Building Semantic Trees from XML Documents
Joe Tekli, Nathalie Charbel, Richard Chbeir

To cite this version:

HAL Id: hal-02079156
https://hal-univ-pau.archives-ouvertes.fr/hal-02079156
Submitted on 25 Mar 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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 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
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.

- **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.

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], 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
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]) 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]:

$$\text{Sim}_{\text{edge}}(c_1, c_2, SN) = \frac{2N_u}{N_1 + N_2 + 2N_u} \in [0,1]$$  \hspace{1cm} (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 $N_u$ 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]:

$$\text{Sim}_{\text{node}}(c_1, c_2, SN) = \frac{2 \log p(c_0)}{\log p(c_1) + \log p(c_2)} + \log p(c_1) \in [0,1]$$

having $p(c_i) = \frac{\text{Freq}(c_i)}{N}$

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]:

$$\text{Sim}_{\text{gloss}}(c_1, c_2, SN) = (\text{gloss}(c_1) \cup \text{gloss}(rel(c_1))) \cap (\text{gloss}(c_2) \cup \text{gloss}(rel(c_2)))$$

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.

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.

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 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 descendant (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, descendant, or both) belong to the target node’s context:

\[
\text{weight}(n_c, n) = 2 \times \frac{e^{-\frac{d^2}{2}}}{\sqrt{2\times\pi}} + 1 - \frac{2}{\sqrt{2\times\pi}} \tag{4}
\]

where \( n_c \) is a context node, \( n \) is the target node, and \( d \) is the distance (in number of edges) separating \( n_c \) from \( n \) 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].

![XSDF Architecture](image)

**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, $T[i].f$ its label, $T[i].d$ its depth (in number of edges), and $T[i].o$ its out-degree (i.e., the node’s fan-out). $R(T) = \{0\}$ designates the root node of tree $T$.

**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 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 beneficial 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.

**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])

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
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 \(t_2)\) 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.

This is also true of one single concept in the semantic network (i.e., a synset in WordNet), with symbolic 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).

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:

\[
Amb_{Polysemy}(x. \ell, SN) = \frac{\text{senses}(x. \ell) - 1}{\text{Max}(\text{senses}(SN)) - 1} \in [0, 1] \tag{5}
\]

where \( \text{Max}(\text{senses}(SN)) \) is the maximum number of senses of a word/expression in \( SN \) (e.g., in WordNet 3.0 [28], \( \text{Max}(\text{senses}(SN)) = 33 \), 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., \( \ell \) 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 \( x. \ell = \text{Max}(\text{senses}(SN)) \).
- When \( \text{senses}(x. \ell) \) 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:

\[
Amb_{Depth}(x, T) = 1 - \frac{x.d}{\text{Max}(\text{depth}(T))} \in [0, 1] \tag{6}
\]

where \( \text{Max}(\text{depth}(T)) \) 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 \( x.d = \text{Max}(\text{depth}(T)) \), 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 \( \tilde{x.f} \), is low, or else it decreases such that:

\[
Amb_{Density}(x, T) = 1 - \frac{\tilde{x.f}}{\text{Max}(\text{fan-out}(T))} \in [0, 1] \tag{7}
\]

where \( \text{Max}(\text{fan-out}(T)) \) 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 \( \tilde{x.f} = 0 \), 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 \( \tilde{x.f} = \text{Max}(\text{fan-out}(T)) \), 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 \( \tilde{x.f} \) 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 \( 1-Amb_{Depth}(x, T) \) and \( 1-Amb_{Density}(x, T) \) on the other hand:

\[
Amb_{Deg}(x, T, SN) = \frac{w_{Polysemy} \times Amb_{Polysemy}(x. \ell, SN)}{w_{Depth} \times (1-Amb_{Depth}(x, T)) + w_{Density} \times (1-Amb_{Density}(x, T)) + 1} \in [0, 1] \tag{8}
\]

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-\( Amb_{Depth} \) and 1-\( Amb_{Density} \)) to counter their usage as denominator factors: varying proportionally with \( Amb_{Deg} \). 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 $\text{Amb}_\text{Deg} = 0$ is obtained when the label of node $x$ has only one possible sense, i.e., when $\text{Amb}_\text{Polysem}(x, t, SN) = 0$, regardless of its depth and density factors, and regardless of parameter weights (Assumption 0 and Lemma 1).

The maximum value $\text{Amb}_\text{Deg} = 1$ is obtained when: i) $x, t$ has the maximum number of senses in $SN$ (e.g., 33 senses in WordNet [28]), ii) $x$ is the root node of $T$, and $d = 0$, and iii) $x$ has the minimum number of children nodes with distinct labels in $T$, $\overline{x, j} = 0$, regardless of $w_{\text{Depth}}$ and $w_{\text{Density}}$ parameter values (Assumptions 1-3, and Lemmas 1-3).

The value of $\text{Amb}_\text{Deg}$ increases with: i) the increase in $x, t$’s polysemy in $SN$, ii) the decrease in $x$’s depth, and iii) the decrease in $x$’s density in the XML document tree. Conversely, the value of $\text{Amb}_\text{Deg}$ decreases with: i) the decrease in $x, t$’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 $\text{Amb}_\text{Deg}(x)$ as the average of the ambiguity degrees of $t_1$ and $t_2$.

$\text{Amb}_\text{Deg}$ 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 $\text{Thresh}_\text{amb}$ automatically estimated or set by the user, i.e., nodes having $\text{Amb}_\text{Deg}(x, T, SN) \geq \text{Thresh}_\text{amb}$ whereas remaining nodes are left untouched. Note that the user can disregard the ambiguity degree measure: i) by setting $w_{\text{Polysem}} = 0$ so that all nodes end up having $\text{Amb}_\text{Deg} = 0$ regardless of polysemy, depth, and density factors, or ii) by setting $\text{Thresh}_\text{amb} = 0$ 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., $w_{\text{Polysem}} = w_{\text{Depth}} = w_{\text{Density}} = 1$, with a minimal threshold $\text{Thresh}_\text{amb} = 0$ 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][7], 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., $R_d(x) = \{x_i \in T \mid \text{Dist}(x, x_i) = d\}$. $R_d(x)$ is more generally noted $R_{rel}(x)$ 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](Image)

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

The distance between two XML nodes in an XML tree, $\text{Dist}(x_i, x_j)$, 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 $R_2(T[2])$ 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., $S_d(x) = \{x_i \in T \mid x_i \in R_{d_j}(x) \land d_j \leq d\}$.


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
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 \(R_{drel}(c)\) 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).

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

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

\[
Freq(\ell_r, S_d(x)) = \sum_{S_d(x) \ni x_i} \text{Struct}(x_i, S_d(x)) \in \left[ \frac{1}{d+1}, \frac{|S_d(x)|+1}{2} \right] \tag{9}
\]

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

\[
\text{Struct}(x_i, S_d(x)) = 1 - \frac{\text{Dist}(x, x_i)}{d+1} \in \left[ \frac{1}{d+1}, 1 \right] \tag{10}
\]

The denominator in \(\text{Struct}(x_i, S_d(x))\) 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 \(\overline{V_d(x)}\), 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:

\[
w_{V_d(x)}(\ell_r) = \frac{Freq(\ell_r, S_d(x))}{\text{MaxFreq}} = \frac{2 \times Freq(\ell_r, S_d(x))}{|S_d(x)|+1} \in [0,1] \tag{11}
\]

\[9\text{ This is also similar to the notion of term frequency (or TF) in information retrieval, where the importance of a given term } \ell \text{ in describing a document } D \text{ increases with the frequent occurrence of } \ell \text{ in } D \{43, 62\}.\]
where \( \text{MaxFreq} = \frac{S(x)+1}{2} \) following formula (9).

For instance, given the XML tree in Fig. 6, Fig. 8 shows context vectors of sphere neighborhoods \( S(T[2]) \) and \( S(T[3]) \) centered around node \( T[2] \) of label "Cast." Considering \( V_j(T[2]) \):

\[
-w_{V_j(T[2])}^{\text{Cast}}(\text{"Cast"}) = 2 \times (\frac{1}{4} + 1) = 0.4 \text{ given that: i) "Cast" is the label of } \text{ } S_j(T[2])'s \text{ target (center) node } T[2], \text{ i.e., } \text{Struct}(T[2], S_j(T[2])) = 1, \text{ ii) } T[2] \text{ is the only node occurrence of label "cast" in } S_j(T[2]), \text{ i.e., } \text{Freq}(\text{"Cast"}, S_j(T[2])) = 1, \text{ and iii) } |S_j(T[2])| = 4.
\]

\[
-w_{V_j(T[2])}^{\text{Star}}(\text{"Star"}) = \frac{2 \times (0.5 + 0.5)}{(4) + 1} = 0.4 \text{ given that: i) } S_j(T[2]) \text{ contains two nodes } T[3] \text{ and } T[5] \text{ of label "Star" at distance = 1 from the target node } T[2], \text{ i.e., } \text{Struct}(T[3], S_j(T[2])) = 1 - \frac{1}{2} = 0.5, \text{ ii) } T[3] \text{ and } T[5] \text{ are the only node occurrences of label "Star" in } S_j(T[2]), \text{ i.e., } \text{Freq}(\text{"Star"}, S_j(T[2])) = 0.5 + 0.5 = 1, \text{ and iii) } |S_j(T[2])| = 4.
\]

Similarly for remaining context vector weight computations.

<table>
<thead>
<tr>
<th>Cast</th>
<th>Picture</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_j(T[2]) )</td>
<td>0.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cast</th>
<th>Picture</th>
<th>Star</th>
<th>Films</th>
<th>Stewart</th>
<th>Kelly</th>
<th>Plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_j(T[2]) )</td>
<td>0.2223</td>
<td>0.1492</td>
<td>0.2964</td>
<td>0.0741</td>
<td>0.0741</td>
<td>0.0741</td>
</tr>
</tbody>
</table>

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 \( V_j(T[2]) \)). \( w_{V_j(T[2])}^{\text{Cast}}(\text{"Cast"}) = 2 \times w_{V_j(T[2])}^{\text{Star}}(\text{"Star"}) \) since node label "Star" occurs twice in \( S_j(T[2]) \) while "Picture" occurs once; similarly in \( V_j(T[2]) \). Formally:

**Lemma 5:** The context vector weight measure \( w_{V_j(x)}(x_j) \) in Definition 11 varies in accordance with Assumptions 5 and 6.

**Proof of Lemma 5:** Given formula (11), \( w_{V_j(x)}(x_j) \) varies as:

- \( w_{V_j(x)}(x_j) \) increases with \( \text{Freq}(x_j, S_j(x)) \), which increases with \( \text{Struct}(x_j, S_j(x)) \):
  - The value of \( \text{Struct}(x_j, S_j(x)) \) is inversely proportional to the distance between the target node \( x \) and context node \( x_j \) (following Assumption 1).
  - The minimum value \( \text{Struct}(x_j, S_j(x)) = \frac{1}{d + 1} \) is reached when \( x_j \in R_d(x) \) where \( R_d(x) \) is the outermost ring in sphere \( S_j(x) \).
  - The maximum value \( \text{Struct}(x_j, S_j(x)) = 1 \) is reached when processing the target node itself, i.e., \( x_j = x \).
  - The value of \( \text{Freq}(x_j, S_j(x)) \) is proportional to the number of occurrences of nodes having the same label \( x_j \) (following Assumption 2).
  - The minimum value \( \text{Freq}(x_j, S_j(x)) = \frac{1}{d + 1} \) is reached when there is only one context node having label \( x_j \) and occurring on the outer-most ring \( R_d(x) \) of the sphere neighborhood, more formally:
    \[ \exists x_j \in S_j(x), (x_j, x_j = \ell_j) \wedge (x_j \in R_d(x)) \wedge (\forall x_j \in S_j(x), x_j, x_j \neq \ell_j) \]
  - The maximum value of \( \text{Freq}(x_j, S_j(x)) \) \( = \frac{S_j(x) + 1}{2} \) is reached when all context nodes have the same label \( \ell_j \) and appear on the inner-most ring \( R_d(x) \) of the sphere neighborhood, more formally:
    \[ \forall x_j \in S_j(x), (x_j, x_j = \ell_j) \wedge (x_j \in R_d(x)) \]

Consequently:

- The minimum value \( w_{V_j(x)}(x_j) = 0 \) is obtained when no nodes of label \( \ell_j \) occur in the sphere context of target node \( x \).
- The maximum value \( w_{V_j(x)}(x_j) = 1 \) is obtained when maximum frequency is obtained, since the weight score is normalized using maximum frequency, i.e., \( \frac{S_j(x) + 1}{2} \).

In short, context nodes are weighted based on their labels’ occurrences as well as the sizes (radiiues) 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 \(x\), \(s\) in a semantic network \(SN\), we define \(\text{Concept\_Score}(s_p, S_d(x), SN)\) to quantify the semantic impact of \(s_p\) as the potential candidate for the intended sense (meaning) of \(x\) 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:

\[
\text{Concept\_Score}(s_p, S_d(x), SN) = \sum_{s_i \in S_d(x)} \frac{\max_{s_j \in S_d(x)} (\text{Sim}(s_p, s_j, SN)) \times w_{ij}(s_i, s_j)}{|S_d(x)|} \in [0, 1]
\]  

(12)

where \(s'_j\) designates the \(j\)th sense of context node \(x, s \in S_d(x)\), and \(\text{Sim}(s_p, s'_j, SN)\) designates the semantic similarity measure between senses \(s_p\) and \(s'_j\) 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:

\[
\text{Sim}(c_1, c_2, SN) = w_{\text{edge}} \times \text{Sim}_{\text{edge}}(c_1, c_2, SN) + w_{\text{node}} \times \text{Sim}_{\text{node}}(c_1, c_2, SN) + w_{\text{gloss}} \times \text{Sim}_{\text{gloss}}(c_1, c_2, SN) \in [0, 1]
\]  

(13)

where:

– \(w_{\text{edge}} + w_{\text{node}} + w_{\text{gloss}} = 1\) and \((w_{\text{edge}}, w_{\text{node}}, w_{\text{gloss}}) \geq 0\)

– \(\text{Sim}_{\text{edge}}\) is a typical edge-based measure from [114], cf. formula (1)

– \(\text{Sim}_{\text{node}}\) is a typical node-based measure from [50], cf. formula (2)

– \(\text{Sim}_{\text{gloss}}\) 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 \(\text{gloss}(c_1)\) and the glosses if its surrounding concepts in \(SN\) connected using different semantic relations \(\text{gloss}(\text{rel}(c_1))\)) the concept’s set of synonymous words/expressions \(\text{syns}(c_1)\) within its extended gloss, thus enriching the latter with more semantically related terms. Formally:

\[
\text{Sim}_{\text{gloss}}(c_1, c_2, SN) = \text{ExtGloss}(c_1) \cap \text{ExtGloss}(c_2) \\
= \{\text{gloss}(c_1) \cup \text{gloss}(\text{rel}(c_1)) \cup \text{syns}(c_1)\} \cap \{\text{gloss}(c_2) \cup \text{gloss}(\text{rel}(c_2)) \cup \text{syns}(c_2)\} \in [0, 1]
\]  

(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:

\[
\text{ExtGloss}(c_i) \cap \text{ExtGloss}(c_j) = \frac{\sum_{n=1}^{m} n^2}{\max[[\text{ExtGloss}(c_i)], [\text{ExtGloss}(c_j)]]} \in [0, 1]
\]  

(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 \(n\) \(\in [1, \max([\text{ExtGloss}(c_i)], [\text{ExtGloss}(c_j)])]\)

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), \(\text{Sim}_{\text{gloss}}\) varies as follows:

– The minimum value \(\text{Sim}_{\text{gloss}} = 0\) 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 \(\text{Sim}_{\text{gloss}} = 1\) 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 \(n = [\text{ExtGloss}(c_i)] = [\text{ExtGloss}(c_j)]\).

– The value of \(\text{Sim}_{\text{gloss}}\) increases with the increase of the number \((m)\) and size \((n)\) of \(n\)-word overlap sequences, and decreases otherwise.

Algorithm XSD\(_{\text{Concept}}\) 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 \(x\)'s sphere neighborhood \(S_d(x)\) and context vector \(\vec{V}_d(x)\) (line 1), each of the target label senses \(s_p\) in \(SN\) is compared with the senses \(s'_j\) of each of the context node labels \(x, \ell\) (lines 3-9) weighted using corresponding sphere neighborhood vector weights \(w_{\text{edge}}(x, \ell)\) (line 10). The target node sense with the maximum combined similarity score is

\(^{10}\) Semantic relatedness is more general than similarity; dissimilar concepts 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 \(n^2 + n^2 + \ldots + n^2\) times.
then 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 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 $x, \ell$

Begin  
1. Generate $S_d(x) \in T$ and $V_d(x)$ // 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. For each $x_i \in S_d(x)$ // Processing senses of context nodes  
      4. For each $s_i \in SN \rightarrow x, \ell$ \{$\text{Sim}_i(s_i, s_p) = \text{Sim}(s_i, s_p)$\}  
      5. $\text{Max}_i = \max_{s_i \in S_d(x)} \text{Sim}_i(s_i, s_p)$  
      6. $\text{Context}_i = \frac{\max_{s_i \in S_d(x)} \text{Sim}_i(s_i, s_p)}{|S_d(x)|}$  
   7. $c = s_i / \text{Context}_i(s_i)$  
Return $c$ // Sense (concept) with maximum score  
End  

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

**Special case:** If the target node label $x, \ell$ 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 ($s_{p_1}, s_{p_2}$) corresponding to each of the individual token senses ($s_{p_1}$ for token $t_1$, and $s_{p_2}$ for $t_2$) is computed to identify the sense combination which is most suitable for the compound node label:

$$\text{Concept}_i(s_{p_1}, s_{p_2}, S_d(x), SN) = \frac{\sum_{s_i \in S_d(x)} \text{Sim}_i(s_{p_1}, s_{p_2}, s_i, SN) \times w_{\text{Sim}_i}(s_i, s_p)}{|S_d(x)|} \in [0, 1]$$

where

$$\text{Sim}_i(s_{p_1}, s_{p_2}, s_i, SN) = \frac{\text{Sim}(s_{p_1}, s_i, SN) + \text{Sim}(s_{p_2}, s_i, SN)}{2} \in [0, 1]$$

Note that a compound context node label $x, \ell$ which tokens $t_1$ and $t_2$ 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 $R_{\text{d'}w}(c)$, $S_d(c)$, and $V_d(c)$ 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 $V_d(x)$, and given $s_p$ as one possible sense for $x, \ell$ in a reference semantic network $SN$, with its sphere neighborhood $S_d'(s_p) \in SN$ and context vector $V_d'(s_p)$, we define $\text{Context}_i(s_p, S_d(x), d', SN)$ to quantify the semantic impact of $s_p$ as the potential candidate designating the intended sense (meaning) of $x, \ell$ within context $S_d(x)$ in $T$ w.r.t. $SN$, computed using a vector similarity measure between $V_d(x)$ and $V_d'(s_p)$:

$$\text{Context}_i(s_p, S_d(x), d', SN) = \cos(V_d(x), V_d'(s_p)) \in [0, 1]$$

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 radiiuses $d$ and $d'$  
$T$ // Source XML tree  
$SN$ // Reference semantic network  
**Output:** $c$ // Semantic concept representing the sense (meaning) of $x, \ell$

Begin  
1. Generate $S_d(x) \in T$ and $V_d(x)$ // 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. Generate $S_d(s_p) \in SN$ and $V_d'(s_p)$ // Context vector of $s_p$ in $SN$  
   4. $\text{Context}_i = \frac{\cos(V_d(x), V_d'(s_p))}{||V_d(x)|| \times ||V_d'(s_p)||}$ // Context similarity  
   5. $c = s_i / \text{Context}_i(s_i)$  
Return $c$ // Sense (concept) with maximum score  
End  

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

XSD\_Context accepts as input a target XML node $x$ and user specified sphere neighborhood radiiuses $d$ 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 $V_d'(x)$ 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 $\hat{V}_d(s_p)$ for each of the target label senses $s_p$ in SN (lines 3-5). It then compares the target node context vector $\hat{V}_d(x)$ with each of the label sense context vectors $\hat{V}_d(s_p)$ 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 $x$. 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:

$$\text{Concept Score}(s_p, s_q, S_d(x), SN) = \cos(\hat{V}_d(x), \hat{V}_d(s_p, s_q)) \in [0, 1]$$

where $\hat{V}_d(s_p, s_q)$ is a compound context vector generated based on compound sphere neighborhood $S_d(s_p, s_q) = S_d(s_p) \cup S_d(s_q)$.

### 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:

$$\text{Combined Score}(s_p, S_d(x), SN) = w_{\text{Concept}} \times \text{Concept Score}(s_p, S_d(x), SN) + w_{\text{Context}} \times \text{Context Score}(s_p, S_d(x), SN) \in [0, 1]$$

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

Algorithm $\text{XSD}_{\text{Combined}}$ is an integration of algorithms $\text{XSD}_{\text{Concept}}$ and $\text{XSD}_{\text{Context}}$ and is thus omitted here for clarity.

### 3.6. Complexity Analysis

The overall time complexity of our XML disambiguation approach following XSD simplifies to:

$$O(|T| \times |x| \times |\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|)$$

where $|T|$ designates the cardinality (number of nodes) of the XML tree, $|x|$ the total number of XML nodes selected for disambiguation, $x$ a target node, $|\text{senses}(x.|\ell|)|$ 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, $x \in S_d(x)$ a context node, $|\text{senses}(x.|\ell|)|$ the total number of possible senses of context node label $x.|\ell|$, $s_p \in \text{senses}(x.|\ell|$ 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 XSD, and mainly amounts to the complexities of our concept-based and context-based disambiguation algorithms:

- The complexity of the three initial phases of XSD: 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: $O(|T| \times n \times ||l|) + O(|T|) + O(|I|)$, which simplifies to $O(|I|)$.

- The complexity of our XML semantic disambiguation phase simplifies to:

$$O(|x| \times |\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|)$$

- The complexity of algorithm $\text{XSD}_{\text{Concept}}$ (cf. Fig. 9) amounts to $O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|) \times O(|\text{SN}| \times \text{depth}(\text{SN}))$, 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 without a single match in the SN), ii) $O(|\text{SN}| \times \text{depth}(\text{SN}))$ underlines the complexity of the edge-based and node-based semantic similarity measures adopted in our study; and iii) $O(|\text{gloss}|^2)$ is the time complexity of our gloss-based measure where gloss designates our extended gloss (including synonyms). Yet, given that $\text{SN}$ designates a fixed reference throughout the disambiguation process (i.e., $|\text{SN}|$, depth(SN), and $|\text{gloss}|$ remain unchanged), and given that the number of tokens per XML node label is usually $n=1$, $\text{XSD}_{\text{Concept}}$‘s complexity simplifies to $O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|)$.

- The complexity of algorithm $\text{XSD}_{\text{Context}}$ (Fig. 10) amounts to $O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|) \times O(|\text{SN}| \times \text{depth}(\text{SN}))$, where: if $O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|)$ designates the complexity of computing the cosine measure between context vectors $\hat{V}_d(x)$ and $\hat{V}_d(s_p)$, and if $n \times |S_d(x)|$ designates the size of vector $\hat{V}_d(s_p)$ 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 $n=1$, $\text{XSD}_{\text{Context}}$‘s complexity simplifies to $O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|)$.

- The complexity of algorithm $\text{XSD}_{\text{Combined}}$ consists of the sum of the complexities of $\text{XSD}_{\text{Concept}}$ and $\text{XSD}_{\text{Context}}$ and thus simplifies to: $O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|) + O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|) = O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|)\times|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|$$

When applied to all target nodes $|x|$, the complexity of our XML semantic disambiguation phase becomes:

$\sum_{x \in X} O(|\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|) = O(|x| \times |\text{senses}(x.|\ell|)\times|S_d(x)|+|S_d(x)|)$$

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 \( O\left(\sum_{x \in X} \left(\sum_{x \in X} + \sum_{x \in X} + \sum_{x \in X}\right)\right) \) 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, or 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, \( Amb_{Deg} \in [0, 1] \). 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:

\[
Struct_{Deg}(x, T) = \frac{w_{Depth} \times x.f.d}{Max(depth(T))} + \frac{w_{Fan-out} \times x.f.f}{Max(fan-out(T))} + \frac{w_{Density} \times x.f.d}{Max(fan-out(T))} \in [0, 1]
\]

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

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 \(w_{Depth} = w_{Fan-out} = w_{Density} = 1/3\) 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.

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, \( pcc = \delta_{XY}/(\sigma_X \cdot \sigma_Y) \) where: \( x \) and \( y \) designate node and system generated ambiguity degree ratings respectively, \( \sigma_x \) and \( \sigma_y \) denote the standard deviations of \( x \) and \( y \) respectively, and \( \delta_{XY} \) 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 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 \( \in [0, 4] \), i.e., \( \in [\min, \max] \) ambiguity) where acquired from each tester for 12-to-13 randomly pre-selected nodes\(^{15}\), 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

\(^{15}\) 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.

\(^{16}\) 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" 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_Depolysem Amb_Depthes, and Amb_Depth): i) Test #1 considers all three factors equally (w_Depolysem = W_Depthes = W_Depth = 0); ii) Test #2 focuses on the polysemous factor (w_Depolysem = 1 while W_Depthes = W_Depth = 0); iii) Test #3 focuses on the depth factor (w_Depthes = 1 while w_DePolysem = 0.2 and w_Depth = 0); iv) Test #4 focuses on the density factor (w_Density = 1, w_Depolysem = 0.2 and w_Depth = 0).

Table 2. Correlation between human tester ratings.

<table>
<thead>
<tr>
<th>Testers</th>
<th>1-2</th>
<th>3-4</th>
<th>5-6</th>
<th>7-9</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1: Dataset 1</td>
<td>-0.254</td>
<td>0.611</td>
<td>-0.205</td>
<td>-0.283</td>
<td>-0.184</td>
</tr>
<tr>
<td>Group 1: Dataset 2</td>
<td>-0.026</td>
<td>0.500</td>
<td>-0.241</td>
<td>0.485</td>
<td>-0.160</td>
</tr>
<tr>
<td>Group 2: Dataset 3</td>
<td>0.061</td>
<td>0.522</td>
<td>0.284</td>
<td>0.284</td>
<td>0.194</td>
</tr>
<tr>
<td>Group 2: Dataset 4</td>
<td>0.128</td>
<td>0.127</td>
<td>0.269</td>
<td>0.269</td>
<td>0.223</td>
</tr>
<tr>
<td>Group 3: Dataset 5</td>
<td>0.527</td>
<td>0.818</td>
<td>0.818</td>
<td>0.365</td>
<td>0.365</td>
</tr>
<tr>
<td>Group 4: Dataset 6</td>
<td>0.552</td>
<td>1</td>
<td>1</td>
<td>0.114</td>
<td>0.553</td>
</tr>
<tr>
<td>Group 4: Dataset 7</td>
<td>0.944</td>
<td>1</td>
<td>1</td>
<td>0.944</td>
<td>0.944</td>
</tr>
<tr>
<td>Group 4: Dataset 8</td>
<td>0.219</td>
<td>0.277</td>
<td>0.178</td>
<td>0.026</td>
<td>0.436</td>
</tr>
<tr>
<td>Group 4: Dataset 9</td>
<td>0.224</td>
<td>-0.068</td>
<td>0.081</td>
<td>-0.08</td>
<td>0.219</td>
</tr>
<tr>
<td>Group 4: Dataset 10</td>
<td>0.401</td>
<td>1</td>
<td>1</td>
<td>-0.123</td>
<td>0.461</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Test</th>
<th>Group 1</th>
<th>Dataset 1</th>
<th>0.394</th>
<th>0.411</th>
<th>0.335</th>
<th>0.439</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Group 2</td>
<td>Dataset 2</td>
<td>0.017</td>
<td>0.184</td>
<td>0.243</td>
<td>0.139</td>
</tr>
<tr>
<td>Test</td>
<td>Group 3</td>
<td>Dataset 3</td>
<td>-0.087</td>
<td>-0.139</td>
<td>-0.071</td>
<td>-0.138</td>
</tr>
<tr>
<td>Test</td>
<td>Group 4</td>
<td>Dataset 4</td>
<td>0.408</td>
<td>0.438</td>
<td>0.390</td>
<td>0.398</td>
</tr>
<tr>
<td>Test</td>
<td>Group 4</td>
<td>Dataset 5</td>
<td>-0.184</td>
<td>-0.185</td>
<td>-0.131</td>
<td>-0.235</td>
</tr>
<tr>
<td>Test</td>
<td>Group 4</td>
<td>Dataset 6</td>
<td>-0.284</td>
<td>-0.291</td>
<td>-0.243</td>
<td>-0.316</td>
</tr>
</tbody>
</table>

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

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).

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.
Table 4. Characteristics of test documents.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Datasets</th>
<th>Source dataset</th>
<th>Grammar</th>
<th>Nr of docs</th>
<th>Avg. Nr of nodes per doc</th>
<th>Node label polysemy (Nr of senses)</th>
<th>Node Depth</th>
<th>Node Fan-out</th>
<th>Node Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>1</td>
<td>Shakespeare collection</td>
<td>shakespeare.dtd</td>
<td>10</td>
<td>192,054</td>
<td>7.052</td>
<td>30</td>
<td>3.687</td>
<td>6</td>
</tr>
<tr>
<td>Group 2</td>
<td>2</td>
<td>Amazon product files</td>
<td>amazon_product.dtd</td>
<td>113,333</td>
<td>8.407</td>
<td>72</td>
<td>4.309</td>
<td>7</td>
<td>0.539</td>
</tr>
<tr>
<td>Group 3</td>
<td>3</td>
<td>SIGMOD Record</td>
<td>ProceedingsPage.dtd</td>
<td>6</td>
<td>39,375</td>
<td>4,815</td>
<td>16</td>
<td>2.741</td>
<td>8</td>
</tr>
<tr>
<td>Group 4</td>
<td>4</td>
<td>IMDb database</td>
<td>movies.dtd</td>
<td>6</td>
<td>15,475</td>
<td>4,10</td>
<td>10</td>
<td>2.666</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Niagara collection</td>
<td>bib.dtd</td>
<td>8</td>
<td>26.5</td>
<td>4.384</td>
<td>13</td>
<td>2.961</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>W3Schools</td>
<td>cd_catalog.dtd</td>
<td>4</td>
<td>16.5</td>
<td>3.937</td>
<td>10</td>
<td>2.312</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>W3Schools</td>
<td>food_menu.dtd</td>
<td>4</td>
<td>16.0</td>
<td>2.375</td>
<td>7</td>
<td>2.437</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>W3Schools</td>
<td>plant_catalog.dtd</td>
<td>4</td>
<td>11.675</td>
<td>3.454</td>
<td>15</td>
<td>2</td>
<td>1.181</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Niagara collection</td>
<td>personnel.dtd</td>
<td>4</td>
<td>19</td>
<td>3.947</td>
<td>9</td>
<td>2.368</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Niagara collection</td>
<td>club.dtd</td>
<td>4</td>
<td>15.5</td>
<td>4.533</td>
<td>10</td>
<td>2.266</td>
<td>4</td>
</tr>
</tbody>
</table>

3) Concerning Amb_Deg’s parameter weight variations (for \(W_{Polysy}m\), \(W_{depth}\), 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.

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 not 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.

21 Available at [http://metalab.unc.edu/bosak/xml/eg/shaks200.zip](http://metalab.unc.edu/bosak/xml/eg/shaks200.zip)
22 Available at [http://simply-amazon.com/content/XML.html](http://simply-amazon.com/content/XML.html)
23 Available at [http://www.acm.org/sigmod/xml](http://www.acm.org/sigmod/xml)
26 Available from [http://www.w3schools.com](http://www.w3schools.com)
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\textsubscript{Concept}) generally produces higher f-value levels in comparison with the context-based approach (XSD\textsubscript{Context}), the latter appearing to be more sensitive to context size (especially with Groups 2, 3, and 4). This is expected since algorithm XSD\textsubscript{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\textsubscript{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\textsubscript{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\textsubscript{Concept} and XSD\textsubscript{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]<sup>27</sup>. 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 approach).
approach\textsuperscript{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.

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\textsuperscript{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.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>$f$-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSDF</td>
<td>0.5447</td>
<td>0.5226</td>
<td>0.5312</td>
</tr>
<tr>
<td>RPD [95]</td>
<td>0.3946</td>
<td>0.4737</td>
<td>0.4828</td>
</tr>
<tr>
<td>VSD [57]</td>
<td>0.4728</td>
<td>0.4598</td>
<td>0.4659</td>
</tr>
</tbody>
</table>

\textsuperscript{28} Manually identified from repeated tests with different parameter values.

\textsuperscript{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\textsubscript{Concept}, cf. Fig. 9).](image)

![Fig. 18. Timing results regarding our context-based approach (when executing algorithm XSD\textsubscript{Context}, cf. Fig. 10).](image)
Results highlight the linear complexities of both our concept-based approach (using algorithm $XSD_{Concept}$ i.e., $O(\|senses(x, \ell)\| \times |S_d(x)| \times |senses(x, \ell)|)$ (Fig. 17) and our context-based approach (using algorithm $XSD_{Context}$), i.e., $O(\|senses(x, \ell)\| \times (|S_d(x)| + |S_d(\ell)|))$ (Fig. 18). Time is also linear w.r.t. the number of target nodes being disambiguated, designated as $[X]$.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Considers linguistic pre-processing</th>
<th>Considers tag tokenization (compound terms)</th>
<th>Addresses XML node ambiguity</th>
<th>Integrates an inclusive XML structure context</th>
<th>Flexible w.r.t. context size</th>
<th>Adopts relational information approach</th>
<th>Combines the results of various semantic similarity measures</th>
<th>Straightforward mathematical functions</th>
<th>Disambiguates XML structure and content</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPD [95]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>VSD [57]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>XSDF (our approach)</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

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.

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.

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: ii) Information Retrieval, iii) Web and Mobile Services, and iv) 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), ii) the type layer (i.e., document/grammar comparison), and iii) 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://sigappfpr.acm.org/Projects/XSD/]
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 warehousing32, 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/OWL34) 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 text 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 $<\text{subject}, \text{predicate}, \text{object}>$ 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 from 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 [33].

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


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, AIAl, 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.