From Structure-based to Semantics-based: towards Effective XML Keyword Search

Thuy Ngoc Le [#], Huayu Wu [*], Tok Wang Ling [#], Luochen Li [#], and Jiaheng Lu [†]

[#] National University of Singapore {*ltngoc,lingtw,luochen*}*@comp.nus.edu.sg*
[*] Institute for Infocomm Research, Singapore *huwu@i2r.a-star.edu.sg*
[†] Renmin Univesity of China *jiahenglu@ruc.edu.cn*

Abstract. Existing XML keyword search approaches can be categorized into tree-based search and graph-based search. Both of them are structure-based search because they mainly rely on the exploration of the structural features of document. Those structure-based approaches cannot fully exploit hidden semantics in XML document. This causes serious problems in processing some class of keyword queries. In this paper, we thoroughly point out mismatches between answers returned by structure-based search and the expectations of common users. Through detailed analysis of these mismatches, we show the importance of semantics in XML keyword search and propose a semantics-based approach to process XML keyword queries. Particularly, we propose to use Object Relationship (OR) graph, which fully captures semantics of object, relationship and attribute, to represent XML document and we develop algorithms based on the OR graph to return more comprehensive answers. Experimental results show that our proposed semantics-based approach can resolve the problems of the structure-based search, and significantly improve both the effectiveness and efficiency.

Keywords: XML, keyword search, object, semantics

1 Introduction

Current approaches for XML keyword search are structure-based because they mainly rely on the exploration of the structure of XML data. They can be classified into the tree-based and the graph-based search. The tree-based search is used when an XML document is modeled as a tree, i.e. without ID References (IDREFs) such as [22, 17, 12, 23, 20], while the graph-based search is used for XML documents with IDREFs such as [7, 9, 3, 13, 10]. Due to the high dependence on hierarchical structure and unawareness of real semantics in XML data, these approaches suffer from several serious limitations as illustrated in the following example.

Consider an XML keyword query Q = {`Bill, John`} issued to the XML data in Fig. 1, in which the query keywords match first name of two students. Let us discuss answers for this query returned by the LCA-based (Lowest Common Ancestor) approach, a representative of the tree-based search. The LCA-based approach returns the document root as an answer for Q, which is intuitively meaningless for users. Suppose we could tell that two objects are the same if they belong to the same object class and have the same object identifier (ID) value. Then `Course_(11)` and `Course_(35)` refer to the same object `<Course:CS5201>` because they belong to the same object

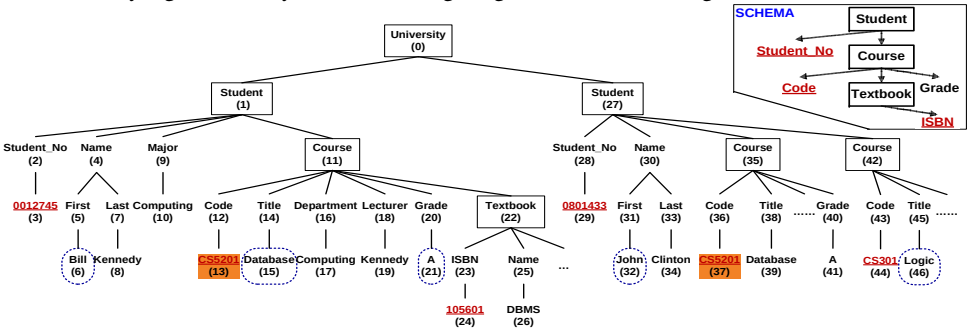


Fig. 1: university.xml

class `Course` and have the same object ID value `CS5201`. `<Course:CS5201>` is the common course taken by both students `Bill` and `John` and should be an answer. LCA-based approaches miss this answer because of unawareness of object, object ID and duplication of the same object. Thus, the common courses taken by both students are not recognized. In fact, other approaches of the tree-based and graph-based search suffer from similar problems, which will be demonstrated in Sec. 3 and 4.

The question we consider in this paper is that besides values, what benefits we can derive from paying attention to the role of tags in XML document, particularly for the problem of keyword queries. For this purpose, we introduce ORA-semantics and exploit it for XML keyword search. ORA-semantics stands for semantics of *Object*, *Relationship* and *Attribute* derived from XML tags. Once an XML document is defined with ORA-semantics, we can develop an effective semantics-based approach for XML keyword search which can solve limitations of the structure-based search.

In brief, the contributions of our work are as follows.

- We illustrate the limitations of both types of the structure-based search (the tree-based and the graph-based) in XML keyword search in details (Sec. 3 and 4).
- We introduce ORA-semantics and show that ORA-semantics plays an important role to effectively process XML keyword queries. Thus we propose semantics-based approach for XML keyword search, in which we use OR graph, which can capture full ORA-semantics, to represent XML document (Sec. 5 and 6).
- We perform comprehensive experiments to compare our semantic-based approach with the structure-based approaches (including XKSearch [22] and BLINK [7]). Experimental result shows the significant superiority of our methods because it can solve limitations of the structure-based search efficiently (Sec. 6).

2 Terminology

This section presents concepts of object, relationship and attribute used in this paper and uses the XML data in Fig. 1 for illustrating examples.

Concept 1 (Object) *In an XML data tree, an object is represented by a group of nodes, starting at a tag w.r.t. object class, followed by a set of attributes and their associated values. Each object belongs to an object class and has a unique object ID value, which can be single or composite.*

Concept 2 (Object node vs. non-object node) *Among nodes describe an object, the one w.r.t. an object class is called object node and all remaining nodes are called non-object nodes. Each non-object node is associated with a corresponding object node.*

For example, Student_(1) is an object node whereas Name_(4) is a non-object node belonging to object node Student_(1).

Concept 3 (The same object) *Two objects are the same if they belong to the same object class and have the same object ID value.*

For example, Course_(11) and Course_(35) refer to the same object Course: CS5201 because they belong to the same object class Course and they have the same object ID value CS5201. Object nodes which refer to the same object are usually identical. If we find two nodes that are not identical, they are still considered as referring to the same object as long as they are of the same object class and have the object ID.

Concept 4 (Relationship) *Objects may be connected through some relationship which can be explicit or implicit. An explicit relationship explicitly appears in an XML data as a node, whereas an implicit relationship is reflected by the connection among objects.*

For example, in Fig. 1, there is no explicit relationship but several implicit relationships such as relationships between Student_(1) and Course_(11).

Concept 5 (Attribute) *An attribute can be an object attribute or a relationship attribute. In XML data, it can be a child of an object node, a child of an explicit relationship node or a child of the lowest object of an implicit relationship node.*

For example, Lecturer is an attribute of object class Course whereas Grade is a relationship attribute of an implicit relationship between a Student and a Course.

3 Revisiting the tree-based XML keyword search

Since almost all tree-based approaches are based on LCA semantics such as SLCA [22], VLCA [12], ELCA [23], we use the LCA-based approach as a representative of the tree-based search onward. In this section, we systematically point out limitations of the LCA semantics by comparing answers returned by the LCA semantics and answers that are probably expected by users. We use the XML data in Fig. 1 for illustration. It is worthy to note that Course_(11) and Course_(35) refer to the same object <Course:CS5201> despite of appearing as different nodes.

3.1 Meaningless answer

Example 3.1. $Q_{3.1}$ = {Bill}.

LCA answer. The LCA-based approach returns node Bill_(6). However, this is not useful since it does not provide any supplementary information about Bill. This happens when a returned node is a non-object node, e.g., an attribute or a value.

Reason. The LCA-based approach cannot differentiate object and non-object nodes. Returning object node is meaningful whereas returning non-object node is not.

Expected answer. The expected answer should be forced up to Student_(1), the object w.r.t. to Bill_(6) since it contains supplementary information related to Bill.

3.2 Missing Answer

Example 3.2. $Q_{3.2}$ = {Bill, John}.

LCA answer. The LCA-based approach returns the document root which is definitely not meaningful.

Reason. Due to unawareness of semantics of object, the LCA-based approach can never recognize that, Course_(11) and Course_(35) refer to the same object Course CS5201. This is the common course taken by the two students Bill and John.

Expected answer. The expected answer should be the common course taken by these two students, i.e., <Course:CS5201> appearing as Course_(11) and Course_(35).

3.3 Duplicated answer

Example 3.3. $Q_{3.3}$ = {CS5201, Database}.
LCA answer. Two answers Course_(11) and Course_(35) of this query are duplicated because the two nodes refer to the same object <Course:CS5201>.
Reason. Similar to Example 3.1, this problem is caused by the unawareness of duplication of object having multiple occurrences.
Expected answer. Either of Course_(11) or Course_(35) should be returned, but not both since they are different occurrences of the same object <Course:CS5201>.

3.4 Problems related to relationships

Example 3.4. $Q_{3.4}$ = {Database, A}.
LCA answer. The LCA-based approach returns Course_(11) and Course_(35) as answers. These answers are incomplete because 'A' grade is not an attribute of a course, but it is grade of a student taking the course instead. On the other hand, Grade is a relationship attribute between Student and Course, not an object attribute.
Reason. The LCA-based approach cannot distinguish between an object attribute and a relationship attribute under an object node.
Expected answer. The proper answer should be all students taking course Database and getting an 'A' grade. To do that, the answer should be moved up to contain other objects (e.g., students) participating in the relationship that 'A' grade belongs to.

3.5 Schema dependence

There may be several designs for the same data source. The XML data in Fig. 1 can be represented by another design as in Fig. 2 with different hierarchical structure among object classes, e.g., Course becomes the parent of Student.
Example 3.5. $Q_{3.5}$ = {Bill, Database}.
LCA answer. With the design in Fig. 1, the LCA-based approach returns Student_(1). With the design in Fig. 2, Course_(1) is returned. As shown, answers for different designs are different though these designs refer to exactly the same information and we are dealing with the same query.

Answers of other queries related to more than one object also depend on XML hierarchical structure. Let us recall $Q_{3.2}$ = {Bill, John}, $Q_{3.4}$ = {Database, A} and discuss the answers from the design in Fig. 2 for these queries. For $Q_{3.2}$, Course_(1) is an answer. For $Q_{3.4}$, Course_(1) is also returned. Compared with the answers the root for $Q_{3.2}$ and Course_(11) and Course_(35) (without students as their children) for $Q_{3.4}$ from the design in Fig. 1, the ones from the design in Fig. 2 are different.
Reason. Answers from the LCA semantics rely on the hierarchical structure of XML data. Different hierarchical structures may provide different answers for the same query.
Expected answer. Users issue a keyword query without knowledge about the underlying structure of the data. Thus, their expectation about the answers is independent to the

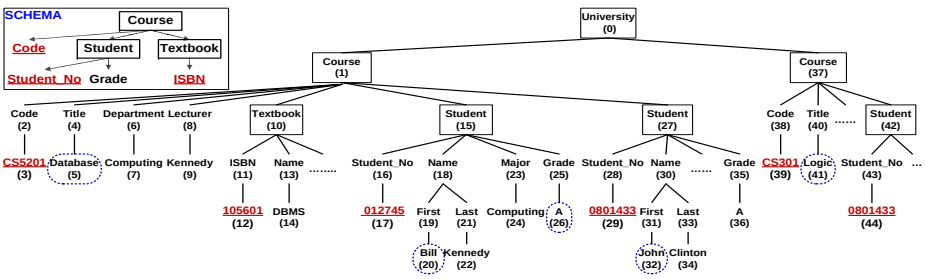


Fig. 2: Another design for the XML data in Fig. 1

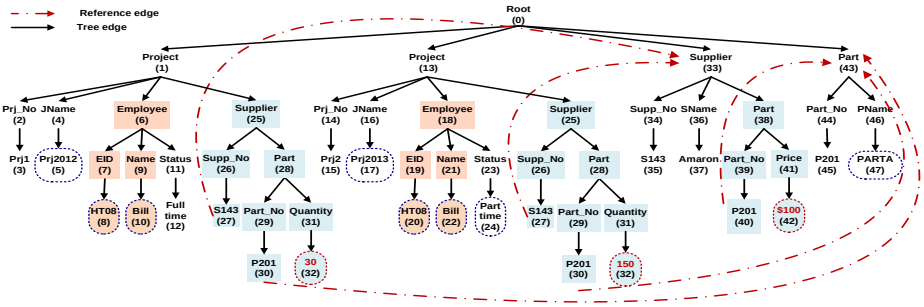


Fig. 3: An XML document with IDREFs

schema design. Therefore, the expected answers should also be semantically the same with all designs of the same data source.

Summary. The main reasons of the above problems are the high dependence of answers returned by the LCA-based search on the hierarchical structure of XML data (e.g., $Q_{3.5}$, $Q_{3.2}$, $Q_{3.4}$), and the unawareness of semantics of object, relationship and attribute. Particularly, unawareness of objects causes *missing answers* (e.g., $Q_{3.2}$), and *duplicated answer* (e.g., $Q_{3.3}$) because the LCA-based approach cannot discover the same object. Unawareness of object and attribute cause *meaningless answer* (e.g., $Q_{3.1}$) because it cannot differentiate XML elements. Unawareness of relationship and attribute cause the *problems related to relationship* (e.g., $Q_{3.4}$) because of it is unable to differentiate an object attribute and a relationship attribute.

4 Revisiting the graph-based XML keyword search

The graph-based search can be applied for both XML tree (without IDREF) and XML graph (with IDREFs). In the absence of IDREF, the graph-based search suffers from the same problems as the LCA-based search does. With IDREFs, object and object ID are observed and utilized. An object can be referenced by an IDREF, which has the same value with its object ID to avoid duplication. This helps the graph-based search handle some but not all problems of the LCA-based search. Particularity, the *problems related to relationship* and *meaningless answers* cannot be solved no mater IDREF is used or not. The other problems including *missing answer*, *duplicated answer* and *schema dependence* can be solved if the ID Reference mechanism applies to all objects. Otherwise (existing some objects without IDREF), they cannot be solved totally.

For generality, in this section, we use the XML data in Fig. 3 which contains both objects with and without IDREF to illustrate problems of the graph-based search. We apply the widely accepted semantics *minimum Steiner tree* [5, 6] for illustrating the problems. In the XML data in Fig. 3, Object `<Employee:HT08>` is duplicated with two occurrences `Employee_(6)` and `Employee_(26)`. Ternary relationship type among `Supplier`, `Project` and `Part` means suppliers supply parts to projects. `Quantity` is an attribute of this ternary relationship and represents the quantity of a part supplied to a project by a supplier. Besides, binary relationship between `Supplier` and `Part` has an attribute `Price` to represent the price of a part supplied by a supplier.

4.1 Problems cannot be solved with IDREF

IDREF mechanism is aware of semantics of object and object ID. However, the semantics of relationship and attribute is still not recognized and utilized which causes the problems of *meaningless answer*, and *problems related to relationship*.

Meaningless answer. Not differentiating object and non-object nodes cause meaningless answer when the returned node is a non-object node. For example, for $Q_{4.1}$ = {Amazon}, the answer is only Amazon_(45) without any other information.

Problems related to relationships. Without semantics of relationship, the graph-based search cannot distinguish object attribute and relationship attribute, and cannot recognize n-ary ($n \geq 3$) relationship. These cause problems related to relationship.

For example, for $Q_{4.2}$ = {PARTA, 100}, the subtree rooted at Part_(46) is an answer. However, this is not complete since price 100 is the price of a part named PARTA supplied by Supplier_(41). It is not the price of Part_(46). Thus, the answer should be moved up to Supplier_(41) to include Supplier_(41) as well.

For another example related to ternary relationship, $Q_{4.3}$ = {PARTA, 150}, the answer is the subtree rooted at Part_(36). This is not complete either since 150 is the quantity of a part named PARTA supplied by Supplier_(41) to Project_(21). Quantity is not an attribute of object Part_(46). Thus, the answer should be moved up to Project_(21) to include Project_(21) and Supplier_(41).

4.2 Problems can be solved with IDREF

IDREF mechanism is based on semantics of object and object ID, thus using IDREF can avoid problems caused by lack of semantics of object, including the problems of *missing answer*, *duplicated answer* and *schema dependence*. However, if ID Reference mechanism is not totally applied for all objects, i.e., there exists some objects without IDREF as object <Employee:HT08> in Fig. 3, then the above problems are not totally solved.

For example, $Q_{4.4}$ = {Bill, HT08} has two duplicated answers, Employee_(6) and Employee_(26). For $Q_{4.5}$ = {Prj2012, Prj2013}, only the subtree containing Supplier_(41) can be returned whereas the subtree containing <Employee:HT08> is *missed*. If object class Employee is designed as the parent of object class Project, the missing answer of $Q_{4.5}$ are found. It shows that the graph-based search also *depends on the design of XML schema* in this case.

Summary. The graph-based search can avoid *missing answer*, *duplicated answer* and *schema dependence* only if the ID reference completely covers all objects. Otherwise, the above limitations cannot avoid. The other problems including *meaningless answer* and *problems related to relationship* are still unsolved no matter IDREFs are used or not because IDREF mechanism only considers semantics of object and object ID but ignores semantics of relationship and attribute.

5 Impact of ORA-semantics in XML keyword search

We pointed out limitations of the structure-based search (the LCA-based and the graph-based approaches) because of unawareness of identification of object, relationship and attribute. We refer such identification as ORA-semantics. This section introduces ORA-semantics, shows the impact of ORA-semantics in XML keyword search and discusses the way to discover ORA-semantics from XML schema and data.

5.1 ORA-semantics

The term *semantics* has different interpretations. In this paper, we define the concept of ORA-semantics to include the identification related to *object*, *relationship* and *attribute*. At data level, an *object* represents a real world entity. Several objects may be

connected through some *relationship*. Objects and relationships may have a set of *attribute values* to describe their properties. Object, relationship and attribute value is an instance of *object class*, *relationship type* and *attribute* respectively at the schema level. Besides such major semantics, there are connecting nodes such as composite attributes or aggregation nodes. In brief, ORA-semantics is defined as follows.

Concept 6 (ORA-semantics (Object-Relationship-Attribute-semantics)) *In an XML schema tree, the ORA-semantics is the identification of object class, OID, object attribute, aggregation node, composite attribute and explicit/implicit relationship type with relationship attributes.*

For example, at schema level, ORA-semantics of the schema in Fig. 1 includes:
- `Student, Course` and `Textbook` are object class
- `Student_No, Code` and `ISBN` are object ID of the above object classes.
- `Grade` is the attribute of the relationship between `Student` and `Course`.
- For simplicity, we do not include object attribute in the schema in Fig. 1. The hidden object attributes include `Student_No, Name`, etc of object class `Student`.

At data level, ORA-semantics in the XML data in Fig. 1 includes that `Course_(11)` is object; `CS5201` is its object ID value; `Database, Computing` are its attribute values; especially `A` is an attribute value of the relationship it involves in, etc.

5.2 Impact of ORA-semantics in XML keyword search

Object semantics. Object identification helps detect multiple occurrences of the same object (duplicated object) appearing at different places in XML document. This enables us to filter *duplicated answers* and discover *missing answers*.

Relationship semantics. Relationship identification helps discover the degree of a relationship to return a more complete answer for queries involving in *ternary relationship*.

Attribute semantics. Differentiating object attribute and relationship attribute avoids returning incorrect answer for queries involving in *relationship attribute*. Moreover, differentiating object and non-object node also avoids *meaningless answers*.

ORA-semantics. Exploiting ORA-semantics in processing keyword query provides answers independent from the schema designs. In brief, ORA-semantics can resolve all problems of the structure-based search discussed in Sec. 3 and Sec. 4.

5.3 Discovering ORA-semantics

ORA-semantics includes the identification of internal nodes and leaf nodes in XML schema. Particularly, an internal node can be classified into object class, explicit relationship type, composite attribute and aggregation node. A leaf node can be a relationship attribute or an object attribute, which further can be object ID or a normal object attribute. We have designed algorithms to automatically discover ORA-semantics with high accuracy for overall process (higher than 94%). More details can be found in [14]. Other effective algorithms can be studied; however, this task is orthogonal to this paper.

6 Our semantics-based XML keyword search

Discussion in Sec. 4.1 shows that even if all objects follow IDREF mechanism and thus there is no duplication, the existing graph-based search still suffers from problems of *meaningless answer*, and especially *problems related to relationship*. In brief, IDREF cannot solve all problems of the existing structure-based search. To solve these problems, a more semantics-enriched model is needed, in which not only objects, but relationships and their attributes must be fully captured as well. We illustrate this model

by proposing Object Relationship (OR) graph to represent an XML document. Both objects and relationships are represented as nodes in an OR graph while attributes and values are associated with the corresponding objects and relationships. As such, an OR graph can capture all objects, relationships and attributes of an XML data.

In this section, we first introduce the OR graph, its advantages, and the process of OR graph generation. We then present search semantics, i.e., formally define the expected answers, and query processing based on the OR graph. To show the advantages of the OR graph, we compare our OR graph based search with the structure-based search. To improve efficiency, we propose indexes and optimized search algorithms.

6.1 Object relationship (OR) graph

Definition 1 (OR graph) *An OR graph* $G = (V_O, V_R, E)$ *is an unweighed, undirected* ***bipartite*** *graph with two types of nodes, where* V_O *is the set of* object nodes*,* V_R *is the set of* relationship nodes*,* $V_O \cap V_R = \emptyset$*, and* E *is the set of* edges*. An edge is between an object node in* V_O *and a relationship node in* V_R*. For every relationship* $r \in V_R$*, there is an edge between each of its participating objects and* r*.*

The information stored for each object node is a quadruple ⟨*nodeID, object class, OID, associated keywords*⟩. The information stored for each relationship node is a similar quadruple ⟨*nodeID, relationship type,* {*participating* o_i}*, associated keywords*⟩. Each object/relationship has a randomly generated *nodeID* as label in the OR graph. *Associated keywords* of an object/relationship include its attributes and values as well.

Consider the XML data in Fig. 3. The OR Graph conforming to this XML data is shown in Fig.4, where *square* and *diamond* stand for *object node* and *relationship node* respectively. Each object/relationship node in an OR graph has an identifier, e.g., o_1, o_2, o_3, o_4 and o_5 are objects and r_1, r_2, r_3, r_4 and r_5 are relationships. Additionally, to serve for a clearer explanation, for each object node in Fig.4, we associate its object class and object ID (e.g., `Project Prj1` for o_1). Moreover, we also show matching nodes of query keywords discussed in Sec. 4 (e.g., `Amazon` matches o_4).

6.2 Features and advantages of the OR graph

FA1. Object-relationship level. Our OR graph represents XML document at object-relationship level by associating attributes and values with their corresponding objects and relationships. This can significantly *reduces the search space* since the number of nodes of an OR graph is much less than that of the corresponding XML data due to not counting attributes, values and duplicated objects. Moreover, it can avoid *meaningless answers* because an answer must correspond to a whole object or relationship rather than an arbitrary XML element.

FA2. Duplicate-free. An object may appear as multiple occurrences in an XML document. In an OR graph, each object node represents an *object*, not an occurrence of

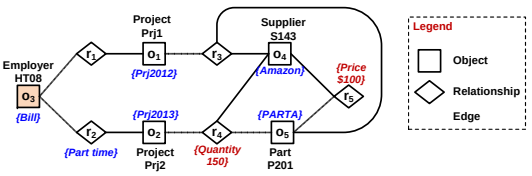


Fig. 4: The OR graph w.r.t. the XML data in Fig. 3

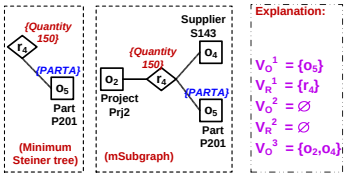


Fig. 5: $mSubgraph$ for $Q_{4.3}$

object. Thus, an object is not duplicated because it corresponds to only one object node. Duplicate-free is a very important feature of OR graph, by which the process can find *missing answers* and avoid *duplicated answers* from the LCA-based approach.

FA3. Explicit appearance of relationships. Differentiating object nodes and relationship nodes enables us to distinguish between object attribute and relationship attribute. This avoids returning incorrect answers for queries involving relationship attribute. Moreover, relationship nodes can provide the degree (e.g., binary or ternary) of a relationship. As a result, we can add *all participating object nodes* of relationship nodes in a returned answer to make it more meaningful.

FA4. Schema-independence. The OR graphs conforming to any XML representations of the same data source are the same no matter which schema design is used. In other words, the OR graph is *independent* of the XML structure. This enables us to return the same answers for all designs of the same data source. For example, no matter `<Paper:ER32>` in the data in Fig. 3 is designed with or without IDREF, the OR graphs conforming is in Fig. 4.

6.3 Generating OR graph from an XML document

Algorithm 1: OR graph generation

Input: D: XML document
Output: OR graph $G(V_O, V_R, E)$

```
1  V_O ← ∅
2  V_R ← ∅
3  E ← ∅
4  for each object o visited by document order do
5      if o is not duplicated with objects in V_O then
6          V_O.Add(o)
7      Rel(o) ← relationships among o and its parents
8      for each relationship r in Rel(o) do
9          if r is not duplicated with relationships in V_R then
10             V_R.Add(r)
11             e ← edge between o and r
12             E.Add(e)
```

To generate OR graph $G(V_O, V_R, E)$ from an XML document D (with or without IDREFs), we identify V_O, V_R, E from nodes in D. Only distinct objects in D are added to V_O. The process of OR graph generation is presented in Algorithm 1. We traverse XML document by document order. For each visited object node o, we add o to V_O if o is not duplicated with any existing object in V_O and find the the set of relationships $Rel(o)$ among o and its parents. For each relationship r in $Rel(o)$, we add r to V_R if r is new. We finally add the edge between o and r to E. In brief, the sequence is adding an object, then a relationship it participates in, and finally the edge between them [11].

6.4 Search semantics

This paper adopts the *minimum Steiner tree semantics* [5, 6] because it is the widely accepted semantics for the graph-based search. A Steiner tree is a subtree of the data graph that contains all keywords. The weight of a Steiner tree is defined as the total weight of its edges. A *minimum* Steiner tree is a Steiner tree having the smallest weight among all the Steiner trees w.r.t. the same set of matching nodes. However, a subtree returned by this semantics may not contain completely meaningful information to answer a query, especially when XML data has complicated structure. Therefore, we determine what other nodes should be added in order to return a more meaningful answer to users. Consequently, we extend a minimum Steiner tree s to become a more meaningful subgraph (called $mSubgraph$) by *adding all participating objects* of all relationships in s. The rationale is that a relationship itself has no or incomplete meaning without all of its participating objects. An answer $mSubgraph$, is defined as follows.

Definition 2 (mSubgraph) *Given a keyword query Q to the OR graph G = (V_O,V_R,E). An answer mSubgraph of Q is a subgraph of G and is denoted as ${}_mS({}_mV_O,\ {}_mV_R,\ {}_mE)$, where ${}_mV_O = V_O^1 \cup V_O^2 \cup V_O^3$ and ${}_mV_R = V_R^1 \cup V_R^2$ such that*

- *V_O^1 and V_R^1 are the sets of matching objects and relationships.*
- *V_O^2 and V_R^2 are the sets of intermediate objects and relationships connecting objects in V_O^1 and relationships in V_R^1.*
- *V_O^3 is the set of added objects which participate in ${}_mV_R$ but are not in $V_O^1 \cup V_O^2$.*
- *${}_mE$ is the set of edges between each object in ${}_mV_O$ to its relationships in ${}_mV_R$.*

Let us recall $Q_{4.3}$ = {PARTA, 150} to illustrate the benefits of $mSubgraph$. Answers for this query (both in form of a minimum Steiner tree and in form of an $mSubgraph$) are shown in Fig. 5. As can be seen, $mSubgraphs$ are more meaningful than minimum Steiner trees with intuitive meaning of 150 parts name PARTA are supplied by <Supplier:S143> to <Project:Prj1> while the minimum Steiner tree does not provide information about the supplier and the project.

6.5 Comparison on the LCA, graph and semantics based approaches

We compare the limitations of the LCA-based, the graph-based and our OR graph based search and show the reasons behind in Table 1.

Table 1: Comparison on the LCA-based, graph-based and OR graph based approaches

Problem	LCA-based	Graph-based	Reason of the problems of the structure-based search	Semantics needed	OR graph based	Reason that the semantic based search can avoid problems
Meaningless answer	Yes	Yes	Returning non-object element because of not differentiating object and non-object node	-Object -Attribute	No	All attributes and values are associated with objects/relationships (Object-relationship level, FA1)
Missing answer	Yes	Partial	- Unable to discover the same object appearing at different places in XML document	Object	No	The same objects can be discovered (Duplicate-free, FA2 of OR graph)
Duplicated answer	Yes	Partial	- Partial because it can be solved with IDREF			
Problems related to relationships	Yes	Yes	- Unable to distinguish relationship attribute and object attribute - Not aware ternary relationships and above	-Attribute -Relationship	No	Relationships and relationship attributes are discovered (Explicit appearance of relationships, FA3 of OR graph)
Schema dependence	Yes	Partial	- Relying on hierarchy (LCA) - Partial because it can be solved with IDREF	Object	No	Different designs of the same data have the same ORA-semantics (Schema-independence, FA4)

6.6 Query processing

This work aims to show the impact of ORA-semantics on effectiveness of XML keyword search. The ORA-semantics is exploited in generating the OR graph. After the generation process, searching over OR graph to find minimum Steiner trees can be implemented by existing efficient algorithms. Answers then will be extended to $mSubgraphs$. Space limitation precludes detailed discussion about their algorithms. However, readers may find them in [5, 6]. We also design our own algorithm and propose index and an optimized techniques to improve the efficiency in Sec. 6.7. We work at object level by assigning nodes describing an object instance the same label. Thereby, we dramatically reduce the search space and greatly improve the efficiency.

6.7 Index and optimization

Indexes. To make distinction between query keywords with document keywords, we call the latter as *term*. To support our optimized algorithm, we propose three indexes: semantic-inverted lists, term-node lists and node-node lists. Due to space constraint, we very briefly describe these indexes. Details are given in our technical report [11].

Semantic-inverted list is to store the set of matching objects and the set of matching relationships of a term. Term-node lists is to store the shortest distance from some node to $mNode(t)$ where $mNode(t)$ is the set of nodes matching term t. Node-node list is to store a set of shortest distances from some node to each node in $mNode(t)$.

The optimized algorithm. The search has two phases. To find a good Steiner tree, we first find a good intermediate node v in sense that it can connect to all keywords with short distances. From v, we then generate the Steiner tree by look up the closest matching node for each keyword. The three indexes enable us to propose very efficient algorithms for the two-phase search. Details of explanations are given in [11].

Algorithm 2: Optimized Algo.

Input: Query $Q = \{k_1, k_2, \ldots, k_n\}$
Term-node lists L_{TN}
Output: top-k answers in $Ans(Q)$
Variables: T_{prune}: pruning threshold
$T_{prune} \leftarrow 0$
for $i \in [1, n]$ **do**
  $c_i \leftarrow$ new Cursor($L_{TN}(k_i)$,0)
while
$\exists j \in [1, n] : c_j.Next() \neq NULL$ **do**
  $i \leftarrow$ pick from [1,n] by BF order;
  $< v > \leftarrow c_i.Next()$
  if $v \neq NULL$ **then**
    KeywordNavigation(v);
  if $c_i.Dist() > T_{prune}$ *and* $T_{prune} \geq 0$ **then**
    Output top-k answers in $Ans(Q)$. Exit

Algorithm 3: KeywordNavigation(v)

Input: a connecting node v
Node-node lists L_{NN}
Variables: $cNode(Q)$: connecting nodes visited, initially $\emptyset$
if $cNode(Q)$.*contain*(v) **then**
  return
$cNode(Q)$.add(v)
while *TRUE* **do**
  for $j \in [1..n]$ **do**
    $< u_j > \leftarrow$ pick from $L_{NN}(v)$ in *BestFS* order;
  $g \leftarrow \langle \{u_1, u_2, \ldots, u_n, \}, V_I, E \rangle$
  if $g \notin Ans(Q)$ **then**
    if *size*(g) $> T_{prune}$ *and* $T_{prune} \geq 0$ **then**
      return;
    $Ans(Q)$.add(g)// `size in ascending order`
    if $|Ans(Q)| \geq k$ **then**
      $T_{prune} \leftarrow$ the k^{th} biggest of $\{size(\alpha) | \alpha \in Ans(Q)\}$

7 EXPERIMENTS

We compare the quality of answers returned by our OR graph based search with XK-Search [22] and BLINK [7] (two state-of-the-art algorithms to represent the LCA-based and graph-based search respectively). Using these two algorithms already show the improvement of our semantic-based search over the structure-based search because other structure-based approaches suffer from the same problems with them. The experiments were performed on a Intel(R) Core(TM)2 Duo CPU 2.33GHz with 3.25GB of RAM with three real data sets: `eBay`[1] (0.36MB), `NBA`[2](45.2MB), `Baseball`[3] (60MB).

7.1 Rating answers

We compare the quality of the answers returned by our approaches and the structure-based approaches by rating their answers. We randomly generated 25 queries from all document keywords. After filtering out some meaningless queries, we chose 10 queries as shown in Table. 6. We asked 21 students major in computer science to rate the top-10 answers on scale [0-5] (0 for totally mismatch and 5 for perfectly match the user expectation). The scores of answers are shown in Fig. 7.

Discussion. As shown, our approach gets the highest scores for all queries, which means that our approach returns more meaningful answers to users. Moreover, the scores of

[1] www.cs.washington.edu/research/xmldatasets/data/auctions/ebay.xml

[2] http://www.databasebasketball.com/

[3] http://www.seanlahman.com/baseball-archive/statistics/

Query	Query Keywords	Dataset
Q1	**wizbang4**	eBay
Q2	**Bill**	NBA
Q3	**id_num, 511364992**	eBay
Q4	**ct-inc, CyberTech**	eBay
Q5	**Michael, Coach**	NBA
Q6	**Player, Bill, Sam**	NBA
Q7	**pizarju01, teams**	Baseball
Q8	**BOS, TOR, vioxji01**	Baseball
Q9	**Celtics, player**	NBA
Q10	**Michael, Celtics, team**	NBA

Fig. 6: 10 keyword queries for users to rate

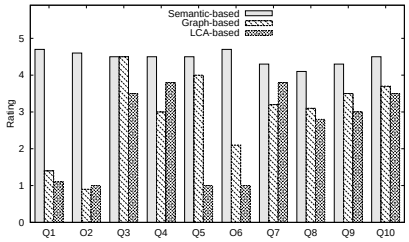
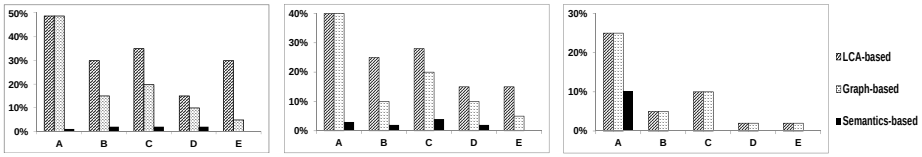


Fig. 7: Scores of answers

our approach are very high (above 4) and stable, which infers that users are satisfied with our answers. In contrast, the structure-based search gets lower scores since its answers mismatch the expectations of users. Especially for queries Q_1, Q_2 and Q_6, the scores are around 1 because they returns hundreds of duplicated attribute nodes without any detailed information. Generally, the graph-based search has higher score than the LCA-based search because it can find missing answers and avoid duplicated answers when IDREFs are considered.

7.2 Statistics on problems of answers

We collected 86 keyword queries for the above three data sets: 12 queries for eBay, 44 queries for NBA and 30 queries for Baseball from 21 students working in computer science. Based on the discussion in Sec. 3 and Sec. 4, the answers which cannot match the users' search intention are classified into five categories: including *(A) Meaningless answer; (B) Missing answer; (C) Duplicate answer; (D) Relationship problem; and (E) Schema dependence*. Fig. 8 shows the percentage of answers containing a certain problem over total answers. Note that an answer may contain more than one problem.



(a) Baseball (b) NBA (c) eBay

Fig. 8: Statistics on problems of answers: *(A) Meaningless answer; (B) Missing answer; (C) Duplicate answer; (D) Relationship problem; (E) Schema dependence*

Discussion. Generally, answers of our approach have fewer problems than those of the structure-based search. The most frequent problem is the meaningless answer. It usually occurs when a query contains only one keyword and the structure-based search returns only one non-object node matching that keyword. Sometimes our approach returns a meaningless answer when it discovers a composite attribute as an object. Missing and duplicated answers are also frequent problems because the structure-based search cannot discover the same object appearing in multiple nodes.

7.3 Efficiency

We evaluate the efficiency of the three compared works by varying the number of query keywords and node frequency of keyword using NBA dataset. The response time is shown in Fig. 9.

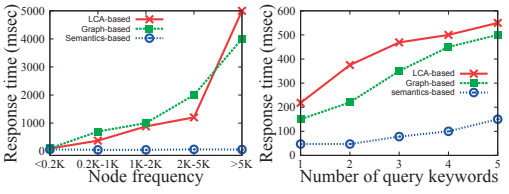


Fig. 9: Response time

Discussion. The response time of our approach depends on the frequency of matching objects and relationships rather than the node frequency since it works at object and relationship level. Thus, our approach runs stably while the others' response time increases very fast with the node frequency. Moreover, working at object and relationship level largely reduces the search space and thus enables our approach to run much faster than the others. More explanations and experiments on efficiency are given in [11].

8 RELATED WORK

Tree-based XML keyword search. Most existing tree-based XML keyword search methods are LCA-based and depend on hierachical structure of the data. XKSearch [22] defines Smallest LCAs (SLCAs) to be the LCAs that do not contain other LCAs. Meaningful LCA (MLCA) [15] incorporates SLCA into XQuery. VLCA and ELCA [12, 23] introduce the concept of valuable/ exclusive LCA to improve the effectiveness of SLCA. MESSIAH [20] handles cases where there are missing values in optional attributes. Although researchers have put efforts on improving LCA-based effectiveness, their works are still based on the hierarchical structure without ORA-semantics. Thus, they face problems as studied in Sec. 3.

Graph-based XML keyword search. Minimum Steiner tree and distinct root semantics [3, 7, 9] are popular but may return answers whose content nodes are not closely related. Recently, subgraph semantics [13, 19] and content based semantics [10, 16] is proposed to handle the above problem, but they still do not consider ORA-semantics. Thus, they face problems as discussed in Sec. 4.

Semantics-based XML keyword search. XSEarch [4] focuses on adding semantics into query but the added semantics is for distinguishing a tag name and a value keyword only. XSeek [17] and MaxMatch [18] infer semantics from keyword query. They can only infer semantics of object since it is impossible to infer any semantics of object ID, relationship and relationship attribute from a keyword query. XKeyword [8] exploits semantics from the XML schema. XReal [1], Bao et. al. [2] and Wu et. al. [21] proposed an object-level for XML keyword search. However, all of these works only consider objects, but they do not have the concepts of object ID, relationship and attribute. Therefore, they can avoid at most the problem of meaningless answer but still suffer from all other problems discussed in Sec. 3 and 4.

9 Conclusion and future work

We have systematically illustrated limitations of the existing LCA-based search, including the problems of *meaningless answer, missing answer, duplicate answer, problems related to relationship*, and *schema dependence* which are caused by unawareness of semantics of object, relationship and attribute. We have also demonstrated that even with IDREFs, the graph-based search can avoid at most the problems of *missing answer*, *duplicate answer* and *schema dependence* because IDREF mechanism is aware of only semantics of object and object ID but not semantics of relationship and attribute. Thus, the graph-based search still suffers from the problems of *meaningless answer* and *problems related to relationship*. We introduced *ORA-semantics* which is semantics of object, relationship and attribute and showed its importance in XML keyword search. To take ORA-semantics into account for XML keyword search, we propose Object Relationship (OR) graph to represent XML data, in which objects and relationships correspond to nodes, while attributes and values are associated with the corresponding object/relationship nodes. As such, OR graph can capture all ORA-semantics of an

XML data. To process a query, we search over the OR graph. Our index and optimization provide an efficient search algorithm. Experimental results showed that our OR graph based approach outperforms the structure-based search in term of both effectiveness and efficiency. Thus, semantics-based approach could be a promising direction for XML keyword search in solving problems of the current structure-based search.

References

1. Z. Bao, T. W. Ling, B. Chen, and J. Lu. Efficient XML keyword search with relevance oriented ranking. In *ICDE*, 2009.
2. Z. Bao, J. Lu, T. W. Ling, L. Xu, and H. Wu. An effective object-level XML keyword search. In *DASFAA*, 2010.
3. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, 2002.
4. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A semantic search engine for XML. In *VLDB*, 2003.
5. B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. Finding top-k min-cost connected trees in database. In *ICDE*, 2007.
6. K. Golenberg, B. Kimelfeld, and Y. Sagiv. Keyword proximity search in complex data graphs. In *SIGMOD*, 2008.
7. H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: ranked keyword searches on graphs. In *SIGMOD*, 2007.
8. V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword proximity search on XML graphs. In *ICDE*, 2003.
9. V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, and R. D. Hrishikesh Karambelkar. Bidirectional expansion for keyword search on graph databases. In *VLDB*, 2005.
10. M. Kargar and A. An. Keyword search in graphs: finding r-cliques. *PVLDB*, 2011.
11. T. N. Le, H. Wu, T. W. Ling, L. Li, and J. Lu. From structure-based to semantics-based: Effective XML keyword search. *TRB4/13, 2013, School of Computing, NUS*.
12. G. Li, J. Feng, J. Wang, and L. Zhou. Effective keyword search for valuable lcas over xml documents. In *CIKM*, pages 31–40, 2007.
13. G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. EASE: Efficient and adaptive keyword search on unstructured, semi-structured and structured data. In *SIGMOD*, 2008.
14. L. Li, T. N. Le, H. Wu, T. W. Ling, and S. Bressan. Discovering semantics from data-centric XML. *DEXA*, 2013.
15. Y. Li, C. Yu, and H. V. Jagadish. Schema-free XQuery. In *VLDB*, 2004.
16. X. Liu, C. Wan, and L. Chen. Returning clustered results for keyword search on xml documents. *TKDE*, 2011.
17. Z. Liu and Y. Chen. Identifying meaningful return information for XML keyword search. In *SIGMOD*, 2007.
18. Z. Liu and Y. Chen. Reasoning and identifying relevant matches for XML keyword search. In *PVLDB*, 2008.
19. L. Qin, J. X. Yu, L. Chang, and Y. Tao. Querying communities in relational databases. In *ICDE*, 2009.
20. B. Q. Truong, S. S. Bhowmick, C. E. Dyreson, and A. Sun. MESSIAH: missing element-conscious slca nodes search in xml data. In *SIGMOD*, 2013.
21. H. Wu and Z. Bao. Object-oriented XML keyword search. In *ER*, 2011.
22. Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In *SIGMOD*, 2005.
23. R. Zhou, C. Liu, and J. Li. Fast ELCA computation for keyword queries on XML data. In *EDBT*, 2010.