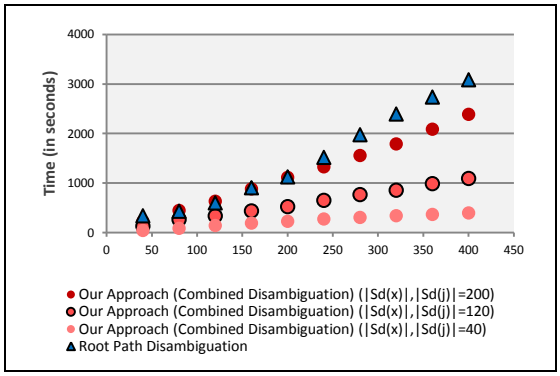


Fig. 19. Comparing time results with existing *Root Path Disambiguation* approach [95].

We have also compared the time complexity of our approach, using different configurations, with one of its most recent predecessors. Results in Fig. 19 show closely correlated and even reduced time results (depending on the configuration used, e.g., smaller context radiuses in XML document and/or in the reference semantic network). This means that our approach was able to produce improved disambiguation quality while preserving (and sometimes reducing) execution time levels in comparison with its alternatives[30].

A future goal is to extend our algorithms' designs and implementations, using XML-based parallel processing techniques, namely micro- and macro-level processing architectures (using bit-level, data-level, and/or instruction-level parallelism) [105], aiming to further increase the processing speed of our disambiguation process. A snapshot of the experimental study (along with sample test documents and user ratings) is available online[31].

5. Applications

In this section, we discuss some of the main application scenarios which can benefit, in one way or another, from XML semantic analysis and disambiguation. Most of these applications are built around methods for *XML structure* and *semantic similarity* evaluation, e.g., [4, 99, 102], i.e., comparing the structural positions of XML element/attribute nodes in the XML document tree while considering the semantic similarities between node labels and/or values. In this context, developing semantic-aware applications usually requires three main steps:

a. *XML semantic disambiguation*: an initial pre-processing step to identify the intended meanings of node labels and/or values,
b. *XML similarity evaluation*: comparing semantically augmented XML trees w.r.t. the meanings of node labels/values identified in the initial step,
c. *Semantic-aware processing*: an application specific step, where semantic-aware processing is undertaken based on XML semantic similarity evaluation.

Accordingly, in this section, we present an overview of such applications which we gradually look at from different angles, starting from i) the layer of abstraction at which XML similarity is evaluated, and then describing high-end application domains covering: iii) Information Retrieval, iv) Web and Mobile Services, and v) the (Social) Semantic Web.

5.1. XML Similarity at Different Abstraction Layers

XML semantic similarity evaluation can take place at three different abstraction levels: i) the data layer (i.e., document/document comparison), i) the type layer (i.e., document/grammar comparison), and ii) in-between the data and type layers [12], each underlining its own battery of semantic-aware applications.

5.1.1 Similarity at the XML Data Layer

Performing XML document/document comparison, is relevant in a variety of applications (cf. reviews in) [4, 102], such as data *versioning*, *monitoring*, and *temporal querying*: a user may want to view or access a version of a certain document (e.g., an XHTML Web site, a Web Service SOAP description, an RSS feed, etc.) which was available during a certain period of time, or may want to view the results of a continuous query, or monitor the evolution of a certain document in time. Such tasks can be implemented using semantic-aware *tree edit distance* similarity measures which produce, along with the similarity score, an *edit script* consisting of a set of *edit operations* describing semantic changes to the disambiguated data (e.g., inserting/deleting/updating semantically related nodes, to transform one XML document tree into another) [102]. Another application is *document clustering*, i.e., grouping XML documents together, based on their structural and semantic similarities, which can improve *data storage indexing* [80] and thus positively affect the *data retrieval* process [4]. Also, clustering is central is *information extraction*, *wrapping*, and *summarization*, allowing to automatically identify the sets of semantically similar XML elements to be extracted from documents to be reformulated (e.g., substituting disambiguated node labels with semantically related ones), restructured, or summarized, making them easier to process in enterprise applications (e.g., adapting/simplifying the content of a Web page, blog, RSS feed, or Web Service description for non-experts) [39, 92].

[30] Note that we did not compare execution time with the *Versatile Structure Disambiguation* approach [58] since we were unable to acquire the system implementation from the authors. We used the authors' online version of the prototype, which is relatively slow due to network access, and thus could not use it to evaluate processing time.
[31] http://sigappfr.acm.org/Projects/XSDF/

5.1.2. Similarity at the XML Type Layer

Performing XML grammar/grammar comparison, is also useful for many tasks (cf. reviews in) [24, 86], namely *data integration*, which consists in: i) comparing (matching) grammars to identify semantically related elements [103], and then ii) merging the matched elements within a unified grammar or semantic view [90]. Here, a disambiguation step is necessary to capture the meaning of grammar elements prior to performing grammar matching. Data integration allows the user to efficiently access and acquire more complete information (e.g., accessing similar Websites, blogs, or RSS feeds concurrently) [91]. It is also essential in performing *data warehousing*[32], where XML information is transformed from different data-sources complying with different grammars into data conforming with grammars defined in the data warehouse [25]. Other applications include *message translation* in Business-to-Business (B2B) integration [15]: reconciling the semantics of XML message grammars used by trading partners in order to translate in-coming and out-going messages accordingly, which is essential in E-commerce and B2B applications [44]; and XML *data maintenance* and *schema evolution*: detecting the structural and semantic differences/updates between different versions of a given grammar to revalidate corresponding XML documents [48].

5.1.3. Similarity between XML Data and Type Layers

Performing XML document/grammar comparison, can also benefit from XML disambiguation applied on the documents and grammars being compared, highlighting various applications (cf. review in) [99]. One such application is *XML document classification*, i.e., categorizing XML documents gathered from the Web against a set of grammars declared in an XML repository. Here, evaluating semantic similarity between incoming disambiguated documents on one hand, and reference disambiguated grammars on the other hand (e.g., defined in the repository), allows the identification of entities that are conceptually close, but not syntactically identical, which is common in handling heterogeneous XML sources, particularly on the Web where users have different backgrounds and no precise definitions about the matter of discourse [11]. Evaluating semantic similarity between documents and grammars can also be exploited in XML document retrieval via *structural queries* [89]: a query being represented as an XML grammar with additional constraints on content; as well as in the *selective dissemination of information:* user profiles being expressed as grammars against which the incoming XML document stream is matched [88].

5.2. Information Retrieval

Information Retrieval (IR) is one of the leading application domains requiring sophisticated semantic-aware and similarity-based processing where systems aim at providing the most relevant (similar) documents w.r.t. a user information need expressed as a search query. In this context, a wide range of techniques extending traditional IR systems to handle XML IR have been designed (cf. reviews in) [76, 102]. In brief, XML IR systems accept as input: i) a user query: expressed as an XML document [74], an XML fragment [16], an XML structured query (e.g., XPath or XQuery [13]), or as a set of keywords [116], and ii) an indexed XML document repository [52], and produce as output: a ranked list of XML elements (and their sub-trees)[33] selected form the repository, and ordered following their relevance (similarity) w.r.t. the user query [98]. Hence, the quality of an XML IR engine depends on two key issues: i) how documents and queries are represented (indexed), and ii) how these representations are compared (matched) to produce relevant results. Here, most solutions in the literature have explored syntactic XML indexing paradigms (based on node positions, paths, or structural summaries) integrated in dedicated inverted indexing structures, e.g., [52, 112].

Yet, as XML data on the Web became increasingly available and diverse, element/attribute labels and values became *noisier*, such that syntactic indexing techniques could not keep pace [27]. As a result, non-expert users have been increasingly faced with what is described as the *vocabulary problem* [30]: query keywords chosen by users are often different from those used by the authors of the relevant documents, lowering the systems' *precision* and *recall* rates. This highlights the need for XML disambiguation, where both the XML query and documents can be processed and represented using semantic concepts, instead of (or in addition to) syntactic keywords and element names/values (e.g., typical XML indexing techniques can be used, except that element names/values would be replaced with semantic concepts) [51]. then, query/document matching can be performed in the semantic concept space, instead of performing syntactic keyword/node label matching, thus extending XML IR toward *semantic XML IR*, or so-called *concept-based XML IR* [27]. Preliminary studies on *semantic XML IR* have shown that representing documents and queries using semantic concepts usually results in a retrieval model that is more effective and less dependent on the specific terms/node labels used, significantly increasing search *precision* and *recall* rates [85]. Note that semantic XML IR, and ontological (RDF/OWL[34]) IR, is presently a hot research topic [76, 85].

5.3. Web and Mobile Services

Another interesting application area which requires XML semantic disambiguation is the matching, search, and composition of Web Services (WS). A WS comes down to a self-contained, modular application that can be described, published, and invoked over the Internet, and executed on the remote system where it is hosted [82]. WS rely on two standard XML schemata: i) WSDL (Web Service Description Language) [20] allowing the definition of XML grammar structures to support the machine-readable description of a service's interface and the operations it supports, and ii) SOAP (Simple Object Access Protocol) [113] for XML-based communications and message exchange among WS endpoints. RESTful services have been recently promoted as a simpler alternative to SOAP and WSDL-based WS [109]: communicating over HTTP using HTTP request methods (e.g., *Get*, *Post*, *Put*, etc., instead of exchanging SOAP messages), and using XHTML or free test to describe the services (instead of WSDL) [78].

Hence, when searching for WS (or RESTful services) achieving specific functions, XML (or XHTML/keyword) based service requests can be issued, to which are then matched and ranked service WSDL (or XHTML/keyword) descriptions, thus identifying those services answering the desired requests. Here, matching and ranking service descriptions requires effective XML semantic analysis and disambiguation techniques, due to service author/user heterogeneity (same as the *vocabulary problem* in XML IR, cf. Section 5.2). The same applies for services discovery, recommendation, and composition: searching and mapping together semantically similar WSDL/SOAP descriptions when processing WS, and performing semantic-aware mapping of XHTML/keyword

[32] A warehouse is a decision support database that is extracted from a set of data sources (e.g., different databases describing related data).

[33] Selecting a whole XML document as a potential answer comes down to selecting its root node (along with the corresponding sub-tree).

[34] Refer to Section 5.4.

descriptions when dealing with RESTful and/or mobile services [56, 115]. XML similarity and differential encoding can also be utilized to reduce processing costs and improve the performance of SOAP communications [104]: comparing new SOAP messages with message patterns or WSDL grammars (at the message sender/receiver side), processing and transmitting only those parts of the messages which are different (cf. review in) [105].

5.4. Semantic Web and Social Semantic Web

Above all, the Semantic Web vision [10] benefits from most of the above-mentioned applications, as it inherently requires XML disambiguation to deal with the semantics of Web documents (encoded in XML-based format), to improve interoperability between systems, ontologies, and users. Typically, XML disambiguation can be utilized in *ontology learning*, to build domain taxonomies [108] and enrich/update large-scale semantic networks [72], based on user input data streams (e.g., Web pages, blogs, image annotations, etc.) encoded in XML. In this context, technologies such as RDF (Resource Description Framework) [59]), and OWL (Web Ontology Language) [63] can be used to construct such ontologies.

RDF enables the definition of statements specifying relationships between instances of data in the form of < *subject*, *predicate*, *object* > triples, which designate that a resource (i.e., the subject) has a property (i.e., the predicate) whose value is a resource or a literal (i.e., the object). OWL is built on top of RDF and adds additional expressiveness which depends on the type of *Description Logic* (DL) language applied (OWL allows different levels of semantic expressiveness, ranging from OWL-Lite, to OWL-DL and OWL-Full [32]). As a result, RDF and OWL highlight the concept of *Linked Data*: the seamless connection of pieces of information and knowledge on the Semantic Web [36], where a given resource (i.e., subject) can be associated with new properties (i.e., objects) via new relationships (i.e., predicates), and where additional statements (i.e., triples) can be easily added to describe resources and properties [32]. In this context, XML disambiguation is essential: to extract the semantic information form XML data so that it can be utilized or integrated with semantic annotations from: i) reference ontologies, ii) previously annotated (disambiguated) XML documents, or iii) user generated annotations (e.g., social tagging). Practical examples include integrating hotel and airline reservations, order processing, and insurance renewal with social networking information [55]. Also, augmenting Web data (in XML) with semantic annotations (i.e., triples) provides a way of blending traditional information with Linked Data and Semantic Web constructs [53].

An emerging trend in this context is the integration of user information (e.g., user annotations, hash-tags, search queries, and selected search results), i.e., so-called *social semantics* [85], to semantically augment Web (XML-based) data. This underlines the concept of the *Social Semantic Web* [39, 85], a Web in which social interactions lead to the creation of collective and collaborative knowledge representations such as Wikipedia, Yahoo Answers, and Flikr, providing semantic information based on human contributions and paving the way for various new applications ranging over: i) *blog classification*, e.g., introducing simple and effective methods to semantically classify blogs [87], ii) *social semantic network analysis*, e.g., disambiguating entities in social networks, and identifying semantic relationships between users [9], and iii) *socio-semantic information retrieval*, e.g., considering user information to improve/adapt Web data indexing, query formulation, result ranking, and result presentation techniques [73] (cf. reviews in [22, 85]).

6. Conclusion

This paper introduces a novel XML Semantic Disambiguation Framework titled *XSDF*, to semantically annotate XML documents with the help of machine-readable lexical knowledge base (e.g., WordNet), which is a central pre-requisite to various applications ranging over semantic-aware query rewriting [21, 68], XML document classification and clustering [94, 101], XML schema matching [24, 103], Web and mobile services' discovery, recommendation, and composition [46, 56, 115], and blog analysis and event detection in social networks [3, 9]. *XSDF* covers the whole disambiguation pipeline from: i) linguistic pre-processing of XML node labels to handle compound words (neglected in most existing solutions), to ii) selecting ambiguous nodes for disambiguation using a dedicated *ambiguity degree* measure (unaddressed in most solutions), iii) representing target node contexts as comprehensive and flexible (user chosen) *sphere neighborhood* vectors (in contrast with partial and fixed context representations, e.g., parent node or sub-tree context), and iv) running a hybrid disambiguation process, combining two (user chosen) methods: *concept-based* and *context-based* (in contrast with static methods). Experimental results reflect our approach's effectiveness in selecting ambiguous XML nodes and identifying node label senses w.r.t. user judgments of semantic ambiguity. Comparative theoretical and experimental analyses highlight our approach's effectiveness in comparison with existing methods. Time analysis underlines the linear impact of context size and polysemy (number of senses) among other factors, on disambiguation time.

As continuing work, we are currently investigating different XML tree node distance functions (including edge weights, density, direction, and the semantic similarity between parent/child nodes, etc. [31, 41]), to define more sophisticated neighborhood contexts. Fine-tuning user parameters using dedicated optimization techniques [38, 60, 66] is another work in progress. We are also investigating the issue of semantic indexing [19], aiming to put forward a dedicated XML indexing approach (based on structural indexing [26, 83], or extended vector-space representations [16, 74]), in order to allow effective and efficient XML semantic search and retrieval. We are also investigating the use of additional/alternative lexical knowledge sources such as Google [42], Wikipedia [23], and FOAF [3] to acquire a wider word sense coverage, and thus explore our approach in practical applications, namely wiki document clustering, semantic blog analysis, and event detection in heterogeneous and collective social data [100]. In the near future, we aim to explore non-traditional processor architectures, including XML-based micro- and macro-level parallel processing solutions [105], to increase the processing speed of our disambiguation process. Comparing parallel and incremental disambiguation strategies would naturally follow.

Acknowledgements. This study is partly funded by the National Council for Scientific Research (CNRS), Lebanon, project: NCSR_00695_01/09/15, and by LAU grant: SOERC1415T004.

References

[1] Abiteboul S., Buneman P., and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publisher (1st Edition), 1999. pp. 258.

[2] Agirre, E., Martinez D., Lopez De Lacalle O., and Soroa A., *Two Graph-based Algorithms for State of the art Word Sense Disambiguation.* In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 2006. pp. 585–593.

[3] Aleman-Meza B., Nagarajan M., Ding L., Sheth A.P., Arpinar I.B., Joshi A., and Finin T.W., *Scalable Semantic Analytics on Social Networks for Addressing the Problem of Conflict of Interest Detection.* ACM Transaction on the Web (TWeb), 2008. 2(1):7.

[4] Algergawy A., Mesiti M., Nayak R., and Gunter Saake, *XML Data Clustering: An Overview.* ACM Computing Survey 2011. 43(4):25.

[5] Amitay E.; Nelken R.; Niblack W.; Sivan R. and Soffer A., *Multi-Resolution Disambiguation of Term Occurrences.* Proceedings of the ACM Conference on Information and Knowledge Management (CIKM), 2003. pp. 255-262.

[6] Artiles J.; Penas A. and Verdejo F., *Word Sense Disambiguation based on Term to Term Similarity in a Context Space.* In Senseval-3: Third International Workshop on the Evaluation of Systems, 2004. pp. 58-63.

[7] Banerjee S. and Pedersen T., *An adapted Lesk algorithm for word sense disambiguation using WordNet.* In Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, 2002.

[8] Banerjee S. and Pedersen T., *Extended Gloss Overlaps as a Measure of Semantic Relatedness.* International Joint Conference on Artificial Intelligence (IJCAI'03), 2003. p. 805-810.

[9] Berendt B.; Hotho A. and Stumme G., *Bridging the Gap - Data Mining and Social Network Analysis for Integrating Semantic Web and Web 2.0.* Journal of Web Semantics, 2010. 8(2-3): 95-96.

[10] Berners-Lee T., Hendler J., and Lassila O., *The Semantic Web.* Scientific American, 2001. 284(5):1:19.

[11] Bertino E., Guerrini G., and Marco Mesiti, *Measuring the structural similarity among XML documents and DTDs.* Journal of Intelligent Information Systems, 2008. 30(1):55-92.

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

[13] Boncz P. A., Grust T., Van Keulen M., Manegold S., Rittinger J., and Teubner J., *MonetDB/XQuery: a fast XQuery processor powered by a relational engine.* . International ACM SIGMOD Conference, 2006. pp. 479-490.

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

[15] Cardoso J. and Bussler C., *Mapping between heterogeneous XML and OWL transaction representations in B2B integration.* Journal of Data & Knowledge Engineering, 2011. 70(12): 1046-1069

[16] 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.* Proceedings of the ACM SIGIR Workshop on XML and Information Retrieval, 2002. pp. 14-25.

[17] Chan Y. S.; Ng H. T. and Zhong Z., *NUS-PT: Exploiting parallel texts for word sense disambiguation in the English all-words tasks.* In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval, Prague, Czech Republic), 2007b. pp. 253-256.

[18] Charbel N., Tekli J., Chbeir R., and Tekli G., *Resolving XML Semantic Ambiguity.* International Conference on Extending Database Technology (EDBT'15), 2015. Brussels, Belgium, pp 277-288.

[19] Chbeir R., Luo Y., Tekli J., Yetongnon K., Ibanez C.R., Traina A.J.M., Traina C., and Al Assad M., *SemIndex: Semantic-Aware Inverted Index.* 18th East-European Conference on Advanced Databases and Information Systems (ADBIS'14), 2014. pp. 290-307.

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

[21] Cimiano P.; Handschuh S. and Staab S., *Towards the Self-Annotating Web.* In Proceedings of the International World Wide Web Conference (WWW'04), 2004. pp. 462-471.

[22] d'Aquin M., Motta E., Sabou M., Angeletou S., Gridinoc L., Lopez V., and Guidi D., *Toward a New Generation of Semantic Web Applications.* IEEE Intelligent Systems, 2008. 23(3):20–28.

[23] Dandala B., Hokamp C., Mihalcea R., and Bunescu R.C., *Sense Clustering using Wikipedia.* Recent Advances in Natural Language Processing (RANLP'13), 2013. pp. 164-171.

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

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

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

[27] Egozi O., Markovitch S., and Gabrilovich E., *Concept-Based Information Retrieval Using Explicit Semantic Analysis.* ACM Transactions on Information Systems 2011. 29(2):8.

[28] Fellbaum C., *Wordnet: An Electronic Lexical Database.* MIT Press, Cambridge, MA, 1998. 422 p.

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

[30] Furnas G., Landauer T.K., Gomez L.M., and Dumais S., *The vocabulary problem in human-system communication.* . Communications of the ACM, 1987. 30(11):964–971.

[31] Ganesan P.; Garcia-Molina H.; and Windom J., *Exploiting Hierarchical Domain Structure To Compute Similarity.* ACM Transactions on Information Systems (TOIS), 2003. 21(1):64-93.

[32] Garcia-Castro R. and Gomez-Perez A., *Interoperability Results for Semantic Web Technologies using OWL as the Interchange Language.* Journal of Web Semantics (JWS), 2010. 8(4):278-291.

[33] Graupmann J.; Schenkel R. and Weikum G., *The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents.* Proceedings of the International Conference on Very Large Databases (VLDB), 2005. pp. 529-540.

[34] Guo Y.; Che W.; Hu Y.; Zhang W. and Liu T., *HIT-IR-WSD: A WSD System for English Lexical Sample Task.* SemEval 2007, ACL.

[35] Hachey B., Radford W., Nothman J., Honnibal M., and Curran J.R., *Evaluating Entity Linking with Wikipedia.* Artificial Intelligence, 2013. 194:130-150.

[36] Heath T. and Bizer C., *Linked Data: Evolving the Web into a Global Data Space.* Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool, 2011.

[37] Hoffart J., Suchanek F.M., Berberich K., and Weikum G., *YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia.* Artif. Intell., 2013. 194: 28-61.

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

[39] Hovy E.H., Navigli R., and Ponzetto S.P., *Collaboratively built semi-structured content and Artificial Intelligence: The story so far.* Artificial Intelligence, 2013. pp. 194: 2-27.

[40] Ide N. and Veronis J., *Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art.* Computational Linguistics, 1998. 24(1):1-40.

[41] Jiang J. and Conrath D., *Semantic Similarity based on Corpus Statistics and Lexical Taxonomy.* Proceedings of the International Conference on Research in Computational Linguistics, 1997. pp 19-33.

[42] Klapaftis I. and Manandhar S., *Evaluating Word Sense Induction and Disamiguation Methods.* Language Resourses and Evaluation, 2013. 47(3):579-605.

[43] Krovetz R. and Croft W. B., *Lexical Ambiguity and Information Retrieval.* ACM Transactions on Information Systems, 1992. 10(2):115-141.

[44] Lampathaki F., Mouzakitis S., Gionis G., Charalabidis Y., and Askounis D., *Business to business interoperability: A current review of XML data integration standards.* Computer Standards & Interfaces, 2009. 31(6):1045-1055.

[45] Leacock C. and Chodorow M., *Combining Local Context and WordNet Similarity for Word Sense Identification.* FellBaum C. editor, WordNet: An Electronic Lexical Database, Chapter 11, The MIT Press, Cambridge, 1998. pp. 265-283.

[46] Lecue F. and Mehandjiev N., *Seeking Quality of Web Service Composition in a Semantic Dimension.* IEEE Trans. on Knowledge and Data Engineering (TKDE), 2011. pp. 942-959.

[47] 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.

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

[49] Lesk M., *Automatic Sense Disambiguation using Machine Readable Dictionaries: How to tell a Pine Cone from an Ice Cream Cone.* In Proceedings of the 5th Annual Inter. Conference on Systems Documentation (SIGDOC'86), 1986.

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

[51] Lu J., *An Introduction to XML Query Processing and Keyword Search.* Springer-Verlag Berlin Heidelberg, 2013. pp. 292.

[52] Luk R., Leong H.V., Dillon T., Chan A., Croft W.B., and Allan J., *A Survey in Indexing and Searching XML Documents.* Journal of American Society for Information Science and Technology, 2002. 53(6):415-437.

[53] MacManus R., *How Best Buy is using The Semantic Web.* The New York Times, 2010. July 1st.

[54] Maguitman A., Menczer F., Roinestad H., and Vespignani A., *Algorithmic Detection of Semantic Similarity.* Proceedings of the International Conference on the World Wide Web (WWW), 2005. pp. 107-116.

[55] Malaika S. and Nicola M., *Data normalization reconsidered: An examination of record keeping in computer systems.* Developer Works, IBM Corporation 2010. pp. 32.

[56] Malki A., Barhamgi M., Benslimane S.M., Benslimane D., and Malki M., *Composing Data Services with Uncertain Semantics.* IEEE Trans. on Knowledge and Data Engineering (TKDE), 2015. pp. 936-949.

[57] Mandreoli F. and Martoglia R., *Knowledge-based sense disambiguation (almost) for all structures.* Information Systems, 2011. 36(2): 406-430.

[58] Mandreoli F., Martoglia R., and Ronchetti E., *Versatile Structural Disambiguation for Semantic-Aware Applications.* In Proceedings of the ACM International Conf. on Information and Knowledge Management, 2005. pp. 209-216.

[59] Manola F. and Miller E., *Resource Description Framework (RDF) Primer : Model and Syntax Specification.* W3C Recommendation, 2004. http://www.w3.org/TR/rdf-primer/.

[60] Marie A. and Gal A., *Boosting Schema Matchers.* In Proceedings of the OTM 2008 Confederated International Conferences, 2008. pp. 283 – 300.

[61] Marquez L.; Escudero G.; Martinez D. and Rigau G., *Supervised corpus-based methods for WSD. In Word Sense Disambiguation: Algorithms and Applications.* E. Agirre and P. Edmonds, Eds. Springer, New York, NY, 2006. pp. 167-216.

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

[63] McGuinness D.L. and Van Harmelen F., *OWL 2 Web - Ontology Language Document Overview.* W3C Proposed Edited Recommendation, 2012. http://www.w3.org/TR/owl2-overview/.

[64] Mihalcea R., *Knowledge-based Methods for WSD.* In Word Sense Disambiguation: Algorithms and Applications, E. Agirre and P. Edmonds, 2006. pp. 107–131, Springer, New York.

[65] Miller G.A.; Leacock C.; Tengi R. and Bunker R.T., *A Semantic Concordance.* In Proceedings of the ARPA Workshop on Human Language Technology, 1993. pp. 303–308.

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

[67] Navigli R., *Word Sense Disambiguation: a Survey.* ACM Computing Surveys, 2009. 41(2):1–69.

[68] Navigli R. and Velardi P., *An Analysis of Ontology-based Query Expansion Strategies.* In proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI'03), 2003. pp. 42-49.

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

[70] Patwardhan S.; Banerjee S. and Pedersen T., *Using Measures of Semantic Relatedness for Word Sense Disambiguation.* In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'03), 2003. pp. 241-257.

[71] Patwardhan S.; Banerjee S. and Pedersen T., *SenseRelate:TargetWord – A Generalized Framework forWord Sense Disambiguation.* Proceedings of the 20th National conference on Artificial intelligence, 2005. Volume 4 (AAAI'05), Anthony Cohn (Ed.), Vol. 4. AAAI Press 1692-1693. .

[72] Pennacchiotti M. and Pantel P., *Ontologizing semantic relations.* Proceedings of the 44th Association for Computational Linguistics (ACL) Conf. joint with the 21th Conf. on Computational Linguistics (COLING), 2006. pp. 793–800.

[73] Peters, I., *Folksonomies. Indexing and Retrieval in Web 2.0.* De Gruyter, 2009. pp. 443.

[74] Pokorny J. and Rejlek V., *A Matrix model for XML Data.* Chap. in: Databases and Information Systems, Selected Papers from the Sixth International Baltic Conference DB&IS´2004, V. 118 Frontiers in Artificial Intelligence and Applications, Ed. J. Barzdins and A. Caplinskas, 2005. IOS Press, pp. 53-64.

[75] Pradhan S.; Loper E.; Dligach D. and Palmer M., *Semeval-2007 task-17: English lexical sample, SRL and all words.* In Proceedings of the 4th InternationalWorkshop on Semantic Evaluations (SemEval, Prague, Czech Republic), 2007. pp. 87-92.

[76] Renteria-Agualimpia W., López-Pellicer F.J., Muro-Medrano P.R., Nogueras-Iso J., and Zarazaga-Soria F.J., *Exploring the advances in semantic search engines.* International Symposium on Distributed Computing and Artificial Intelligence, in: Advances in Intelligent and Soft Computing, Springer, Berlin, Heidelberg, 2010. vol. 79, pp. 613–620.

[77] Resnik P., *Disambiguating Noun Groupings with Respect to WordNet Senses.* In Proceedings of the 3rd Workshop on Large Corpora, 1995. pp. 54-68.

[78] Richardson L. and Ruby S., *RESTful Web Services.* O'Reilly Media, Inc., 2008. pp. 454.

[79] Richardson R. and Smeaton A., *Using WordNet in a Knowledge-based approach to information retrieval.* Proceedings of the BCS-IRSG Colloquium on Information Retrieval, 1995.

[80] Rusu L.I., Rahayu J.W., and Taniar D., *Storage Techniques for Multi-versioned XML Documents.* Database Systems for Advanced Applications (DASFAA'08), 2008. pp. 538-545.

[81] Saari P. and Eerola T., *Semantic Computing of Moods Based on Tags in Social Media of Music.* . IEEE Trans. on Knowledge and Data Engineering (TKDE), 2014. pp. 2548-2560.

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

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

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

[85] Schoefeggera K., Tammetb T., and Granitzerc M., *A survey on socio-semantic information retrieval.* Computer Science Review, 2013. 8:25–46.

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

[87] Singh A. K. and Joshi R.C., *Semantic Tagging and Classification of Blogs.* International Conference on Computer and Communication Technology (ICCCT), 2010, 2010. pp. 455-459.

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

[89] Staworko S. and Chomicky J., *Validity-Sensitive Querying of XML Databases.* Current Trends in Database Technology – EDBT 2006, DataX'06, 2006. Lecture Notes in Computer Science, Springer, vol. 4254/2006, pp. 164–17.

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

[91] Taddesse F.G., Tekli J., Chbeir R., Viviani M., and Yetongnon K., *Semantic-based Merging of RSS Items.* World Wide Web Journal: Internet and Web Information Systems Journal Special Issue: Human-Centered Web Science., 2010. Vol. 12 (No. 11280), Springer Netherlands.

[92] Tagarelli A. and Greco S., *Semantic Clustering of XML Documents.* ACM Transactions on Information Systems, 2010. 28(1):3.

[93] Tagarelli A., Longo M., and Greco S., *Word Sense Disambiguation for XML Structure Feature Generation.* In Proceedings of the European Semantic Web Conference, 2009. LNCS 5554, pp. 143–157.

[94] Tagarelli A. and Greco S., *Semantic Clustering of XML Documents.* ACM Transactions on Information Systems, 2010. 28(1):3.

[95] Tagarelli A.; Longo M. and Greco S., *Word Sense Disambiguation for XML Structure Feature Generation.* In Proceedings of the European Semantic Web Conference, 2009. LNCS 5554, pp. 143–157.

[96] Taha K. and Elmasri R., *OOXKSearch: A Search Engine for Answering XML Keyword and Loosely Structured Queries Using OO Techniques.* J. Database Manag., 2009. 20(3):18-50.

[97] Taha K. and Elmasri R., *CXLEngine: A Comprehensive XML Loosely Structured Search Engine.* Proceedings of the EDBT workshop on Database Technologies for Handling XML Information on the Web (DataX'08), 2008. pp. 37-42, Nantes, France.

[98] Taha K. and Elmasri R., *XCDSearch: An XML Context-Driven Search Engine.* IEEE Transactions on Knowledge and Data Engineering, 2010. 22(12):1781-1796.

[99] Tekli J, Agma J.M. Traina, Caetano Traina Jr., and Chbeir R., *XML Document-Grammar Comparison: Related Problems and Applications.* Central European Journal of Computer Science, 2011. Inaugural Issue, 1(1):117-136.

[100] Tekli J., Abou Rjeily A., Chbeir R., Tekli G., Houngue P., Yetongnon K., and Ashagrie Abebe M., *Semantic to intelligent web era: building blocks, applications, and current trends.* . International Conference on Managment of Emergent Digital EcoSystems (MEDES), 2013. pp. 159-168.

[101] Tekli J., Chbeir R., and Yetongnon K., *A Novel XML Structure Comparison Framework based on Sub-tree Commonalities and Label Semantics.* Elsevier Journal of Web Semantics (JWS): Science, Services and Agents on the World Wide Web, 2012. 11: 14-40.

[102] 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.

[103] Tekli J., Chbeir R., and Yétongnon K., *Minimizing User Effort in XML Grammar Matching.* Elsevier Information Sciences Journal, 2012. 210:1-40.

[104] Tekli J., Damiani E., and Chbeir R., *Using XML-based Multicasting to Improve Web Service Scalability.* International Journal on Web Services Research (IJWSR), 2012. 9(1):1-29.

[105] Tekli J., Damiani E., Chbeir R., and Gianini G., *SOAP Processing Performance and Enhancement.* IEEE Transactions on Services Computing (IEEE TSC), 2012. 5(3): 387-403.

[106] Theobald M.; Schenkel R. and Weikum G., *Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data.* In Proceedings of the ACM SIGMOD International Workshop on Databases (WebDB), 2003. pp. 1-6, San Diego, California.

[107] Tratz S.; Sanfilippo A.; Gregory M.; Chappel A.; Posse C. and Whitney P., *PNNL: A supervised maximum entropy approach to word sense disambiguation.* In Proceedings of the 4th InternationalWorkshop on Semantic Evaluations (SemEval, Prague, Czech Republic), 2007. pp. 264-267.

[108] Velardi P., Faralli S., and Navigli R., *OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction.* Computational Linguistics, 2013. 39(3): 665-707.

[109] Verma R. and Srivastava A., *A Novel Web Service Directory Framework for Mobile Environments.* . IEEE International Conference on Web Services (ICWS'14), 2014. pp. 614-621.

[110] Veronis J., *Hyperlex: Lexical cartography for information retrieval*. Comput. Speech Lang., 2004. 18(3):223–252.

[111] W3 Consortium. *The Document Object Model*. 2005 [cited 28 May 2009]; Available from: http://www.w3.org/DOM.

[112] Wang H. and Meng X., *On the sequencing of tree structures for XML indexing*, . In Proceedings of the International Conference on Data Engineering (ICDE'05), 2005. pp. 372–383.

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

[114] Wu Z. and Palmer M., *Verb Semantics and Lexical Selection.* Proceedings of the 32nd Annual Meeting of the Associations of Computational Linguistics, 1994. pp. 133-138.
[115] Xia X., Wang X., and Zhou X., *Evolving Recommender System for Mobile Apps: A Diversity Measurement Approach.* Smart Computing Review, 2013. 3(3):139-154.
[116] Xiang Y., Deng Z., Yu H., Wang S., and Gao N., *A New Indexing Strategy for XML Keyword Search.* Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010), 2010.
[117] Yaworsky D., *Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora.* Proceedings of the Inter. Conference on Computational Linguistics (COLING), 1992. Vol 2, pp. 454-460. Nantes.
[118] Zhang X.; Jing L.; Hu X.; Ng M. and Zhou X., *A Comparative Study of Ontology Based Term Similarity Measures on PubMed Document Clustering.* Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA' 07), 2007. pp. 115-126.
[119] Zhang Z., Li R., Cao S., and Zhu Y., *Similarity Metric in XML Documents.* Knowledge Management and Experience Management Workshop, 2003.



Joe Tekli is an Assistant Professor in the ECE Department, Lebanese American University (LAU). He holds a PhD in Computer Science from the Univ. of Bourgogne (UB), LE2I CNRS, France (2009), awarded with Highest Honors. He has completed three post-docs: in the Univ. of Milan, Italy (Fall 2009), in the Univ. of Shizuoka, Japan (Spring 2010), and in ICMC, Univ. of Sao Paulo (USP), Brazil (2010-2011). He was awarded various prestigious fellowships: French Ministry of Education (France), Fondazione Cariplo (Italy), JSPS (Japan), FAPESP (Brazil), and AUF (France). His research covers XML; MM data semantics, data-mining, and information retrieval. He is a member of IEEE and ACM SIGAPP, and an organizing member of various international conferences such as IEEE ICWS, AIAI, SITIS, ACM MEDES, etc. He has more than 30 publications in various prestigious journals and conferences: Elsevier JWS , IEEE TSC, WWW J., Elsevier Info. Sciences, IJWSR, IEEE ICWS, EDBT, ER, WISE, ADBIS, etc.



Nathalie Charbel is a Ph.D. student in the University of Pau et des Pays des Adours (France). She's currently working in the T2i team of the LIUPPA laboratory (France), in collaboration with Nobatek, a private technology center, applying research to innovative services in the fields of sustainable development and construction. She holds a Master's degree (M2) in Database and Artificial Intelligence from the University of Bourgogne (France) and a Master's degree in Telecommunications and Computer Engineering from the Antonine University (Lebanon), both acquired with high distinction (September 2013). Her main areas of research interest are Semantic Web, XML Similarity Measures, XML Semantic Disambiguation, Data Mining, Data Visualization and 3D Modeling. Her first research results have been published in the international EDBT 2015 conference, in a scientific paper entitled "Resolving XML Semantic Ambiguity".



Richard Chbeir received his PhD in Computer Science from the University of INSA deLyon, France in 2001 and then his Habilitation degree in 2010 from the University of Bourgogne, France. He is currently a Full Professor in the Computer Science Department in IUT de Bayonne in Anglet, France. His current research interests are in the areas of multimedia information retrieval, XML and RSS Similarity, access control models, and digital ecosystems. Richard Chbeir has published in international journals, books, and conferences, and has served on the program committees of several international conferences. He is currently the Chair of the French Chapter ACM SIGAPP.