Computer Science and Information Systems 20(1):423–457 https://doi.org/10.2298/CSIS220228063T

Combining Offline and On-the-fly Disambiguation to Perform Semantic-aware XML Querying

Joe Tekli[1], Gilbert Tekli[2], and Richard Chbeir[3]

[1] School of Engineering, ECE dept., Lebanese American University,
36 Byblos, Lebanon
joe.tekli@lau.edu.lb
[2] Faculty of Technology, Mechatronics dept., University of Balamand,
100 Tripoli, Lebanon
gilbert.Tekli@balamand.edu.lb
[3] LIUPPA Lab., IUT de Bayonne, University of Pau and Pays Adour
64000 Anglet, France
richard.chbeir@univ-pau.fr

Abstract. Many efforts have been deployed by the IR community to extend free-text query processing toward semi-structured XML search. Most methods rely on the concept of Lowest Comment Ancestor (LCA) between two or multiple structural nodes to identify the most specific XML elements containing query keywords posted by the user. Yet, few of the existing approaches consider XML semantics, and the methods that process semantics generally rely on computationally expensive word sense disambiguation (WSD) techniques, or apply semantic analysis in one stage only: performing *query relaxation/refinement* over the *bag of words* retrieval model, to reduce processing time. In this paper, we describe a new approach for XML keyword search aiming to solve the limitations mentioned above. Our solution first transforms the XML document collection (offline) and the keyword query (on-the-fly) into meaningful semantic representations using context-based and global disambiguation methods, specially designed to allow almost linear computation efficiency. We use a semantic-aware inverted index to allow semantic-aware search, result selection, and result ranking functionality. The semantically augmented XML data tree is processed for structural node clustering, based on semantic query concepts (i.e., key-concepts), in order to identify and rank candidate answer sub-trees containing related occurrences of query key-concepts. Dedicated weighting functions and various search algorithms have been developed for that purpose and will be presented here. Experimental results highlight the quality and potential of our approach.

Keywords: Semi-structured data, XML, Semantic Analysis, Semantic Disambiguation, Keyword Search, Query Processing.

1. Introduction

Various methods have been proposed for XML ranked retrieval. While most approaches consider content-and-structure features in specifying XML query constraints, few approaches have targeted semantic XML search based on simple keyword queries. Most approaches in this category exploit the concept of LCA (Lowest Common Ancestor)

between two or multiple structural nodes to identify the most specific XML elements containing query keywords posted by the user. Yet LCA-based methods underline various limitations: i) each result candidate must contain all query keywords, which is not always intuitive since a candidate result (element or sub-tree) containing most (and not necessarily all) keywords might be deemed relevant by the user; ii) some meaningful results might be missed: as XML trees underline different nesting hierarchies, restricting results to the LCA encompassing all keywords might miss some more general, and yet relevant results; iii) few of the proposed approaches consider semantics: for instance, when submitting sample keyword query "Universities in Sao Paulo", the user is probably interested in information concerning universities, academies and colleges in Sao Paulo, and cities in its vicinity such as Campinas, Sao Carlos, etc. Hence, semantic analysis becomes essential in such a context in order to improve search results; iv) the few existing methods that do target XML semantics generally rely on word sense disambiguation (WSD) and are computationally expensive, or v) apply semantic analysis in one stage only, performing query relaxation/refinement over the bag of words retrieval model, to reduce processing time.

In this paper, we introduce *XSemSearch*, a semantic-aware XML keyword search solution aiming to solve the limitations mentioned above. We propose to integrate semantic analysis and structural clustering in formulating an efficient solution to the problem. Our solution first transforms the XML document collection (offline) and the keyword query (on-the-fly) into meaningful semantic representations using context-based and global disambiguation methods, specially designed to allow almost linear computation efficiency. The semantically augmented XML data tree is processed for structural node clustering, based on semantic query concepts (i.e., key-concepts), in order to identify and rank candidate answer sub-trees containing related occurrences of query key-concepts. The overall architecture of our approach is depicted in Fig. 1

An initial description of *XSemSearch*'s architecture is given in [84]. This paper adds: i) a dedicated inverted index structure to handle semantically augmented XML data, ii) an dedicated query formalism to allow structure-and-content queries with only partial knowledge of the data collection structure and semantics, iii) two alternative query processing algorithms including *Query As You Type Search* and *Parallel Semantic Search*, and iv) an extended empirical study to evaluate query processing time and quality.

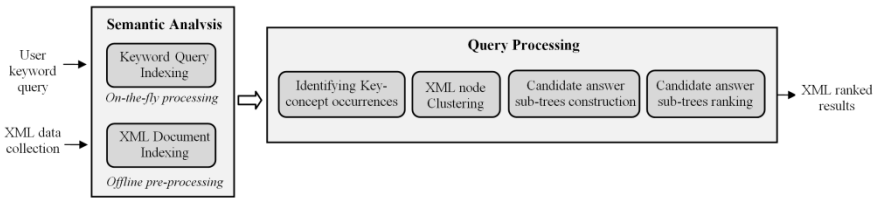


Fig. 1. Overall architecture of our XML semantic-aware search approach

The remainder of the paper is organized as follows. Section 2 reviews the background in XML and query semantic analysis. Section 3 provides an overview of our solution framework. Sections 4 and 5 respectively describe the XML semantic analysis and keyword query semantic analysis components. Section 6 describes the

query processing component. Section 7 provides experimental results, before concluding in Section 8.

2. Background

In this section, we review the background in semantic information retrieval, while focusing on XML and keyword query semantic analysis and disambiguation.

2.1. Semantic Information Retrieval

The retrieval model for an information retrieval system specifies how documents and queries are represented, and how these representations are compared to produce relevant result estimates [7]. A core problem in this context is lexical ambiguity: a word may have multiple meanings (homonymy), a word maybe implied by other related words (metonymy), and/or several words can have the same meaning (synonymy) [42].

The lexical ambiguity problem becomes even more acute on the Web, with the latter's heterogeneous and unstructured nature which makes it even more difficult to query and retrieve meaningful information. Semantic IR is part of the Semantic Web vision [76] that promises to solve the retrieval ambiguity problem, by i) associating terms in Web pages and queries with explicit semantics (i.e., word senses or concepts), and then ii) performing search functions based on document/query concepts rather than plain terms [55]. A core challenge in this context is word sense disambiguation (WSD): how to resolve the semantic ambiguities and identify the intended meanings of document terms and query keywords [11]. Various methods have been proposed for WSD in the literature [42, 53, 75]. They fall in two main categories: *corpus-based* WSD and *knowledge-based* WSD. The corpus-based approach is data-driven, as it involves information about words previously disambiguated and requires supervised learning from sense-tagged corpora to enable predictions for new words. Knowledge-based methods are knowledge-driven, as they handle a sense inventory and/or a repository of information about words that can be exploited to distinguish their meanings in the text. Machine-readable knowledge bases (e.g., dictionaries or semantic networks: thesauri, taxonomies, or ontologies) 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 [6, 38], 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 or feasible in practice. Therefore, knowledge-based methods have been receiving more attention [16, 42]. In the remainder of our study, we focus on knowledge-based WSD and semantic analysis.

2.2. XML Semantic Analysis and Disambiguation

While a considerable amount of research has been undertaken around (knowledge-based) WSD in flat textual data [53], yet few approaches have been developed in the

context of XML and semi-structured information [75]. The main difference resides in the notion of XML (structural) contextualization. The context of a keyword, in traditional textual data, consists of the set of terms in the keyword's vicinity (i.e., terms occurring to the left and right of the considered keyword, within a certain predefined distance from the keyword [11]). However, there is no clear definition regarding the context of a node in an XML tree. The authors in [72, 73] 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, using context-driven search techniques for determining the relationships between the different unified entities, so as to identify related semantic labels.

In [70, 71], the authors extend the notion of XML node context to include the whole XML root path, i.e., path consisting of the sequence of nodes connecting a given node with the root of the XML document (or document collection). They consequently perform per-path sense disambiguation, comparing every node label in each path with all possible senses of node labels occurring in the same path (using a gloss-based WordNet similarity measure [8]) in order to select the most appropriate sense for the label at hand. Different from the notions of parent context and path context, the authors in [85] consider the set of XML tag names contained in the sub-tree rooted at a given element node, i.e., the set of labels corresponding to the node at hand and all its subordinates, to describe the node's XML context. The authors apply a similar paradigm to identify to contexts of all possible node label senses in WordNet. Consequently, they perform label sense disambiguation by comparing the XML label context to all candidate sense contexts in WordNet, identifying the sense (semantic concept) with the highest similarity.

In [49], the authors combine the notions of parent context and descendent (sub-tree) context in disambiguating generic structured data (e.g., XML, web directories, and ontologies). The authors consider that a node's context definition depends on the nature of the data and the application domain at hand. They propose various edge-weighting heuristics (namely a Gaussian decay function) to identify *crossable* edges, i.e., nodes reachable from a given node through any *crossable* edge belong to the target node's context. Consequently, 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 (using an edge-based semantic similarity measure [43], following the hypernymy/hyponymy relations in WordNet) in order to identify the highest matching semantic concept.

Another concern in XML-based WSD is how to effectively process the context of an XML node taking into account the structural dispositions of XML data. In fact, most existing WSD methods developed for flat textual data [42, 53], and those developed for XML-based data [70-73], follow the bag-of-words paradigm where the context is processed as a plain set of words surrounding the term/label (node) to disambiguate. In other words, all context nodes are treated the same, despite their structural positions in the XML tree. We encountered an approach in [49] which extends the traditional bag-or-words paradigm with additional information considering distance weights separating the context and target nodes (identified as *relational information model* [49]). The authors employ a heuristic Gaussian distance decay function estimating edge weights such that the closer a node (following a user-specified direction, e.g., ancestor, descendent, or both), the more it influences the target node's disambiguation [49]. The

semantic contribution of each context node is weighted by its position in the context graph of the target node.

2.3. Query Semantic Analysis and Disambiguation

Semantic query analysis in information retrieval usually involves two steps: i) WSD to identify the user's intended meaning of query terms, and ii) semantic query representation/expansion in order to alter the query so that it achieves better (precision and recall) results [67]. As described in the previous section, traditional semantic analysis and disambiguation techniques usually rely on the notion of context such that terms (e.g., node labels in the context of XML) that appear together in the same context have related meanings [11]. While context-based solutions are applicable with classic IR queries which are rather lengthy (e.g., 15 terms on average for short queries [91], reaching up to 50-85 terms for long queries [12]), nonetheless, keyword queries on the Web are usually 2-3 words long [15] which is generally insufficient in identifying a meaningful context [42, 49]. In fact, lexical ambiguity with Web search is often the consequence of the low number of query words entered on average by Web users [40]. Therefore, some sort of user interaction is usually required to counter the lack of contextualization, and more accurately identify the intended senses of Web query terms [24, 89].

Various methods for interactive keyword querying have been proposed in the literature, e.g., [41, 66] [30, 74]. Most existing approaches are *corpus-based* in that they expand user queries by adding words that co-occur with the query terms in a given corpora, i.e. words that, on a probabilistic ground, are believed to describe the same *semantic concept* (e.g. *car* and *driver*). Here, expansion terms are usually identified from i) user feedback: extracting frequent terms occurring in previous results deemed relevant by the user [41, 66], and/or ii) query logs: identifying frequent terms in the document collection based on the associations between past queries and the documents downloaded by the user [30, 74]. Yet, the extensive training and huge corpora requirements of *corpus-based* methods makes them less practical in the context of Web search applications, which has led to a growing interest in *knowledge-based* solutions [35, 61]. The latter family of methods investigates the use of ontological information to assist the user in formulating and/or expanding keyword queries by: i) allowing user interaction to identify the intended senses of query-terms, and then ii) expanding/modifying query keywords via their most related semantic concepts in the reference semantic source (e.g., WordNet) [67].

Following [14], a keyword query is first processed for lexical normalization, and then presented to the user as a set of lexical tokens, where each token is associated with a set of possible semantic meanings (identified using WordNet and/or domain specific ontologies). Consequently, the user is asked to select the most relevant sense for each lexical token. The system then exploits the selected user senses to reformulate the query using dedicated heuristics (e.g., replacing actual keywords via their synonyms with highest frequency of usage in WordNet, identifying negative keywords, i.e., the terms corresponding to the highest frequency synset remaining beside the one selected by the user, etc.), thus obtaining a semantically augmented keyword query. A similar approach is adopted in [45] with a special emphasis on failed-query reformulation. The authors in [45] assume that the reformulation of a failed query without help from the system can

be frustrating to the user, and thus suggest to assist the later by proposing semantically meaningful keywords selected from WordNet (using heuristics similar to those adopted in [14]). The method in [45] is developed in the context of the NALIX project for building an interactive natural language interface for querying XML [44].

A fully automated approach to *knowledge-based* query disambiguation is introduced in [54], where the authors exploit structural pattern recognition [25] in mapping query keyword senses. The proposed method creates a local semantic network for each keyword-sense in the query, including most semantic relations utilized in WordNet [28] (hypernymy, hyponymy, meronymy, etc.). Then, for each possible configuration of senses, the system identifies the intersections between corresponding pair-wise local semantic networks using an adapted structure pattern recognition algorithm. Common nodes are those that can be reached through both semantic networks being compared. The configuration with the highest intersection score (i.e., highest number of intersecting nodes) is selected as the one encompassing the most relevant keyword senses. In a subsequent step, the authors propose various heuristics to expand the query using synset, hyponymy and/or gloss information. Experimental results in [54] show a 26.85% improvement in retrieval precision over the plain query words.

Note that most existing studies targeting *knowledge-based* query semantic analysis, e.g., [35, 61] [14, 45, 56], do not evaluate the complexity (or execution time) levels of their proposed methods. Nonetheless, time complexity is critical for on-the-fly execution on the Web (in comparison with document semantic analysis which could be performed offline). The time complexity of query semantic analysis might even prove to be problematic in the case of the pattern recognition-based methods [19, 54], since traditional structure pattern recognition problems are usually of exponential complexity [25, 58].

3. Proposal Overview

Semantic similarity evaluation between two terms usually consists in looking up each term's lexical concept in a reference knowledge base (e.g., a semantic network such as WordNet), and consequently comparing the underlying concepts. Nonetheless, semantic similarity evaluation has been proven to be an expensive task: comparing two semantic concepts following one of the most prominent semantic similarity measures in the literature, i.e., [47], requires $O(|SN| \times Depth(SN))$ time where $|SN|$ is the size (i.e., cardinality in number of concepts) of the reference semantic network SN, and $Depth(SN)$ its maximum depth. Evaluating the semantic similarity between query keywords and each label/term in the XML document collection becomes extremely complex, and practically unfeasible.

A way of getting round the complexity problem would be to perform semantic analysis of the XML document collection, offline, and prior to the retrieval phase. This consists in transforming the XML documents into weighted semantic trees (graphs), and transforming and expanding the keyword query into a set of weighed semantic concepts. Consequently, an adapted XML IR engine (cf. Section 6) processes the semantically indexed documents and queries, so as produce more meaningful results. Our semantic analysis processes are depicted in Fig. 2.

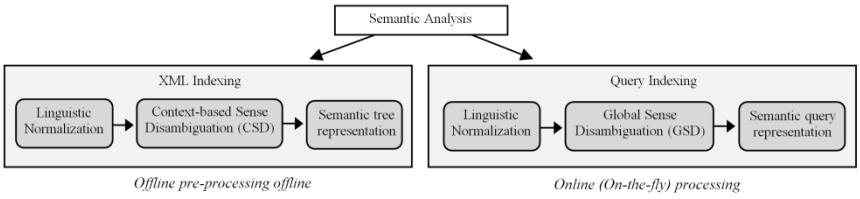


Fig. 2. Semantic analysis of XML document and keyword query

Note that while semantic query indexing is performed online, XML document indexing is performed offline, and does not affect the complexity of the approach. As shown in Fig. 2, semantic indexing consists of three main phases: i) Linguistic Normalization, ii) Sense Disambiguation, and iii) Semantic Representation. While the first phase (Linguistic Normalization, including *tokenization*, *expansion*, *stop word removal*, and *stemming*) is similar for both document labels and query keywords, yet, we design the latter two (sense disambiguation, and semantic representation) differently following the data models and requirements at hand. Sense disambiguation usually relies on the notion of context, where terms that appear together in the same context have related meanings [11]. While context information is available for XML document nodes (e.g., the context of a node could be its parent node, its root path, the whole document tree containing the node, etc.), yet, keyword queries on the Web are usually two-to-three words long [15] which is generally insufficient in identifying a meaningful context [42, 49]. Hence, we introduce two different methods for document and query sense disambiguation: i) Context-based Sense Disambiguation (CSD) for XML documents, ii) Global Sense Disambiguation (GSD) for keyword queries.

In the following, Sections 4 and 5 present the XML document semantic analysis and the keyword query semantic analysis processes respectively

4. Semantic XML Document Analysis

Our XML document semantic analysis process consists in: i) disambiguating each label following its context, to associate each label with the proper semantic concept in the reference knowledge base (e.g., WordNet), and ii) producing a semantically indexed XML tree, with the corresponding index structures and pair-wise concept semantic weights, to be consequently utilized in the query processing task. We describe the latter in the following sub-sections.

4.1. XML Data Model

XML documents represent hierarchically structured information and are generally modeled as ordered labeled trees (cf. Fig. 3). In a traditional DOM (Document Object Model) ordered labeled tree [87], nodes represent XML elements, and are labeled with corresponding element tag names, ordered following their order of appearance in the document. Attributes usually appear as children of their encompassing element nodes, sorted by attribute name, and appearing before all sub-element siblings [57, 93]. Other

types of nodes, such as entities, comments and notations, are commonly disregarded in most XML comparison approaches, e.g., [22, 36], since they underline complementary information and are not part of the core XML data.

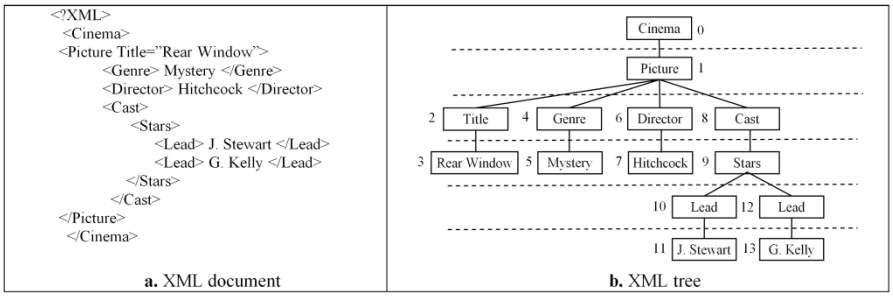


Fig. 3. A sample XML document with corresponding tree

In general, element/attribute values are disregarded when evaluating the structural properties of heterogeneous XML documents (originating from different data-sources and not conforming to the same grammar), so as to perform XML structural classification/clustering [22, 36, 57, 59] or structural querying (i.e., querying the structure of documents, disregarding content [10, 63]). Nonetheless, values are usually taken into account with methods dedicated to XML change management [18, 20], data integration [32, 46], and XML structure-and-content querying applications [64, 65], which is the main application in our current study. More formally:

Definition 1 - XML Tree: We represent an XML document as a rooted ordered labeled tree $T = (N_T, E_T, L_T, \Delta_T, g_T)$ where N_T is the set of nodes in T, $E_T \subseteq N_T \times N_T$ is the set of edges (element/attribute containment relations), L_T is the set of labels corresponding to the nodes of T ($L_T = El_T \cup Ev_T \cup Al_T \cup Av_T$ such as El_T (Al_T) and Ev_T (Av_T) designate respectively the labels and values of the elements and attributes of T), Δ_T is the set of data-types associated to the elements and attribute nodes of T (Δ_T= {*Concept*} $\cup$ $E\Delta \cup A\Delta$, having $E\Delta = A\Delta$ = {*Text*, *Number*, *Date*}), and g_T is a function $g_T : N_T \rightarrow L_T, \Delta_T$ that associates a label $l \in L_T$ and a data-type $t \in \Delta_T$ to each node $n \in N_T$. We denote by *root*(T) the root node of T, and by $T' \blacktriangleright T$ a sub-tree of T ●

Value data-types in the XML tree model are extracted from the corresponding XML schema. In other words, during XML tree construction time, the XML document and corresponding schema are assessed simultaneously so as to build the XML tree. Textual values are treated for stemming and stop word removal, and are mapped to leaf nodes of type *Text* in the XML tree. Numerical and date values are mapped to leaf nodes of types *Number* and *Date* respectively. As for the disambiguated element/attribute nodes, they are assigned the data-type *Concept*, their labels corresponding to the semantic concepts defined through the reference knowledge base. To model the XML data repository, we connect all XML trees to a single root node, with a unique label (e.g., *'Root'*).

4.2. Semantic Knowledge Representation and Indexing

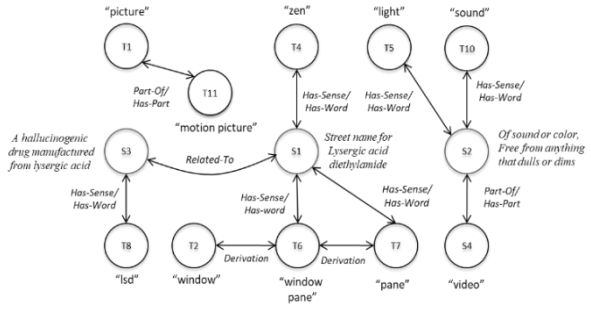
Semantic knowledge bases (i.e., thesauri, taxonomies, and/or Ontologies such as WordNet [51], Roget's thesaurus [90], and Yago [37]) provide a framework for organizing words/expressions into a semantic space [13]. A knowledge base is usually modeled as a semantic network made of a set of entities representing semantic concepts (or groups of words/expressions), and a set of links between the entities, representing semantic relationships (*synonymy*, *hyponymy*, etc.). We adopt a graph-based structure to model a semantic network from, where entities are represented as vertices, and the semantic relationships between entities are represented as directed edges. Formally:

Definition 1 — Semantic Network: A semantic network is represented as a graph SN (V, E, L, f_V, f_E) where:

- V is a set of vertices (nodes), designating entities in the semantic network. V includes both: i) *sense* nodes, representing semantic senses (*synsets*) with glosses, and ii) *term* nodes, representing literal words/expressions.
- E is a set of directed edges, an edge consisting of an ordered pair of vertices in V.
- L is a set of edge labels denoting semantic/lexical relationships. For WordNet, L includes: semantic relationships between concepts (e.g., *hyponymy*, *hypernymy*, *meronymy*), semantic relationships between concepts and terms (e.g., *has-sense* and *has-term*), and lexical relationships between terms (e.g., *derivation*).
- f_V is a function defined on V, designating the string value of each node in V. For WordNet, string values include: i) glosses/definitions, when dealing with *sense* nodes, and ii) and literal words/expressions,
- f_E is a function defined on E, assigning a label from L to each edge in E. Multiple edges may exist between the same pair of vertices when dealing with *term* nodes, which makes SN a multi-graph •

An extract from the WordNet lexical ontology is shown in Fig. 4, where S_1, S_2 and S_3 represent senses (i.e., synsets), and their string values (i.e., the synsets' glosses/definitions), and T_1, T_2, …, T_{11} represent terms, and their string values (i.e., literal words/expressions) shown along aside the nodes. Given that most semantic/lexical relationships are symmetrical (*hyponymy*/*hypernymy*, *meronymy*/*holonymy*, *has-sense/has-term*, etc.), and given that a relationship cannot exist without its symmetrical counterpart, we simplify our graph model by representing each couple of symmetrical relationships between senses and/or terms with one edge having opposite directions (instead of two edges), labeled with the names of the symmetrical relationships.

A simple inverted index *InvIndex*(*SN*) can be subsequently built for the textual tokens of each *SN* entity (i.e., string values of *term* nodes and *sense* nodes, cf. Fig. 4b) to speed up term/sense lookup when creating and then querying the XML structure.



a. Sample graph *SN* representing an extract from WordNet

Term	Sense IDs[]
"acid"	⟨S1, S3⟩
"clean"	⟨S2⟩
"light"	⟨S2⟩
"lsd"	⟨S3⟩
"lysergic"	⟨S1, S3⟩
"window pane"	⟨S1⟩
…	…

b. Extract of inverted index *InvIndex*(*SN*) connecting terms in *SN* with corresponding senses (to speed up term/synset lookup)

Fig. 4. Extract from the semantic graph of WordNet, with the corresponding index

4.3. XML Context-based Sense Disambiguation

Our XML sense disambiguation approach was introduced in [17, 78]. Here, we only provide an overview of the approach describing the constructs and methods required in our current study.

Different from previous approaches which limit XML context to the parent node [72, 73], to the root node path [70, 71], to the node sub-tree [85], or to nodes reachable through heuristically identified crossable edges [49], we introduce the notion of XML *Sphere-Ring* context, inspired from the sphere-search paradigm in XML IR [31], to consider the whole structural surrounding of an XML node, including its ancestors, descendants, and siblings, tuned to better describe the node's context. An XML *ring* w.r.t. to a given node consists of the set of nodes situated at a specific distance from the center node. An XML sphere encompasses all rings contained at distances lesser or equal to the size (diameter) of the sphere. The size of the XML sphere is tuned following the nature of the XML data at hand (e.g., certain XML trees might underline specialized and domain-specific data, and thus would only require small contexts so as to achieve relevant WSD results, whereas more heterogeneous and generic XML data might require larger contexts to better describe the intended meaning of each node label).

In addition, we extend the traditional bag-of-words WSD paradigm, adopting a relational information approach, i.e., considering the interconnections among XML nodes in computing disambiguation scores (in contrast with the classic bag-of-words approach [70-73], where all context nodes are treated as a homogeneous set of words regardless of their proximity/relations with the target node). We consider the structural distance separating the center node and each of its context nodes, following the intuition that the farther the context node from the sphere center, the lesser should be its impact in determining the semantic meaning of the center node label. Formally, consider $R_d(n)$ to be the ring corresponding to the center node n at distance d, i.e. the set of all nodes whose distance from n is d. Hence, the context sphere $S_D(n)$ of node n, with size D, consists of all the rings contained in $S_D(n)$, such that $S_D(n) = \{\text{all } R_d(n) \,/\, d \leq D\}$. Following our *Sphere-Ring* context model, node scores can be weighted following the sizes of the sphere rings to which they correspond, such that the larger the sphere ring

radius, the lesser the node weight. Hence, we can represent the context of a node *n* as a weighed vector, whose dimensions correspond to the all distinct nodes in its sphere context, weighted following their distances from the center node. In short, our approach:

- Integrates all notions of XML context, including ancestor, decedent, and sibling structural relations, which were considered separately in existing studies [70-73, 85],
- Allows the user/system administrator to manually and/or automatically tune the size of the XML context window following the nature and properties of the XML data at hand, in comparison with most existing static methods [70-73, 85],
- Extends the traditional bag-of-words WSD paradigm, adopting a relational information approach so as to consider the interconnections among XML nodes in computing disambiguation scores, in contrast with most existing methods using the traditional bag-of-words approach [70-73].

Once the contexts of all XML nodes have been determined, we process each target node label and its context node labels for WSD. Here, we evaluate the semantic similarity/relatedness between the target node label and each of its context node labels, by comparing the node's context with the context of each of its potential senses, extracted from the reference semantic source (a similar paradigm is utilized in [85] for XML node annotation). The idea is to first identify all possible senses of the target word node label in the reference semantic network. Consequently, we exploit the same notion of *Sphere-Ring*, which we adopted for XML trees (graphs), to identify the context of each potential sense in the reference semantic network (e.g., WordNet). Having computed the weighted context for the XML target node in the XML document tree (graph), and each of its possible senses in the semantic network, we compute the similarity between the node vector and each of its sense vectors. The sense vector yielding the highest similarity would underline the most meaningful sense describing the XML node label. This approach requires polynomial complexity: $O(|senses(x.\lambda)|\times(|S_D(n)|+|S_D(s_p)|))$, where $|S_D(s_p)|$ designates the maximum context sphere cardinality for any sense (concept) in the semantic network.

Note that to our knowledge, existing approaches have seldom provide a complexity and time performance analysis of their WSD methods. Despite being performed offline, nonetheless, WSD time performance remains potent w.r.t. practicability, when indexing documents published on Web. The proposed approach has to be: i) effective in identifying the correct senses, but also ii) reasonably efficient in order to be practically applied to the large corpora of XML documents published online. Here, the complexity of our combined XML sense disambiguation approach is polynomial and simplified to $O(|X|\times|senses(x.\lambda)|\times(|S_D(n)|+|S_D(s_p)|))$, where $|X|$ represents the number of nodes to be disambiguated in the target XML document.

4.4. XML Document Semantic Indexing

Having disambiguated all XML labels, the latter are replaced with their corresponding semantic concepts extracted from the reference semantic network (e.g., WordNet). Dedicated index structures (Concept-Doc and Concept-SN indexes [79-81], cf. Fig. 5) are utilized to handle the mapping between XML document labels and semantic network concepts. The output of the semantic document indexing process is a conceptual XML tree, i.e., an XML tree which labels consist of concepts with explicit

semantic definitions (which is at the core of the vision of the Semantic Web: Extending the WWW by giving information well defined meaning [76]).

Given an data collection *C*, an inverted index (also referred to as a posting file, or inverted list) built upon *C*, is (in its most basic form) a sorted list of index terms associated each with a set of object identifiers from *C*, disregarding structural information. In this study, we extend the basic inverted index to handle semi-structured data elements, introducing an *element-attribute* (*OA*) index:

Definition 2 - Element-Attribute (EA) Inverted Index: Given a XML data collection *C*, an *EA* inverted index built on *C*, denoted as $InvIndex_{EA}(C)$, is a structure of the form (*dom*(*A*), *EAs*, *f*) where:

- *dom*(*A*) designates the set of values within the domains of all attributes $\forall A_j \in C.A$. Considering text-only domains, values come down to textual tokens, i.e., *terms* (words/expressions),
- *EAs* designates the set of element (identifier)-attribute doublets, i.e., $EAs = \{(id(E_i), A_j)\} \ \forall E_i \in C$ and $\forall E_j \in C.A \ / \ \exists E_i.a_j \neq \emptyset$, where A_j is an attribute for which object E_i has a non-null value,
- *f* is a function mapping each $term \in dom(A)$ with a list of element-attribute doublets *EAs*[] designating the term's occurrence locations in *C*, i.e., $EAs[] = \langle (id(E_i), A_j) \rangle \ / \ term \in E_i.a_j$

A term used as textual token in the inverted index is referred to as index term, whereas the list of element-attribute doublets, i.e., *EAs*[], mapping to each index term is referred to as the term's posting list •

Consequently, we compute the semantic relatedness between each pair of node concepts in the XML tree. The idea is to produce a semantically weighted XML tree to be consequently exploited in keyword query processing (cf. Section 6). Here, various semantic similarity measures can be used (as briefly mentioned in the previous section): i) edge-based measures (computing semantic similarity based on the distance separating the concepts in the semantic network) [88], ii) node-based (computing semantic similarity based on the information content of each concept in the semantic network, w.r.t. a given text corpus) [47], and iii) gloss-based (comparing the glosses associated with each concept definition in the semantic network) [8]. Gloss-based approaches are particularly interesting in the context of WSD since they allow 'semantic relatedness' evaluation, which is a more general notion than '*semantic similarity*', including the latter as well as any kind of functional relation between terms [39] (e.g., *penguin* and *Antarctica* are not necessarily similar, but they are semantic related due to their *natural_habitat* connection), particularly *antonymy* (e.g., *hot* and *cold* are semantically dissimilar since they have opposite meanings, but they are semantically related).

A simple example depicting the semantic indexing of a sample XML tree is shown in Fig. 5. The sample XML document describes the movie *Rear Window*, one of *Alfred Hitchcock*'s masterpieces. While the XML labels seem meaningful and straightforward for a human user, nonetheless, they are highly ambiguous for a computer system. Most labels can be associated with more than 2 or 3 semantic senses (concepts) in WordNet reference. For instance, the label *Stewart* is associated with 2 semantic concepts: i) *James Stewart* (the leading actor who starred in *Rear Window*), and ii) *Dugald Stewart* (an 18th century Scottish philosopher). Likewise for most remaining labels in the input tree (e.g., *Kelly* underlines 3 semantic concepts, among which is *Grace Kelly*, the co-

star of *Stewart* in *Rear Window*; *plot* underlines 4 different senses, among which *movie plot*, etc.).

Recall that semantic XML document indexing is performed offline, as a pre-processing step prior to query evaluation, and does not affect the online computational complexity of the approach.

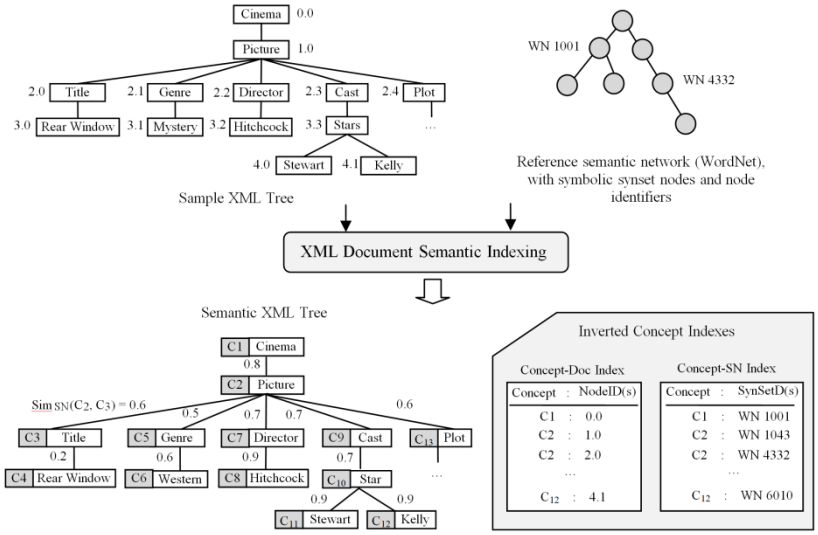


Fig. 5. Semantic analysis of XML document

5. Semantic Query Analysis

While semantic XML document analysis relies on the notion of XML context (e.g., the surroundings of a given node) in identifying the meanings of XML labels, nonetheless, semantic keyword query analysis differs in the lack of sufficient contextualization (keyword queries on the Web are usually 2-3 words long [15], which might not be sufficient in identifying a meaningful context [42, 49], cf. background in Section 2). To get round the lack of keyword contextualization in identifying meaningful query keyword senses, we introduce a method to *global query sense disambiguation*. Our proposal is based on the following assumption: *A keyword query on the Web usually conveys a certain global semantic meaning, reflecting a certain global information need*. Hence, rather than analyzing the individual senses of each query-term separately, considering each term's context information (similarly to most existing approaches, e.g., [35, 61]), we evaluate the aggregate semantic meaning of the query as a whole such that: the higher the semantic homogeneity of the query, the higher the consistency of the unified global semantic meaning conveyed by the query, and thus the more likely the query reflects the user's need. This is in accordance with the traditional assumption in WSD: *the most plausible assignment of senses to multiple co-occurring words is the one that maximizes the relatedness of meaning among the chosen senses* [52].

In short, we disambiguate the query as a whole, by i) pinpointing all possible configurations of query-term senses, and ii) consequently estimating a global semantic relatedness score (given a reference information source, e.g., WordNet) for all senses combined in each configuration. The configuration with the highest score would underline the most semantically meaningful query. Global query sense ranking can also be performed to identify the top most meaningful query sense configurations.

A major problem with the above approach is its computational complexity. In fact, computing semantic similarity/relatedness for all possible sense configurations for a set of lexical terms was shown to be intractable [52] due to its best case exponential complexity (i.e., $O(senses(k)^N)$ where N is the number of query keywords, and $senses(k)$ is the maximum number of senses per keyword). A few approximation methods have been proposed, such as computing pair-wise keyword similarities [52], and evaluating the similarity between each keyword sense and all remaining node senses [9]. Nonetheless, in contrast with existing approximation solutions, e.g., [9, 60], we introduce a sense disambiguation method to solve the computational complexity described above, producing optimal results similarly to the initial (exponential complexity) approach, while confining to polynomial complexity. We do so by transforming the problem of identifying all possible sense configurations, into that of identifying the shortest (semantic) path in a (semantically) weighted graph, using an adaptation of Dijkstra's shortest path algorithm [21]. In short, we capitalize on Dijkstra's polynomial computation approach to eliminate all unnecessary similarity computations, while still considering all possible query sense configurations.

Our query semantic analysis approach is described in the following sub-sections. Sub-section 5.1 presents our global query sense disambiguation approach, while Sub-section 5.2 describes our semantic query representation method. Recall that linguistic normalization (including *tokenization*, *expansion*, *stop word removal*, and *stemming*) is similar for both XML documents labels and query keywords, and will not be discussed hereunder

5.1. Structure-and-Content Query Model

In addition to the keyword query model, we put forward a structure-and-content query model to allow a higher level of expressiveness in querying semi-structured XML data. We suggest a simple model consisting of an XML tree variant with special leaf nodes to represent query predicates. A query with an *Or* logical operator is decomposed into a *disjunctive normal form* [64], and is thus represented as a set of XML trees, corresponding to the set of conjunctive queries.

Definition 3 – Structure-and-Content Query: It is expressed as an XML tree, $Q = (N_Q, E_Q, L_Q, T_Q, g_Q, n_d)$ encompassing a *distinguished* node n_d underlining the matches in the data tree that are required as answers to the query (i.e., the query's return clause). The query's root node $R(Q)$ designates its search scope/context. Its set T_Q encompasses the node type for distinguishing disambiguated XML nodes, and predicate types P_t_i corresponding to every value data-type t_i considered in the data model (e.g., $T_Q = \{Concept\} \cup \{P\_Text, P\_Number, P\_Date\}$) ●

Definition 4 - Query Node: It is a XML tree node with additional properties to represent predicates. With $n.t = P\_t_i$ (predicate corresponding to data-type t_i), the node's

label *n.l* underlines a composite content made of the predicate operator *n.l.op* and value *n.l.val* (e.g., leaf node $Q_1[2]$ of query Q_1 in Fig. 6 is of $Q_1[2].l.op$ = *'<'* and $Q_1[2].l.val$ = *'1965'*, having $Q_1[2].t$ = *P_Date*, which underlines that the predicate value *'1965'* is of type *Date*) ●

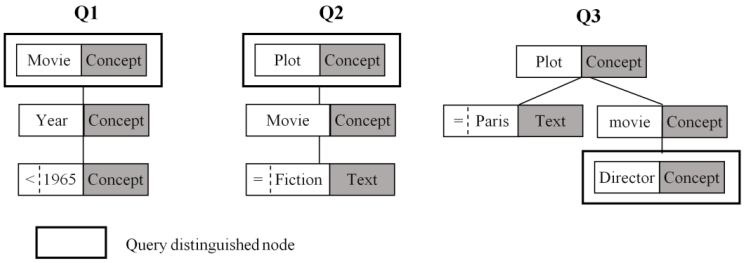


Fig. 6. Sample XML query trees

Note that each data-type has its own set of operators (e.g., {=, <, ≤, >, ≥, ≠} for numbers and dates, and {=, *like*} for text). Sample query trees are depicted in Fig. 6. Recall that query trees can be constructed via a dedicated GUI, which would suggest, on-the-fly, the list of possible query nodes following the context of the query at hand.

Definition 5 – Predicate Satisfaction: Given a predicate XML query node q_i, and a data node s_j, s_j satisfies q_i ($s_j \models q_i$) if:

- The data node type corresponds to that of the query ($s_i.t \approx q_i.t$, i.e., $\forall\, t_r \in$ {Text, Number, Date}, $q_i.t = P\_t_r \wedge s_j.t = t_r$),
- The data node label $s_j.l$ verifies the logical condition defined by $q_i.l$ ●

For instance, leaf node $T[13]$ of data tree T_2 in Fig. 3, having $T_2[13].l$ = *'1954'* and $T_2[13].t$ = *'date'*, satisfies predicate node $Q_1[2]$ of query Q_1 in Fig. 6, with $Q_1[2].l$ = *'<1965'* and $Q_1[2].t$ = *'date'*.

Definition 6 - Query Scope: Given a structure-and-content query *Q*, the scope of *Q* is identified by its root node *R*(*Q*), and corresponds to the XML sub-trees, in the data collection, having identical or semantically similar root nodes as that of the query ●

We assume that the user defines, with the query, the kind of XML data she is looking for, i.e. the scope/context of her query. If for instance the root of the query is labeled *University*, then XML data in the context of XML data entity *University*, or semantically similar entities such as *College*, *Academy*, etc., would naturally interest the user.

Definition 7 - *Template* and *Minimal constraint* querying: An structure-and-content query *Q* could be either evaluated as a i) *template* of the XML data the user is searching for, ii) or could represent the *minimal constraints* the data should meet to belong to the query answer set. In the former case, all query and data nodes would be considered in query/data similarity evaluation. Following the latter strategy, only elements required by the query tree are taken into account in query/data similarity evaluation, additional elements in the data tree being disregarded in the evaluation process ●

Note that XML queries most likely follow the *minimal constraint* style, the user usually specifying her information needs in the simplest form possible (cf. queries Q_1,

Q_2 and Q_3 in Fig. 6). Nonetheless, *template* querying could be particularly useful in *search-by-document* and *search-by-image* systems for instance, where the query could be a whole document or an SVG image [62] the user is searching for in the XML repository. A *template* style query could be any of the sub-trees in the XML tree of Fig. 3.

5.2. Global Query Sense Disambiguation

As mentioned previously, we assume that a query on the Web conveys a certain global semantic information request. The main objective is to associate each query-term with the appropriate semantic sense (concept) maximizing global query sense homogeneity. To do so, we proceed as follows:

Step 1 – Identifying Keyword Senses: The first step consists in identifying the set of possible senses corresponding to each individual query-term (keyword). Formally, for each keyword k_r, we obtain a set of senses $S_r = \{s^r_1, s^r_2, \ldots, s^r_{|Sr|}\}$ where s^r_i underlines the ith possible sense of keyword k_r extracted from the reference semantic network (e.g., WordNet), and $|senses(k_r)|$ the maximum number of possible senses corresponding to k_r. This first step is similar to most existing semantic based approaches, and the process is applied to structure-and-content queries.

Step 2 – Building the Semantic Query Graph: Having identified all possible senses for each query-term, we construct a semantic graph where each node represents of a possible keyword sense. The graph is structured in different layers, such that:

i. Each layer corresponds to a query-term, and consists of nodes representing all possible semantic senses for that query-term,
ii. The layers are ordered following the order of appearance of the query-terms in the keyword query,
iii. Nodes within the same layer (i.e., representing possible senses for the same term) are not connected to each other. In fact, same layer nodes underline senses of the same query-term and thus should not appear simultaneously in the same path (i.e., same query sense configuration),
iv. Each pair of nodes corresponding to two consecutive layers (i.e., describing the possible meanings of two consecutive query-terms), are connected together via a weighted edge, underlining the semantic distance (as an inverse function of semantic similarity/relatedness) between node senses,
v. Two virtual *start* and *end* nodes are added to the graph, connected to the nodes of the first/last graph layers respectively, via edges of null distances. These are introduced to guide the execution process of our adapted shortest path discovery algorithm (described hereunder),
vi. With content-and-structure queries, the query tree structure is considered when ordering the query nodes.

Step 3 - Identifying the Shortest Semantic Path: Consequently, the problem of identifying the most homogeneous configuration of query-term senses, simplifies to that of identifying the shortest semantic path in the semantic query graph. Here, we introduce an adaptation of Dijkstra's famous shortest path algorithm [21]. Our approach can be summarized as follows:

i. Initialize node distance scores such that: the *start node* score is set to zero, and all other node scores are set to infinity,
ii. Mark all nodes as *unvisited*, and set the *start node* as *current node*,

iii. For current node n_c, calculate the semantic distance with each of its connected nodes n_j in the consecutive layer, and preserve minimum distance scores, i.e., for each n_j, $Dist(n_j) = Min\{ Dist(n_c) + Weight(Egde(n_c, n_j)), Dist(n_j) \}$,
iv. When scores for all nodes connected to the current node n_c have been computed, n_c is marked as *visited*. A visited node would have a minimal and final distance score,
v. Select the *unvisited* node with the smallest distance score (from the initial node, considering all nodes in the graph) as the *current node* and continue from step 3,
vi. Terminate the algorithm when *end node* is deemed *visited*.

Consider keyword query *'Stewart Mystery Films'* (a similar process is applied to structure-and-content queries). The corresponding semantic query graph, built based on query-term semantic senses extracted from WordNet [50], is depicted in Fig. 7. Each graph layer corresponds to a query-term, and each node in a given layer underlines a semantic sense (concept) corresponding to the term at hand. The weight of an edge underlines the semantic distance between the connected nodes. Semantic distance can be computed as an inverse function of semantic similarity/relatedness, e.g., $Dist_{Sem} = 1 - Sim_{Sem}$. Recall that we adopt an aggregate semantic similarity/relatedness function combining *edge-based* methods [88], *node-based* methods [47], and *gloss-based* methods [8], w.r.t. WordNet. For ease of presentation, Fig. 7 shows sample semantic weight values for some (and not all) of the graph edges (e.g., $weight(edge(n_1, n_4)) = 0.3$ indicating that semantic concepts *James Stewart* and *Mystery story* are more similar than *James Stewart* and *Enigma*, having $weight(edge(n_1, n_3)) = 0.5$).

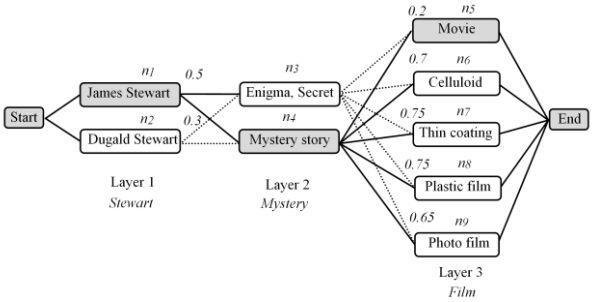


Fig. 7. Semantic analysis of keyword query

The result of applying our adapted shortest path algorithm to the semantic query graph in Fig. 7 is highlighted in the graph, and consists of nodes: n_1, n_4 and n_5. These underline the (WordNet) semantic concepts maximizing global query sense homogeneity: *James Stewart*, *Mystery story*, and *Movie*.

5.3. Semantic Query Representation

Having identified the best (i.e., most homogeneous) query sense configuration, we represent the query as a set of weighted semantic concepts (i.e., key-concepts), allowing the user to semantically expand the query, including additional concepts related to those originally conveyed by the query, in order to improve search result precision/recall.

Formally, a user keyword query Q consisting of a sequence of lexical keywords k_r, $Q = \{k_1, k_2, \ldots k_N\}$ is transformed into a semantic query representation Q_{Sem} consisting of a set of weighted concepts, $Q_{Sem}(D)= \{(c_1, w_1), (c_2, w_2), \ldots, (c_M, w_M)\}$ where c_i is a key-concept, w_i is the weight of c_i, and D is the query semantic depth parameter. The number of resulting key-concepts $M \geq N$ since additional key-concepts can be added following the user-chosen D expansion parameter as explained in the following. Semantic query expansion is performed using our *Sphere-Ring* model (cf. Section 4.1) to consider the semantic context of each query key-concept in the reference semantic network (e.g., WordNet). Note that the semantic contexts of query concepts can be determined, since the latter have already been disambiguated (as opposed to the pre-disambiguation keyword query where the semantic meanings of query-terms were undefined). The idea is to expand the query with additional concepts within the semantic vicinity of the original query key-concepts. Following our *Sphere-Ring* model, a semantic *ring* $R_d(c)$ w.r.t. to a given concept c consists of the set of concept nodes, in the reference semantic network, situated at a specific distance d from the target concept node c. The semantic context sphere SD(ci) encompasses all semantic rings contained at distances lesser or equal to the size (diameter D) of the sphere, such that SD(c) = {all Rd(c) / d ≤ D}. The sphere context size is specified by the user as a query semantic depth parameter:

- For $D = 0$, the query is represented with it original key-concepts, associated maximum (unit, =1) weights,
- For $D > 0$, the query is expanded with concepts situated within each original key-concept's semantic sphere (in the reference semantic network). Expanded query concepts are weighted such that concepts farther away from the semantic sphere center have a larger semantic distance w.r.t. the sphere's center, and hence should have a lesser impact on the query's semantic meaning. Following our *Sphere-Ring* context model, concept weights can be computed following the sizes of the sphere rings to which they correspond, such that the larger the sphere ring radius, the lesser the concept weight (e.g., a given weight decay function could be computed as $weight(c_i) = w_i = \frac{1}{1+d} \in [0, 1]$ having $c_i \in R_d(c) \subset S_D(c)$). Note that parameter D can be normalized in the $[0, 1]$ interval, following the maximum depth of the reference semantic network SN at hand (e.g., $\frac{D}{Depth(SN)}$), to simplify the user's task in specifying the expansion threshold.

Consider for instance the sample keyword query Q = *'Stewart Mystery Films'*:

- For D = 0, $Q_{Sem}(0)$ = {(James Stewart, 1), (Mystery story, 1), (Movie, 1)},
- For $D = 1$, the resulting query representation includes all semantic concepts appearing in the unit ($D=1$) semantic context spheres of each original key-concept. Here, following the WordNet extracts in Fig. 8, the semantic context of concept *James Stewart* includes concept *Actor* (cf. Fig. 8.a). Likewise, the semantic context of concept *Mystery movie* includes *Story*, *Detective story* and *Murder story* (Fig. 8.b). The semantic context of concept *Movie* includes *Show*, and *17* children (hyponym) concepts including *Telefilm*, *Feature film*, *Final cut*, *Home movie*, etc., (the remaining child concepts are omitted here for ease of presentation, cf. Fig. 8.c). The weights of all expanded concepts are equal to $\frac{1}{1+d} = \frac{1}{1+D}$ = 0.5, following our adopted decay function. Hence, the semantic query becomes:

$Q_{Sem}(1)$={ (James Stewart, 1), (Actor, 0.5), (Mystery story, 1),
(*Story*, 0.5), (*Detective story*, 0.5),
(*Murder story*, 0.5), (*Movie*, 1), (*Show*, 0.5),
(*Telefilm*, 0.5), (*Final cut*, 0.5), (*Home movie*, 0.5) }

The time complexity of our global query disambiguation approach comes down to that of the shortest path computation process, which comes down to almost linear $O(N\times\log(N))$ time where $N=|S_D(c)|\times|Q|\times|senses(k_r)|$. The latter simplifies to $N=|S_D(c)|\times|senses(k_r)|$ since $|Q|$ is usually limited to 2-3 keywords [15] and can be omitted as a fixed parameter.

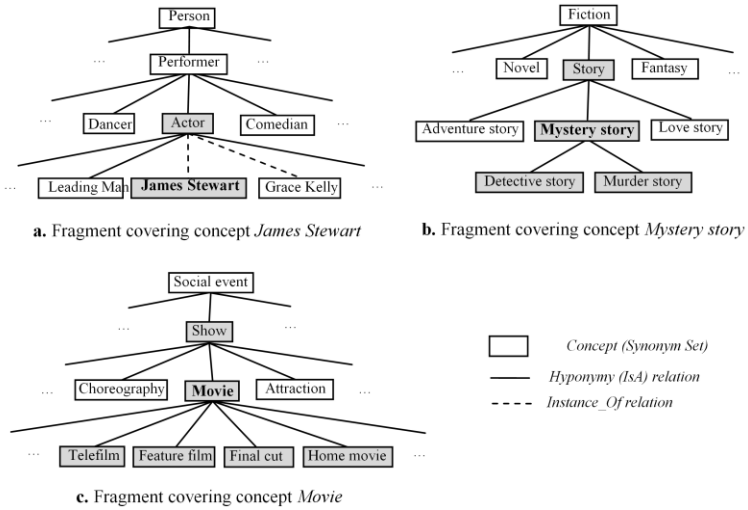


Fig. 8. Taxonomy fragments extracted from WordNet, covering the key-concepts in our example

6. Semantic Query Processing

6.1. Candidate Answer Tree

The first step in assessing a query is to identify its search scope. Following the traditional IR logic, whole physical files are considered as candidate answers. Nonetheless, XML documents differ in their structural organization and granularity: some documents may contain information about *movies*, while others include information about *actors* acting in many *movies*. Hence, it is not relevant to retrieve the entire *movie* when the user is searching for certain *actors*. Hence, the XML query search scope should be identified dynamically, considering the query at hand.

Following our XML data and query models, the query scope can be identified as the set of XML data sub-trees (which we identify as Candidate Answer Trees, *CAT*s), in the data repository, having identical, or semantically similar enough, root nodes as that of the query (i.e., same/similar label, with the same data-type). Consider for instance query Q_1, searching for *movies* that have certain characteristics. When considering root

node identity, query Q_1's *CAT*s would be all data sub-trees having root node *movie*. When taking into account semantic similarity, Q_1's *CAT*s would also encompass subtree T_1 of root node *picture* from Fig. 5.

Definition 8. Candidate Answer Tree: Given an XML node similarity measure $Sim_{Semantic}$, and reference semantic network *SN* for evaluating the semantic similarity between XML concept and node labels, and a semantic similarity threshold α, the set of candidate answer trees Q_{CAT}, for a given query *Q*, in an XML data collection *C*, $Q_{CAT} = \{S / S \blacktriangleleft C \wedge ((R(Q) = R(S) \text{ if } \alpha = 1) \vee Sim_{Semantic}(R(Q), R(S), SN) \geq \alpha \text{ otherwise})\}$ ●

The semantic similarity threshold also serves as a structural/semantic similarity parameter, underlying the extent of structural/semantic similarity considered while identifying candidate answers. It allows the user to assign more importance to the structural or semantic characteristics of XML data in answering the query at hand:

- For $\alpha = 1$, only *CAT*s with root nodes identical to that of the query are the only ones considered. This corresponds to purely structural querying.
- For $0 < \alpha < 1$, *CAT*s with root nodes of semantic similarity higher than α are considered. As α decreases, the size of the answer set Q_{CAT} will increase, following the semantic similarities between query and *CAT* root nodes.
- For $\alpha = 0$, all data sub-trees in the XML data collection are considered as *CAT*s.

As for the semantic similarity measure $Sim_{Semantic}$, it is evaluated w.r.t. the nodes' constituents, i.e. their concepts and tag labels, where existing semantic similarity measures (e.g. Lin [47], Wu and Palmer [88]) can be exploited (cf. background in Section 2), taking into account the concerned reference semantic network. In our approach, our measure consists of a linear combination of Lin [47], and Wu and Palmer [88]), assigning equal weights to both measures. Other measures can be used according to the admin user's preferences.

6.2. Relevance Weight Function

We introduce a set of weighting functions to assign weight scores to XML nodes and edges, allowing to weight and rank the candidate answer trees. Considering an XML node n_i in the semantic XML tree, the weight of n_i is computed according to the below formula where we consider "Fan-in" to be the number of nodes connected with the target XML node:

$$W_{XMLNode}(n_i) = \frac{Fan-in(n_i)}{\underset{\forall v_j \in V_{index}}{Max}\ (Fan-in(n_j))} \in [0,1] \quad (1)$$

The rational is that an XML node is more important if it shares more links from other XML nodes. Given an XML edge e_i^j connecting XML nodes n_i and n_j in the XML tree, we define the weight of e_i^j as follows:

$$W_{XMLEdge}(e_i^j) = \frac{1}{Fan-out_{Label}(n_i)} \in]0,1] \quad (2)$$

The weight of an XML edge is inversely proportional to the number of links from a certain node to another, taking into account the semantic relation type of the link at hand (e.g., parent-child, element-attribute, element-value). The rationale here is that an XML edge designates a stronger connection between two XML nodes when it carries most of

the descriptive power from the source node to the destination node, such that the source node has few other out-going connections.

The scores of XML nodes/edges returned as query answers are computed using typical *Diskstra*-style shortest distance computations. Yet, instead of identifying the shortest (smallest) distance, we identify as answers XML sub-tree root nodes having the maximum similarity (similarity being the inverse function of distance) w.r.t. the starting nodes (mapping to keyword queries). In other words, given the sample CAT T in 0, with root node $n_d = root(T)$ and leaf nodes $n_{i...j}$, we define the relevance score of n_d w.r.t. $n_{i...j}$ as follows:

$$score(n_d, n_{i...j}) = \frac{\sum_{i...j} \frac{W_{XMLNode}(n_d) \times W_{XMLEdge}(e_p^d) \times \frac{1}{d(n_p, n_d)} + ... + W_{XMLNode}(n_x) \times W_{XMLEdge}(e_i^x) \times \frac{1}{d(n_i, n_d)}}{d(n_i, n_d)}}{|n_{i...j}|} \in [0,1] \quad (7)$$

where $d(n_i, n_d)$ is the distance in number of edges between two nodes, and $|n_{i...j}|$ is the number of leaf nodes in the CAT T rooted at n_d. In other words, in the following example, $d(n_p, n_d) = 1$, $d(n_j, n_d) = 2$, and $d(n_i, n_d) = 3$.

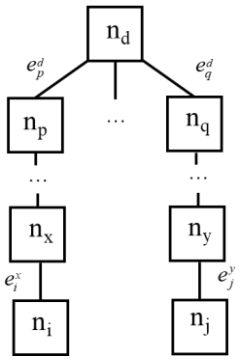


Fig. 9. Sample node linkage in an XML candidate answer tree

6.3. Semantic Query Processing

Having transformed the XML document collection and the keyword query into meaningful semantic representations, XML semantic search comes down to identifying and ranking the most relevant semantic XML sub-structures encompassing the semantic key-concepts in the query. Our extended framework includes three query processing algorithms: i) the core semantic search algorithm and two other variants designed to improve: ii) user involvement, and iii) query efficiency:

i. Core algorithm: titled *Semantic Search* and originally described in [84], it performs semantic-aware search using shortest path navigation in the *SemIndex* graph,
ii. User involvement: *Query-As-You-Type Search*, allows users to manually choose the meanings of query keywords before performing semantic search, aiming to involve the user in improving search result quality,

iii. Query Efficiency: *Parallel Semantic Search* is a parallel processing (multithreading) version of *SI_SS*, aiming to reduce query execution time.

6.4. Semantic Search

Our main querying method is based on a structural clustering technique to group together key-concept occurrences, in the XML data collection, which are structurally close. Our objective is to identify and rank the most prominent candidate answer sub-trees, in the XML data set, containing related occurrences of query key-concepts. Our semantic search algorithm is shown in Fig. 10 and is described below:

Step 1 - Identifying concept occurrences: The first step consists in pinpointing the XML nodes, in the data collection, containing occurrences of the query key-concepts.

Step 2 - Performing XML node clustering: Having identified the XML nodes encompassing key-concept occurrences, we perform structural clustering [55] to group together the XML nodes which are closest in the XML tree. The algorithm is applied on the weighted distances separating concept occurrences (cf. Section 6.2).

Step 3 – Constructing Answer Trees: We construct candidate answer trees based on the XML node clusters. An answer tree consists of the sub-tree rooted at the lowest common ancestor of all concept occurrences in the corresponding cluster.

Step 4 – Ranking Answer Trees: Having identified the candidate answer sub-trees, we rank them following their relevance to the query. Here, we utilize an integrated function combining various ranking criteria including i) weights of semantic concepts; ii) answer tree size (compactness), iii) common usage of senses (e.g., WordNet estimates the average usage frequency of word meanings in the English language, following the Brown corpus [29]), where the most commonly used senses are deemed more relevant in ranking results [49]. Other weighting functions can be used.

```
Algorithm SemanticSearch
Input:  T        // Semantic XML tree
        K        // Set of query selection terms
        D        // Sphere diameter designating query context size

Ouput: N_Out     // List of ranked trees from T designating query answers

Begin                                                                    1
N_Out = ϕ                                                                2
Step 0:  S = getSemanticQuerySenses(K)   // Global disambiguation        3
For each term s_i ∈ S                    // For each keyword sense
{                                                                        4
    Step 1: n_In = getNodeID(s_i, T)    // Identify concept occurrences
    Step 2: SP = PerformClustering(n_In, D, T)                           5
    Step 3: N_init = constructAnswerTree(SP, T)                          6
    Step 4: N_Out = rankAnswerTree(N_init, T)                            7
}                                                                        8
  Return N_Out                                                           9
                                                                        10
End
```

Fig. 10. Pseudo-code of *Semantic Search* algorithm

Note that the complexity of the semantic search algorithm comes down to the complexity of the structure clustering algorithm in Step 2. We utilize Lloyd's heuristic algorithm [48] to bound clustering complexity to $O(N \times C \times I)$ where N is the number of XML nodes to be clustered, C the number of produced clusters, and I the number of iterations to reach convergence.

6.5. Query As You Type Search

This algorihtm allows the user to choose the proper meaning for every query keyword, by allowing her to choose the intended sense from the set of all possible senses provided by WordNet. Once the senses have been chosen, the algorithm pinpoints in semantic network graph the indexing nodes corresponding to the chosen senses, and then runs typical shortest patth search starting from the chosen nodes. The pseudo-code of is basically the same as that of *Semantic Search*, except for adding a *step 0*: n_{In} = *manual*(*K*, *SN*), i.e., allowing the user to manually choose the proper meaning of every query term, among the list of possible meanings presented to the user through the system's GUI (cf. Fig. 11). Then, *Semantic Search* resumes by identifying and only processing the starting nodes corresponding to the term senses (synsets) chosen by the user. The algorithm's main steps can be described as follows:

i. Allow the user to choose the sense of each term in the query according to WordNet,
ii. Identify in the semantic XML tree the nodes corresponding to the chosen senses,
iii. Run the resulting query, starting from the identified index nodes, as a typical semantic keyword query search.

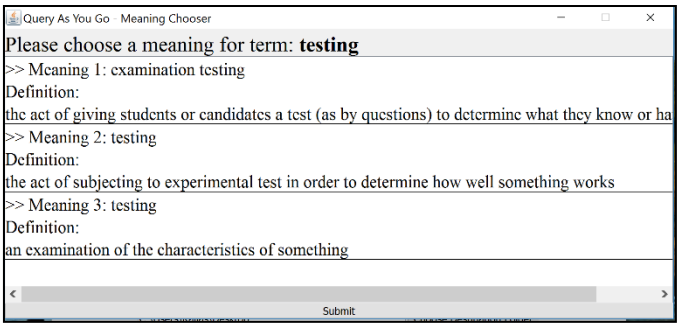


Fig. 11. *Query-As-You-Type* sub-interface

6.6. Parallel Semantic Search

We have also introduced a parallelized version of algorithm *Semantic Search* (cf. Fig. 12), which preserves (more or less) the same workflow of the original algorithm except

that it processes query terms and starting XML nodes using multiple threads running in parallel. The algorithm's main steps are described as follows:

i. Every query term is assigned a dedicated thread, and is thus processed independently from other threads (lines 1-2),
ii. After identifying the starting nodes for a query term (line 4), every starting node is then assigned its own dedicated thread (line 5), allowing to: compute the shortest paths from the starting node to data nodes in the XML tree (line 7), and then identify the reached data nodes designating potential query answers (i.e., CATs, line 8),
iii. Results are gradually merged (line 9) as they are produced by each thread, to rank and select (lines 10-12) query answers.

The implementation of the algorithm is configured to run as many threads as there are terms in the user query, where thread scheduling and parallel execution is left to the operating system.

```
Algorithm ParallelSemanticSearch

Input:  T        // Semantic XML tree
        K        // Set of query selection terms
        D        // Sphere diameter designating query context size

Ouput: N_Out     // List of ranked trees from T designating query answers

Begin                                                                    1

N_Out = ϕ                                                                2

Step 0:  S = getSemanticQuerySenses(K)   // Global disambiguation         3

Create Thread for each term s_i ∈ S       // For each keyword sense       4
{
    Step 1: n_In = getNodeID(s_i, T)   // Identify concept occurrences    5
    Create Thread for each n_i ∈ n_In                                     6
    {                                                                    7
        Step 2: SP = PerformClustering(n_i, D, T)                        8
        Step 3: N_init = constructAnswerTree(SP, T)                      9
        Step 4: N_Out = rankAnswerTree(N_init, T)                        10
    }                                                                    11
}                                                                        12
Return N_Out                                                             13

End
```

Fig. 12. Pseudo-code of *Parallel Semantic Search* algorithm

7. Experimental Evaluation

7.1. Experimental Scenario

We conducted a battery of experiments to test and evaluate our approach. We used a collection of 80 test documents gathered from several data sources having different properties[1]. Target XML nodes were first subject to manual disambiguation (12-to-13 nodes were randomly selected per document, yielding a total of 1000 target nodes, allowing human testers to annotate each node by choosing appropriate senses from WordNet) followed by automatic disambiguation. We formulated different with varying numbers of keywords, e.g., from 1 (single term query) to 5, where each query expands its predecessor by adding an additional selection term to the latter cf. sample queries in Table 1). We then compared user and system generated senses to compute *precision* (PR), *recall* (R), *f-value*, and *mean average precision* (MAP) scores.

Table 1. Sample test queries used in our experiments

Query *Q1*	
ID	Terms
Q1_1	"music"
Q1_2	"music", romance"
Q1_3	"music", romance", "dinner"
Q1_4	"music", romance", "dinner", "trip"
Q1_5	"music", romance", "dinner", "trip", "Paris"

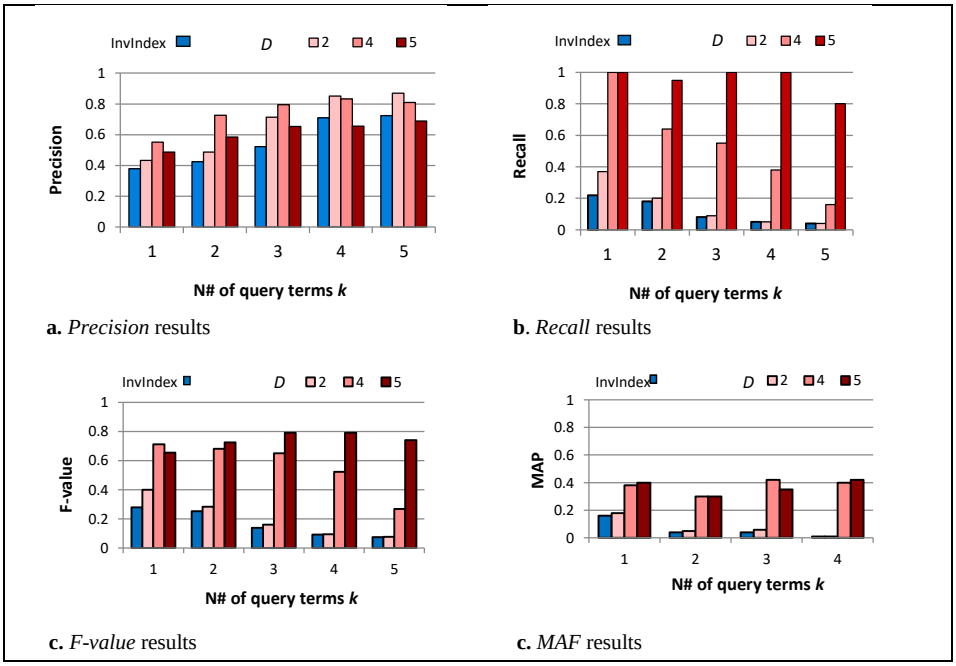
Query *Q2*	
ID	Terms
Q2_1	"play"
Q2_2	"play", "theater"
Q2_3	"play", "theater", "scene"
Q2_4	"play", "theater", "scene", "hero"
Q2_5	"play", "theater", "scene", "hero", "climax"

7.2. Query Result Quality

We first tested the effectiveness of our approach considering its different features and configurations: i) the properties of XML data (w.r.t. ambiguity and structure), and ii) context size (sphere neighborhood radius). Results in Fig. 13 show that precision levels increase with the number of query terms *k*. This is due to the human testers' expectations: given that queries are expanded versions of one another, result quality is evaluated based on the user's intent: which is expressed with the most expanded (i.e., most expressive) query (e.g., *Q1_5* and *Q2_5*). One can realize that using fewer query terms produces lower precision levels, which is due to the system returning more results which are (semantically related to the query terms but which are) not necessary related to the user's intent. As for recall, one can realize that levels steadily increase with concept depth *D*, where the number of correct (i.e., user expected) results returned by the system increases as more semantically related terms are covered in the querying process. F-value results increase with the increase of context depth *D*, and they slightly decrease with the increase of the number of query keywords *k*. This confirms the precision and recall results, where the determining factor affecting retrieval quality

[1] Shakespeare collection http://metalab.unc.edu/bosak/xml/eg/shaks200.zip, Amazon product files simply-amazon.com/content/XML.html,SIGMOD Record, http://www.acm.org/sigmod/xml, Niagara collection http://www.cs.wisc.edu/niagara/

remains context depth *D*. An increase in the number of keywords *k* tends to reduce system recall with higher values of *k* (queries becoming very selective, thus missing some relevant results). F-value levels are significantly higher than those obtained with the legacy inverted index, highlighting a clear improvement over syntactic retrieval quality. Also, mean average precision levels seem to concur with those of *f-value*, such that the ranking of relevant results compared with non-relevant ones in the queries' result lists seems to increase with the increase of *D* and fluctuate (based on the values of *D*) with the increase of *K*. In other words, increasing *D* not allowed retrieving more relevant results and improved the ranking of relevant results w.r.t. non-relevant ones in the query result list.



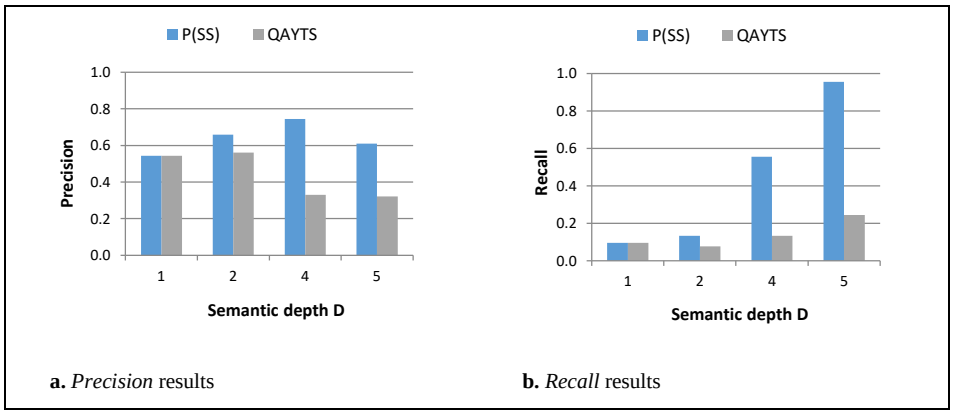
a. *Precision* results

b. *Recall* results

c. *F-value* results

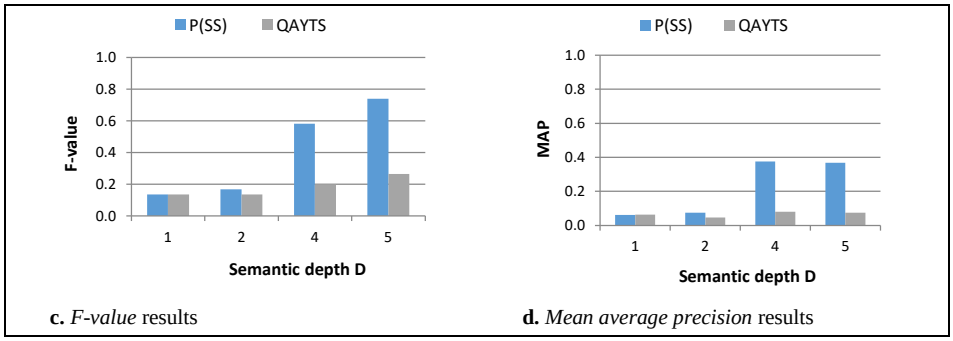
c. *MAF* results

Fig. 13. Comparing *Semantic Search* average precision (PR), recall (R), f-value, and mean average precision results with legacy inverted index syntactic search



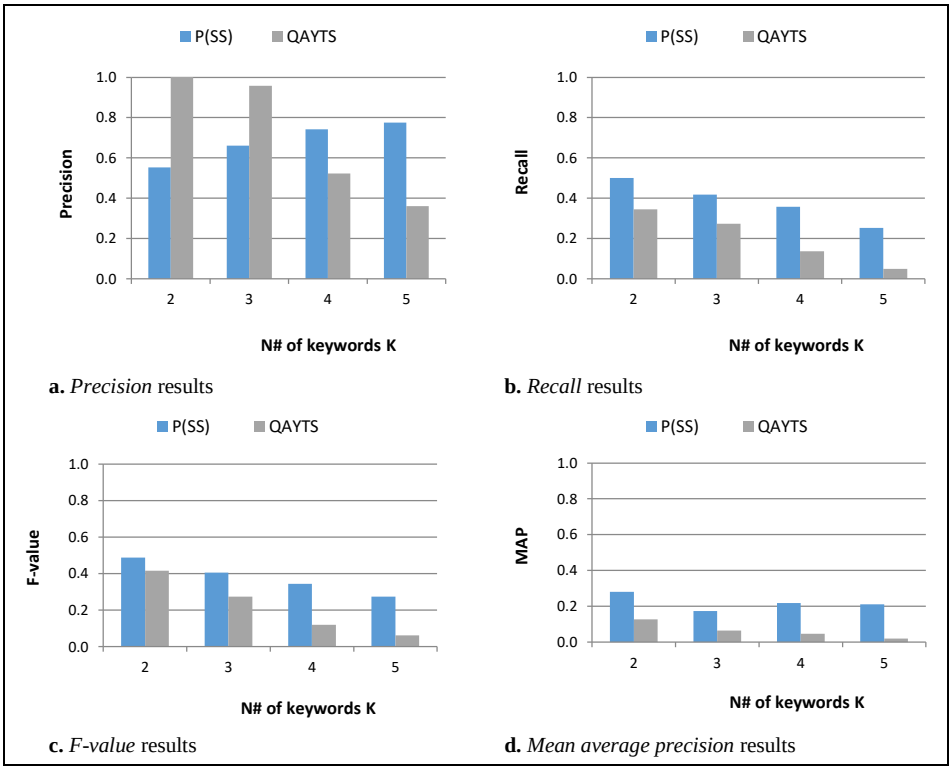
a. *Precision* results

b. *Recall* results



c. *F-value* results **d.** *Mean average precision* results

Fig. 14 Comparing *Semantic Search* (*SS*), *Query As You type Search* (*QAYTS*), and *Parallel semantic Search* (*PSS*) average precision (PR), recall (R), f-value, and mean average precision (MAP) results when varying semantic depth *D*



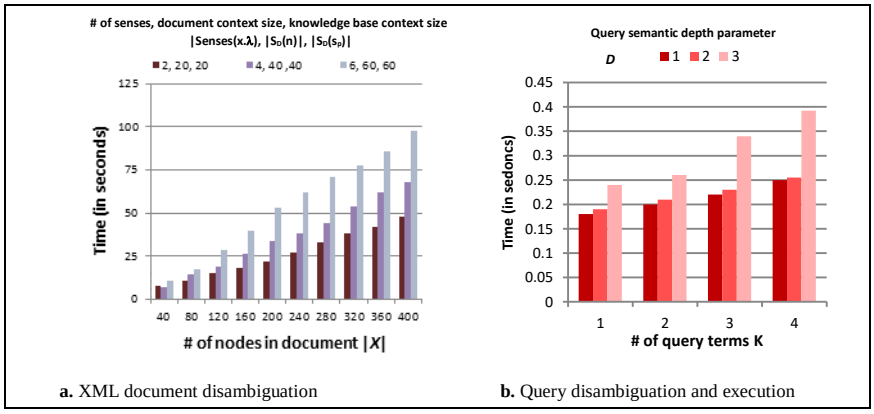
a. *Precision* results **b.** *Recall* results

c. *F-value* results **d.** *Mean average precision* results

Fig. 15. Comparing *Semantic Search* (*SS*), *Query As You type Search* (*QAYTS*), and *Parallel semantic Search* (*PSS*) average precision (PR), recall (R), f-value, and mean average precision (MAP) results when varying semantic depth *D*

7.3. Query Processing Time

We evaluated our solution's almost linear efficiency. Results in Fig. 16 highlight the polynomial (almost linear) complexities of both our (offline) XML document disambiguation and (online) global query disambiguation approaches, considering different parameter configurations for both processes. Results in Fig. 16.b show total query execution time including online disambiguation, by varying both the number of keywords and query semantic depth *D* (i.e., semantic context size).

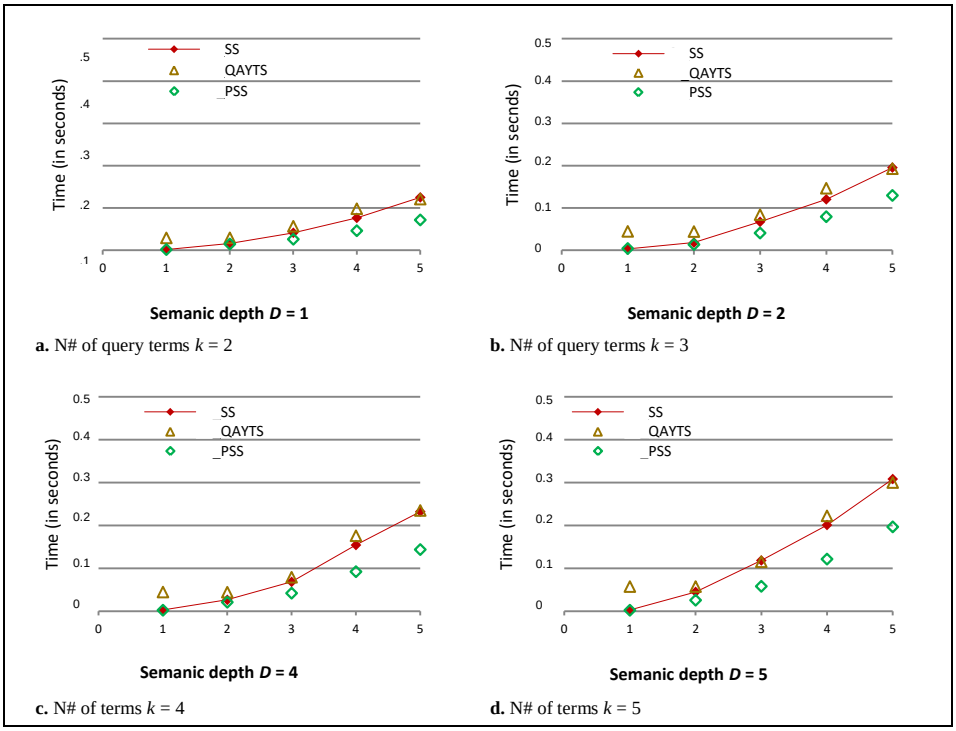


a. XML document disambiguation **b.** Query disambiguation and execution

Fig. 16. XML document disambiguation time (a) and query disambiguation and execution time (b)

We ran the same queries through the three querying algorithms: *SS*, *QAYTS*, and *PSS*. Fig. 17 provides average processing time results for all queries, plotted by varying the number of query terms *K* and link distance threshold *D*. First, results of all three algorithms show that query execution time increases almost linearly with the number of query terms *K* (when fixing link distance *D*), and increases linearly with *D* (when fixing *K*), highlighting the algorithms quadratic complexity levels. Second, results show that all three algorithms have very close query time levels when both *K* and *D* are small (=1 and 2), such that time difference increases as both *K* and *D* increase. This is due to the fact increasing either *K* or *D* means increasing the number of nodes to be navigated in the semantic XML tree: increasing *k* means navigating the XML tree starting from a larger number of initial nodes, and increasing *D* means reaching deeper into the tree structure to identify more semantically relevant results. Third, algorithms *SS* and *QAYTS* produced almost identical time levels (disregarding the manual effort required in *QAYTS*[2]), whereas the parallel processing *PSS* algorithm is clearly the most efficient of its counterparts, requiring almost 33.34% less time than *SS* and *QAYTS* with maximum *k*=5 and *D*=5.

[2] *QAYTS*'s time shown in Fig. 16 does not encompass the time it took the testers to manually choose the meanings of query terms (which we did not consider to be part of the algorithm itself), but only considers actual algorithm (CPU and SQL) execution time.



a. N# of query terms $k = 2$ **b.** N# of query terms $k = 3$

c. N# of terms $k = 4$ **d.** N# of terms $k = 5$

Fig. 17. Comparing average query execution time of *Semantic Search* (*SS*), *Query As You type Search* (*QAYTS*), and *Parallel semantic Search* (*PSS*), while varying semantic depth *D* and the number of query terms *K*

8. Conclusion

In this paper, we describe *XSemSearch*, a solution for XML keyword search allowing to transform both XML documents and keyword queries into semantic representations, using semantic concepts in a reference knowledge base. We describe two approaches for i) offline context-based XML document disambiguation and ii) online global keyword query disambiguation, both designed to run in almost linear time. Our solution is: i) fully automated, compared with existing interactive solutions which require user input to manually identify the intended query senses e.g., [35, 61], and ii) tractable (of almost linear time) and thus reasonably applicable on the Web, compared with polynomial or exponential solutions, e.g., [23, 58]. Our solution also provides iii) a dedicated index structure to handle semantic XML trees, iv) a dedicated query formalism to allow structure-and-content queries with only partial knowledge of the data collection structure and semantics, and iv) three alternative query processing algorithms to evaluate query processing time and quality.

We are currently investigating the integration of semantic-aware indexing capabilities [79-81] and different clustering algorithms to form XML answer trees [33, 77]. This would provide more opportunities toward both speed-ups and semantic-based

filtering. We are also investigating the use of alternative knowledge sources such as Google [1], Wikipedia [86], and FOAF [4] to acquire a wider word sense coverage, and explore our approach in practical applications, namely semantic-aware document and schema matching [82, 83], RSS news feed merging [68, 69], affective blog analysis [26, 27], social event detection [3, 5], and semantic relations' identification from social media data [2]. On the long run, we aim to investigate word embeddings and learning statistical distributions in a corpus [34, 92], to infer semantics without the need for predefined knowledge bases.

References

1. Abdulhayoglu M. and Thijs B., *Use of ResearchGate and Google CSE for author name disambiguation.* Scientometrics 2017. 111(3): 1965-1985.
2. Abebe M., et al., *Generic Metadata Representation Framework for Social-based Event Detection, Description, and Linkage.* Knowledge Based Systems 2020. 188.
3. Abebe M. A., et al., *Overview of Event-Based Collective Knowledge Management in Multimedia Digital Ecosystems.* International Conference of Signal Image Technology and Internet-based Systems (SITIS'17), 2017. pp. 40-49.
4. Amith M., Fujimoto K., Mauldin R.,, and Tao C., *Friend of a Friend with Benefits ontology (FOAF+): extending a social network ontology for public health.* BMC Medical Informatics & Decision Making - Supplement, 2020. 20-S(10): 269.
5. Ashagrie M., et al., *A General Multimedia Representation Space Model toward Event-based Collective Knowledge Management.* Submitted to 19th IEEE International Conference on Computational Science and Engineering (CSE 2016), 2016. Paris, France.
6. Azzini A., et al., *A Neuro-Evolutionary Corpus-based Method for Word Sense Disambiguation.* IEEE Intelligent Systems, 2012. 27(6): 26-35.
7. Baeza-Yates R. and Ribeiro-Neto B., *Modern Information Retrieval: The Concepts and Technology behind Search.* ACM Press Books, Addison-Wesley Professional, 2nd Ed., 2011. p. 944.
8. Banerjee S. and Pedersen T., *Extended Gloss Overlaps as a Measure of Semantic Relatedness.* International Joint Conference on Artificial Intelligence (IJCAI'03), 2003. p. 805-810.
9. Baziz M.; Boughanem M. and Traboulsi S., *A concept-based approach for indexing documents in IR.* INFORSID 2005, 2005. pp. 489-504, Grenoble, France.
10. Bertino E.; Guerrini G.; and Mesiti, M., *A Matching Algorithm for Measuring the Structural Similarity between an XML Documents and a DTD and its Applications.* Elsevier Information Systems, 2004. (29):23-46.
11. Bobed C. and Mena E., *QueryGen: Semantic Interpretation of Keyword Queries over Heterogeneous Information Systems.* Information Sciences, 2016. 329: 412-433.
12. Bonab H., et al., *Incorporating Hierarchical Domain Information to Disambiguate Very Short Queries.* International Conference on the Theory of Information Retrieval (ICTIR'19), 2019. pp. 51-54.
13. Budanitsky A. and Hirst G., *Evaluating WordNet-based Measures of Lexical Semantic Relatedness.* Computational Linguistics, 2006. 32(1): 13-47.
14. Burton-Jones A.; Storey V.C.; Sugumaran V. and Purao S., *A Heuristic-Based Methodology for Semantic Augmentation of User Queries on the Web.* In Proceedings ot the International Conference on Conceptual Modeling (ER'03), 2003. pp. 476–489.
15. Calì A., Martinenghi D., and Torlone R., *Keyword Queries over the Deep Web.* International Conference on Conceptual Modeling (ER'16), 2016. pp. 260-268.

16. Chaplot D. and Salakhutdinov R., *Knowledge-based Word Sense Disambiguation using Topic Models.* AAAI Conference on Artificial Intelligence (AAAI'18), 2018. pp. 5062-5069.
17. Charbel N., et al., *Resolving XML Semantic Ambiguity.* International Conference on Extending Database Technology (EDBT'15), 2015. Brussels, Belgium, pp 277-288.
18. Chawathe S.; Rajaraman A.; Garcia-Molina H.; and Widom J., *Change Detection in Hierarchically Structured Information.* Proceedings of the ACM International Conference on Management of Data (SIGMOD), 1996. pp. 26-37. Montreal.
19. Che D., Ling T., and Hou W., *Holistic Boolean-Twig Pattern Matching for Efficient XML Query Processing.* IEEE Transactions on Knowledge and Data Engineering, 2012. 24(11): 2008-2024.
20. Cobéna G.; Abiteboul S.; and Marian A., *Detecting Changes in XML Documents.* Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2002. pp. 41-52.
21. Cormen T. H.; Leiserson C. E.; Rivest R. L. and Stein C., *Introduction to Algorithms (Second ed.) - Section 24.3: Dijkstra's Algorithm.* MIT Press and McGraw-Hill, 2001. pp. 595–601.
22. Dalamagas T.; Cheng T.; Winkel K.; and Sellis T., *A Methodology for Clustering XML Documents by Structure.* Information Systems, 2006. 31(3):187-228.
23. de Campos L., et al., *XML Search Personalization Strategies using Query Expansion, Reranking and a Search Engine Modification.* ACM Symposium on Applied Computing (SAC'13) 2013. pp. 872-877.
24. Demidova E., ZhouIrina X., and Nejdl O., *Evaluating Evidences for Keyword Query Disambiguation in Entity Centric Database Search.* International Conference on Database and Expert Systems Applications (DEXA'10), 2010. pp. 240-247.
25. Di Iorio A., et al., *A First Approach to the Automatic Recognition of Structural Patterns in XML Documents* ACM Symposium on Document Engineering, 2012. pp. 85-94.
26. Fares M., et al., *Difficulties and Improvements to Graph-based Lexical Sentiment Analysis using LISA* IEEE International Conference on Cognitive Computing (ICCC'19), 2019.
27. Fares M., et al., *Unsupervised Word-level Affect Analysis and Propagation in a Lexical Knowledge Graph.* Elsevier Knowledge-Based Systems, 2019. 165: 432-459.
28. Fragos K., *Modeling WordNet Glosses to Perform Word Sense Disambiguation.* International Journal of Artificial Intelligence Tools, 2013. 22(2).
29. Francis W. N. and Kucera H., *Frequency Analysis of English Usage.* Houghton Mifflin, Boston, 1982.
30. Gao J., et al., *Learning Lexicon Models from Search Logs for Query Expansion.* Conference on Empirical Methods in Natural Language Processing (EMNLP'12), 2012. pp. 666-676.
31. Graupmann J.; Schenkel R. and Weikum G., *The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents.* Proceedings of the International Conference on Very Large Databases (VLDB), 2005. pp. 529-540.
32. Guha S.; Jagadish H.V.; Koudas N.; Srivastava D.; and Yu T., *Approximate XML Joins.* Proceedings of ACM International Conference on Managemenet of Data (SIGMOD), 2002. pp. 287-298.
33. Haraty R., Dimishkieh M., and Masud M., *An Enhanced k-Means Clustering Algorithm for Pattern Discovery in Healthcare Data.* Intelligent Journal on Distributed Sensor Networks, 2015. 11: 615740:1-615740:11.
34. Haraty R. and Nasrallah R., *Indexing Arabic Texts using Association Rule Data Mining.* Library Hi Tech, 2019. 37(1): 101-117.
35. Harman D., *Towards Interactive Query Expansion.* SIGIR Forum 2017. 51(2): 79-89.
36. Helmer S., *Measuring the Structural Similarity of Semistructured Documents Using Entropy* Proceedings of the International Conference on Very Large Databases (VLDB), 2007. pp. 1022-1032.
37. Hoffart J., et al., *YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia.* Artif. Intell., 2013. 194: 28-61.

38. Holub M., et al., *Tailored Feature Extraction for Lexical Disambiguation of English Verbs Based on Corpus Pattern Analysis.* International Conference on Computational Linguistics (COLING'12), 2012. pp. 1195-1210.
39. Iranzo P. and Sáenz-Pérez F., *Implementing WordNet Measures of Lexical Semantic Similarity in a Fuzzy Logic Programming System.* Theory and Practice of Logic Programming, 2021. 21(2): 264-282.
40. Kamvar M. and Baluja S., *A Large Scale Study of Wireless Search Behavior: Google Mobile Search.* In Proceedings of the SIGCHI Conference on Computer Human Interaction, 2006. pp. 701–709, Montreal, Canada.
41. Kumar R., Guggilla B., and Pamula R., *Book search using social information, user profiles and query expansion with Pseudo Relevance Feedback.* Applied Intelligence, 2019. 49(6): 2178-2200.
42. Kwon S., Oh D., and Ko Y., *Word Sense Disambiguation based on Context Selection using Knowledge-based Word Similarity.* Information Processing and Management, 2021. 58(4): 102551.
43. Leacock C. and Chodorow M., *Combining Local Context and WordNet Similarity for Word Sense Identification.* FellBaum C. editor, WordNet: An Electronic Lexical Database, Chapter 11, The MIT Press, Cambridge, 1998. pp. 265-283.
44. Li Y.; Yang H. and Jagadish H.V., *NaLIX: an interactive natural language interface for querying XML.* Proceedings of the International ACM Conference on Management of Data (SIGMOD), 2005. pp. 900-902.
45. Li Y.; Yang H. and Jagadish H.V., *Term Disambiguation in Natural Language Query for XML.* In Proceedings of the International Conference on Flexible Query Answering Systems (FQAS), 2006. LNAI 4027, pp. 133–146.
46. Liang W.; and Yokota H., *LAX: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration.* Proceedings of the British National Conference on Databases (BNCOD), 2005. pp. 82-97.
47. Lin D., *An Information-Theoretic Definition of Similarity.* Proceedings of the International Conference on Machine Learning (ICML), 1998. pp. 296-304. Morgan Kaufmann Pub. Inc.
48. Lloyd S., *Least Squares quantization in PCM.* IEEE Transactions on Information Theory, 1982. 28(2):129-137.
49. Mandreoli F. and Martoglia R., *Knowledge-based sense disambiguation (almost) for all structures.* Information Systems, 2011. 36(2): 406-430.
50. Miller G., *WordNet: An On-Line Lexical Database.* International Journal of Lexicography, 1990. 3(4).
51. Miller G.A. and Fellbaum C., *WordNet Then and Now.* Language Resources and Evaluation, 2007. 41(2): 209-214.
52. Mohammad S., Hirst G., and Resnik P., *Tor, TorMd: Distributional Profiles of Concepts for Unsupervised Word Sense Disambiguation.* SemEval@ACL 2007, 2007. pp. 326-333.
53. Navigli R., *Word Sense Disambiguation: a Survey.* ACM Computing Surveys, 2009. 41(2):1–69.
54. Navigli R. and Velardi P., *Structural Semantic Interconnections: A knowledge-based Approach to Word Sense Disambiguation* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005. 27(7):1075–1086.
55. Navigli R. and Crisafulli G., *Inducing Word Senses to Improve Web Search Result Clustering.* In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, 2010. pp. 116–126, MIT, USA.
56. Navigli R. and Velardi P., *An Analysis of Ontology-based Query Expansion Strategies.* In proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI'03), 2003. pp. 42-49.
57. Nierman A. and Jagadish H. V., *Evaluating structural similarity in XML documents.* Proceedings of the ACM SIGMOD International Workshop on the Web and Databases (WebDB), 2002. pp. 61-66.

58. Qtaish A. and Alshammari M., *A Narrative Review of Storing and Querying XML Documents using Relational Database.* Journal of Information & Knowledge Management, 2019. 18(4): 1950048:1-1950048:28.
59. Rafiei D.; Moise D.; and Sun D., *Finding Syntactic Similarities between XML Documents.* Proceedings of the International Conference on Database and Expert Systems Applications (DEXA), 2006. pp. 512-516.
60. Resnik P., *Disambiguating Noun Groupings with Respect to WordNet Senses.* In Proceedings of the 3rd Workshop on Large Corpora, 1995. pp. 54-68.
61. Russell-Rose T., Gooch P., and Kruschwitz U., *Interactive Query Expansion for Professional Search Applications.* CoRR abs/2106.13528, 2021.
62. Salameh K., Tekli J., and Chbeir R., *SVG-to-RDF Image Semantization.* 7th International SISAP Conference, 2014. pp. 214-228.
63. Sanz I.; Mesiti M.; Guerrini G.; Berlanga La R.; and Berlanga Lavori R., *Approximate Subtree Identification in Heterogeneous XML Documents Collections.* XML Symposium, 2005. pp. 192-206.
64. Schlieder T., *Similarity Search in XML Data Using Cost-based Query Transformations.* Proceedings of the ACM SIGMOD International Workshop on the Web and Databases (WebDB), 2001. pp. 19-24.
65. Schlieder T. and Meuss H., *Querying and Ranking XML Documents.* Journal of the American Society for Information Science, Special Topic XML/IR, 2002. 53(6):489-503.
66. Singh S., Murthy H., and Gonsalves T., *Dynamic Query Expansion based on User's Real Time Implicit Feedback.* Conference on Knowledge Discovery and Information Retrieval (KDIR'10) 2010. pp. 112-121.
67. Soudani N., Bounhas I., and Ben Babis S., *Ambiguity Aware Arabic Document Indexing and Query Expansion: A Morphological Knowledge Learning-Based Approach.* The Florida AI Research Society Conference (FLAIRS'18 Conference), 2018. pp. 230-235.
68. Taddesse F.G., et al., *Semantic-based Merging of RSS Items.* World Wide Web Journal: Internet and Web Information Systems Journal Special Issue: Human-Centered Web Science., 2010. 13(1-2): 169-207, Springer Netherlands.
69. Taddesse F.G., et al., *Relating RSS News/Items.* Proceedings of the 9th International Conference on Web Engineering (ICWE'09), LNCS, 2009. pp. 44-452, San Sebastian, Spain.
70. Tagarelli A. and Greco S., *Semantic Clustering of XML Documents.* ACM Transactions on Information Systems, 2010. 28(1):3.
71. Tagarelli A.; Longo M. and Greco S., *Word Sense Disambiguation for XML Structure Feature Generation.* European Semantic Web Conference, 2009. LNCS 5554, pp. 143–157.
72. Taha K. and Elmasri R., *CXLEngine: A Comprehensive XML Loosely Structured Search Engine.* Proceedings of the EDBT workshop on Database Technologies for Handling XML Information on the Web (DataX'08), 2008. pp. 37-42, Nantes, France.
73. Taha K. and Elmasri R., *XCDSearch: An XML Context-Driven Search Engine.* IEEE Transactions on Knowledge and Data Engineering, 2010. 22(12):1781-1796.
74. Tannebaum W. and Rauber A., *Using Query Logs of USPTO Patent Examiners for Automatic Query Expansion in Patent Searching.* Information Retrieval, 2014. 17(5-6): 452-470.
75. Tekli J., *An Overview on XML Semantic Disambiguation from Unstructured Text to Semi-Structured Data: Background, Applications, and Ongoing Challenges.* IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), 2016. 28(6): 1383-1407.
76. Tekli J., et al., *Semantic to intelligent web era: building blocks, applications, and current trends.* . International Conference on Managment of Emergent Digital EcoSystems (MEDES), 2013. pp. 159-168.
77. Tekli J., et al., *(k, l)-Clustering for Transactional Data Streams Anonymization.* Information Security Practice and Experience, 2018. pp. 544-556.
78. Tekli J., Charbel N., and Chbeir R., *Building Semantic Trees from XML Documents.* Elsevier Journal of Web Semantics (JWS), 2016. 37–38:1–24.

79. Tekli J., et al., *SemIndex: Semantic-Aware Inverted Index.* Symposium on Advances in Databases and Information Systems (ADBIS), 2015. pp. 290-307.
80. Tekli J., et al., *SemIndex+: A Semantic Indexing Scheme for Structured, Unstructured, and Partly Structured Data.* Elsevier Knowledge-Based Systems, 2019. 164: 378-403.
81. Tekli J., et al., *Full-fledged Semantic Indexing and Querying Model Designed for Seamless Integration in Legacy RDBMS.* Data and Knowledge Engineering, 2018. 117: 133-173.
82. Tekli J., Chbeir R., and Yétongnon K., *A Fine-grained XML Structural Comparison Approach.* 26th International Conference on Conceptual Modeling (ER), 2007. LNCS 4801, pp. 582-598.
83. Tekli J., Chbeir R., and Yétongnon K., *Structural Similarity Evaluation between XML Documents and DTDs.* Proceedings of the 8th International Conference on Web Information Systems Engineering (WISE), 2007. pp. 196-211.
84. Tekli J., Tekli G., and Chbeir R., *Almost Linear Semantic XML Keyword Search.* Inter. ACM Conf. on Management of Emergent Digital EcoSystems (MEDES'21), 2021. pp. 129-138.
85. Theobald M.; Schenkel R. and Weikum G., *Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data.* In Proceedings of the ACM SIGMOD International Workshop on Databases (WebDB), 2003. pp. 1-6, San Diego, California.
86. Tu H., et al., *Word Sense Disambiguation Using Wikipedia Link Graph.* IEEE BigData 2019, 2019. pp. 6235-6236.
87. World Wide Web Consortium. *The Document Object Model*. http://www.w3.org/DOM, [Accessed Feb. 2022].
88. Wu Z. and Palmer M., *Verb Semantics and Lexical Selection.* Proceedings of the 32nd Annual Meeting of the Associations of Computational Linguistics, 1994. pp. 133-138.
89. Yang D., et al., *Query Intent Disambiguation of Keyword-Based Semantic Entity Search in Dataspaces.* Journal of Computer Science and Technology, 2013. 28:382–393.
90. Yaworsky D., *Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora.* Proceedings of the International Conference on Computational Linguistics (Coling), 1992. Vol 2, pp. 454-460. Nantes.
91. Yi J., Maghoul F., and Pedersen J., *Deciphering Mobile Search Patterns: a Study of Yahoo! Mobile Search Queries.* The Web Conference (WWW'08), 2008. pp. 257-266.
92. Zhang H. et al., *Learning from collective intelligence: Feature learning using social images and tags.* ACM transactions on multimedia computing, communications, and applications (TOMM), 2017. 13(1):1.
93. Zhang Z.; Li R.; Cao S.; and Zhu Y., *Similarity Metric in XML Documents.* Knowledge Management and Experience Management Workshop, 2003.

Joe Tekli is an Associate Professor in Computer Engineering in the Lebanese American University (LAU). He obtained his Ph.D. from the University of Bourgogne, LE2I-CNRS (France_2009). He completed various post-docs/research missions: University of Michigan (USA_2018), University of Pau (France 2017), University of Sao Paulo (Brazil_2011), University of Shizuoka (Japan_2010), University of Milan (Italy 2009). He was awarded various fellowships: Fulbright (USA 2018), FAPESP (Brazil 2011), JSPS (Japan 2010), Cariplo Foundation (Italy 2009), French Ministry of Education (France 2006-09), and AUF (France 2005). He has coordinated/participated in various projects: FAPESP (Brazil 2016-20), LAU-NCSR-L (Lebanon 2018-20), NCSR-L (Lebanon 2017-18), STICAmSud (France 2013-14), and CEDRE (France 2012-13). His research covers semi-structured, semantic, and multimedia data processing, and has more than 50 peer-reviewed publications. He is Vice Chair of ACM SIGAPP French Chapter (2018-) and founding member of UN-ESCWA Knowledge Hub

Gibert Tekli is an Associate Professor in the Mechatronics Engineering Technology Dept., the associate dean of the Issam Fares Faculty of Technology and an R&D engineering specialist in full stack agile cloud-based development, soft robotics and artificial intelligence. He holds a PhD in Computer Engineering from Telecom Saint Etienne, University of Lyon, France. He thrives on challenges rising from merging both worlds, Industrial R&D and Academia. He has successfully secured funds, lead, developed and consulted on international R&D projects (such as H2020 EU projects) while ensuring proper technology transfer from universities to the industry and vice versa.

Richard Chbeir received his PhD in Computer Science from the University of INSA-de-Lyon, France, in 2001. The author became a member of IEEE since 1999. He is currently a Full Professor in the Computer Science Department of the University of Pau and Pays de l'Adour (UPPA), Anglet, France. His is also Director of the UPPA Computer Science research laboratory (LIUPPA). His research interests are in the areas of distributed multimedia database management, XML similarity and rewriting, spatio-temporal applications, indexing methods, multimedia access control models, security and watermarking. He has published (more than 180 peer-reviewed publications) in international journals, books, and conferences, and has served on the program committees of several international conferences. He has been organizing many international conferences and workshops (ICDIM, CSTST, SITIS, MEDES, etc.). He is currently the Chair of the French Chapter ACM SIGAPP and the vice-chair of ACM SIGAPP.

Received: February 28, 2022; Accepted: August 22, 2022.