Exploit Keyword Query Semantics and Structure of Data for Effective XML Keyword Search

Khanh Nguyen  Jinli Cao
Department of Computer Science and Computer Engineering
La Trobe University, Melbourne Australia
Email: {tuan.nguyen, j.cao}@latrobe.edu.au

Abstract

Keyword search is a natural and user-friendly mechanism for querying XML data in information systems and Web based applications. One of the key tasks is to identify and return meaningful fragments as results, due to the limited expressiveness and the ambiguity of keyword queries. In this paper, we first studied query keyword patterns in order to exploit the user’s search intention behind the input keywords. The outcome of this task is that keywords in the query are classified as required information and search conditions (or predicates). In addition, unlike previous work that our work only returns desired fragments as results. Each returned result must satisfy the search conditions rather than simply contain all query keywords. To further prune irrelevant fragments we introduce a novel notion called Relevant Lowest Common Ancestor (RLCA) which effectively and precisely captures the meaningful and relevant fragments to the given keyword query. We conducted extensive experimental studies to prove the effectiveness of our approach.

Keywords: XML, Keyword Search.

1 Introduction

With XML rapidly emerging as a standard for representing, publishing and exchanging data over the Internet, it is demanding for adapting keyword search to XML data. This is because keyword search provides a friendly mechanism to access XML data without the knowledge of the structured query languages and the underlying complex data schema. The structured query languages (e.g. XPath and XQuery) convey obvious semantic meanings, thus they can retrieve precise results. In contrast, keyword search may not be able to return precise results as users expected due to the limited expressiveness and the ambiguity of keyword queries. Thus, identifying meaningful and relevant results becomes a very challenging and key task of XML keyword search.

Several attempts have been made toward identifying meaningful and relevant fragments to XML keyword search (Guo et al. 2003, Xu & Papakonstantinou 2005, Cohen et al. 2003, Li et al. 2007). Based on the notion of Lowest Common Ancestor (LCA) from graph theory (Harel & Tarjan 1984, Schieber & Vishkin 1988, Bender et al. 2005) and Smallest LCA (SLCA), we can classify previous attempts into LCA-based approaches (Guo et al. 2003, Cohen et al. 2003, Li et al. 2007) and SLCA-based approaches (Xu & Papakonstantinou 2005). Given an XML tree T and a keyword query, the semantics of LCA is to return a node n in T which the subtree rooted at n contains all query keywords, excluding other LCA nodes and their keyword nodes which are descendants of n (Guo et al. 2003). As for SLCA nodes, they are subsets of LCA nodes, however they have no descendants which also contain all keywords of the query (Xu & Papakonstantinou 2005).

A well known problem of the SLCA-based approaches is that they may miss relevant results. For example, when querying Q1 on document D1 (refer to Table 1 and Figure 1), the SLCA-based approaches consider the fragment rooted at article [0.2.0.1] (Figure. 3b) as an irrelevant result. This is because this fragment contains another fragment (Figure. 3a) which also consists of all query keywords. However, we can see that this fragment is definitely relevant to the query Q1. In contrast, LCA approaches (Xu & Papakonstantinou 2005) return all fragments consisting of query keywords that may include irrelevant results. For example, suppose that Q3 issued on the document D2, the nodes name [0.1.0.0] and nationality [0.1.1.1] are two matches to keywords “Gasol” and “USA” respectively and their LCA is [0.1]. However, this subtree (Figure. 4a) is not a desired result because name [0.1.0.0] and nationality [0.1.1.1] do not belong to the same player.

More recently, XSEarch (Cohen et al. 2003), CVLCA (Li et al. 2007) and MLCA (Li, Yu & Jagadish 2008) have been proposed to improve the effectiveness of the keyword search. Rather than simply return all fragments rooted at LCA or SLCA nodes, the interconnection between nodes in the fragment is used to prune undesired fragments. To fulfill this aim, node labels and structures of XML data are two main sources of considerations. However, these approaches have the following limitations:

1. They fail to identify the meaningless relationship between nodes with different labels but having the same type.
2. The fail to recognize the meaningful relationship between nodes with same label.
3. The semantics of keyword query has not been utilized to prune irrelevant results, especially in the presence of keyword ambiguity.

The detailed discussion and analysis of these problems will be presented in Section 2.

This paper introduces a novel approach called Relevant LCA (RLCA) to effectively and accurately capture relevant fragments to XML keyword search. RLCA is a LCA-based approach, but it is different from previous approaches which mainly focus on the evaluating node labels or the structure of a fragment.
Proposed a novel notion called Relevant LCA (RLCA) to capture the semantics of LCA nodes which the fragments rooted at those nodes are meaningful and relevant to the keyword query.

Exploited the both structure of data and the semantics of keyword queries to effectively identify meaningful fragments for XML keyword search.

Addressed the problem of keyword ambiguity by exploiting the search conditions from the query.

Implemented the algorithms and conducted extensive experiments on real data sets to show that the proposed RLCA outperforms existing approaches in term of effectiveness and precision while maintaining the efficiency.

The remainder of this paper is organized as follows. Next section discusses and analyzes the weaknesses of existing solutions and inspires our motivation. Section 3 introduces the notion of RLCA and its semantics. In Section 4, we present the algorithm of RLCA. Extensive experimental evaluations are provided in Section 5. Section 6 discusses several orthogonal studies could be incorporated into our work. Finally, the conclusions of the paper are presented in Section 7.

2 Preliminary and Motivation

2.1 Data Model and Query

Each XML document is modeled as a rooted and labeled tree. Every internal node in the tree has a name which is either an element name or a tag name from the XML document. Each leaf node in the XML tree has got a data value. We call internal nodes and leaf nodes as structure nodes and value nodes respectively. Each structure node is encoded by Dewey code as its ID.

A user query is expressed as a set of keywords which may match the labels or values of the nodes in XML trees. Without loss of generality, the 'AND' semantics is default in query results which is similar to the existing approaches (Guo et al. 2003, Xu & Papakonstantinou 2005, Li, Yu & Jagadish 2008, Li et al. 2007, Liu & Chen 2008). A keyword search on an XML data returns a set of meaningful subtrees. Conditions for a fragment to be meaningful will be presented in next Section.

2.2 Analysis on Existing Approaches and Motivations

Recently, several attempts such as XSEarch (Cohen et al. 2003), CVLCA (Li et al. 2007) and MLCA (Li, Yu & Jagadish 2008) have been made toward improving the meaningfulness of return fragments for XML keyword search. Specifically, in XSEarch, two nodes $u$ and $v$ are meaningfully related if the shortest path between $u$ and $v$ does not have two distinct nodes with same label, except $u$ and $v$. A fragment $F$ is meaningful if it satisfies two conditions: (i) $F$ contains the matches to all search terms and (ii) every two matches in $F$ has to be meaningfully related (pair semantics) or there exists a node in $F$ such that every match in $F$ has to be meaningfully related to it (star-semantics). In (Li et al. 2007), an approach called Compact Valuable LCA (CVLCA) has been introduced to answer keyword search. A node $u$ is called VLCA if the fragment $F$ rooted at $u$ satisfies star-semantics. A node $u$ is a VLCA node if it is a VLCA node and $u$ dominates every node in $F$, where $u$ dominates a node $v$ in $F$ if there does not exist another LCA node $u'$ which is a descendant of $u$ and the fragment rooted at $u'$ contains $v$.

Let us consider query $Q_1 = \{\text{Gasol, USA}\}$. The aforementioned algorithms will identify the fragment in Figure 4a meaningless to $Q_1$ because there exist two nodes $[0.1.0]$ and $[0.1.1]$, and both have the same label player on the shortest path between the matching nodes name $[0.1.0.0]$ and nationality $[0.1.1.0]$. On the other hand, these approaches may fail to identify the meaningless relationships between nodes with different labels but having the same type since they use node labels as the identifiers. Moreover, they also may fail to identify the meaningful relationship between nodes with the same labels. Therefore, they may fail to identify irrelevant fragments and also may miss relevant results.

Consider query $Q_4 = \{\text{Grizzlies, Gasol, Brown, position}\}$ and sample document $D_2$, both XSEarch and CVLCA fail to identify Figure 4b as an irrelevant fragment to query $Q_4$ because there do not exist two nodes with the same label on the path between matching nodes. Now, we consider query $Q_5 = \{\text{Grizzlies, Gasol, Brown, position}\}$ on XML document $D_2$. A subtree contains the nodes which satisfy all search terms depicted in Figure 4c. However, the approaches XSEarch and CVLCA will exclude this subtree from the meaningful results of $Q_5$ because there are two nodes $[0.1.0]$ and $[0.1.1]$ with the same label player on the shortest path between two matches name $[0.1.0.0]$ and nationality $[0.1.1.0]$. Actually, we can see that this fragment is absolutely relevant to $Q_5$.

To tackle with aforementioned problems, (Li, Yu & Jagadish 2008) proposed an approach called MLCA which prunes irrelevant subtrees based on the structure, rather than using node labels as XSEarch and

<table>
<thead>
<tr>
<th>$Q_i$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_1$</td>
<td>Ziyang Liu, XML Keyword Search</td>
</tr>
<tr>
<td>$Q_2$</td>
<td>Ziyang Liu, article, title, XML Keyword Search</td>
</tr>
<tr>
<td>$Q_3$</td>
<td>Gasol, USA</td>
</tr>
<tr>
<td>$Q_4$</td>
<td>Gasol, Brazil</td>
</tr>
<tr>
<td>$Q_5$</td>
<td>Grizzlies, Gasol, Brown, position</td>
</tr>
</tbody>
</table>

Table 1: Sample keyword queries for the relevancy. Our approach exploits the semantics of the keyword query in evaluation process. In addition, unlike previous approaches which simply base on the tag name or the label of a node to identify the relationship between nodes, we use node categories to capture node types and their equivalent node types. Therefore, our approach can tackle the aforementioned limitations of existing approaches. The contributions of this paper are described as follows.

- Proposed a novel notion called Relevant LCA (RLCA) to capture the semantics of LCA nodes which the fragments rooted at those nodes are meaningful and relevant to the keyword query.
- Exploited the both structure of data and the semantics of keyword queries to effectively identify meaningful fragments for XML keyword search.
- Addressed the problem of keyword ambiguity by exploiting the search conditions from the query.
- Implemented the algorithms and conducted extensive experiments on real data sets to show that the proposed RLCA outperforms existing approaches in term of effectiveness and precision while maintaining the efficiency.

The remainder of this paper is organized as follows. Next section discusses and analyzes the weaknesses of existing solutions and inspires our motivation. Section 3 introduces the notion of RLCA and its semantics. In Section 4, we present the algorithm of RLCA. Extensive experimental evaluations are provided in Section 5. Section 6 discusses several orthogonal studies could be incorporated into our work. Finally, the conclusions of the paper are presented in Section 7.

![Sample XML document $D_2$.](image)

Figure 2: Sample XML document $D_2$. 
CVLCA. Two nodes u and v are meaningfully related if there does not exist a node u' (or v') which has the same label with u (or v) and LCA(u', v) (or LCA(u, v)) is a descendant of LCA(u, v). Based on this definition, MLCA cannot succeed in identifying Figure 4b as an irrelevant fragment because two nodes [0.1.0.0] and [0.1.2.1] are meaningfully related. The reason is that there exists node [0.1.0.1] which has the same label "nationality" with node [0.1.2.1] and LCA([0.1.0.0],[0.1.0.1])= node [0.1.0] is a descendant of node [0.1]=[LCA([0.1.0.0],[0.1.2.1])]. However, MLCA will fail to identify Figure 4b as an irrelevant result when node [0.1.0.1] is null, e.g. due to being optional.

In addition, those approaches regarded the keyword query as a set of keywords. The semantics and relationships between keywords have not been exploited yet in evaluating the meaningfulness of a fragment. For example, if query \(Q_2=\{\text{Ziyang Liu}, \text{article}, \text{title}, \text{XML Keyword Search}\}\) is issued on the sample document \(D_1\) which is to find all articles of Ziyang Liu with titles containing "XML Keyword Search", all previous approaches identify Figure 3b, 3c and 3d as relevant fragments to query \(Q_2\). However, we can see that only Figure 3b has the title consisting of "XML Keyword Search". Figure 3c and Figure 3d contain "XML Keyword Search" in the node abstract and node reference respectively. Hence they are unlikely relevant to query \(Q_2\). Current approaches simply return all these fragments, so they have very low precision.

To solve the discussed problems, this paper exploits the structure of XML data in combination with the semantics of keyword query in order to improve the precision and recall of the existing approach. The details of our work is provided in the following sections.

3 Relevant LCA

3.1 XML Data and Node Types

As discussed previously, existing approaches use node labels to identify the relationship between nodes. Therefore, they may fail to recognize the meaningless correlations between nodes with different labels but having equivalent types as well as the meaningful relationship between nodes with the same label. To tackle with these problems, we use the prefix path of a node, combined with the node category to form the node type. The prefix path of a node is the path from the root to its parent. For instance, the prefix path of node [0.1.0.0] in document \(D_2\) is \[team/players/player\] (refer to Fig 2). For node categories, we adopted the classifications defined by (Liu & Chen 2007).

1. A node represents an entity if it corresponds to a *-node in the DTD.
2. A node denotes an attribute if it does not correspond to a *-node, and only has one child, which is a value.
3. A node is a connection node if it represents neither an entity nor an attribute. A connection node can have a child that is an entity, an attribute or another connection node.

As pointed out by the authors, the above classifications may fail to identify some multi-attributes as entities (refer to (Liu & Chen 2007) for details). Thus, we apply human intervention to fix the misleading identified nodes after classifying the node categories.

Definition 1 (Node Equivalence & Same Type)

Two nodes u and v are equivalent if they have the same prefix path and the node category. Two nodes u and v have the same node type if they are equivalent and they have the same node label.

Using Definition 1, our work can distinguish that nodes player [0.1.0] and goalkeeper [0.1.1] in the document \(D_2\) are in the equivalent type because they have the same prefix path which is [team/players/player] and both are entities. Notice that two nodes with the same label OR the same prefix path do not guarantee they are equivalent. For example, two nodes name [0.0] and name [0.1.0.0] in Figure. 4 have the same label but
they have different types since they have different prefix paths, which are \{team\} and \{team/players/player\} respectively. Similarly, two nodes name \{0.0\} and players \{0.1\} have the same prefix path (team) but they have different types because name \{0.0\} is an attribute while players \{0.1\} is a grouping node. Based on node types, we formally define the meaningful relation between two nodes as follows:

**Definition 2 (Meaningfully Related Nodes)**

Two nodes \(u\) and \(v\) are meaningfully related if either of following conditions holds:

(i) \(u\) and \(v\) have the same type.

(ii) there do not exist two nodes with the equivalent type on the shortest path between \(u\) and \(v\), except \(u\) and \(v\).

Notice that our definition is different from XSearch in two main points: (i) It accepts that two nodes with the same type can be meaningfully connected in a subtree, due to the fact that a user may be interested in finding more than one entities with the same type. Considering query \(Q_3\), for instance, the user may be interested in returned fragments which contain information about both players "Gasol" and "Brown". (ii) For queries to find information about single entity (e.g query \(Q_4\)), we use node types from Definition 1 to detect the relevancy of fragments rather than simply use node labels. Hence, we can detect that some nodes can be equivalent even though they have different labels, such as nodes player \{0.1.1\} and goalkeeper \{0.1.2\} in document \(D_2\) (refer to Fig. 2). In contrast, some nodes with the same label (e.g name \{0.0\} and name \{0.1.0.0\} in document \(D_2\)) can have different types.

By Definition 2, we can improve the precision and recall of previous approaches. For instance, Figure 4b can be detected as an uninteresting fragment to query \(Q_4\). Conversely, Figure 4c is returned as a relevant result to query \(Q_5\). However, both definitions 1 & 2 cannot help us identify which of those fragments in Figure 3 is relevant to query \(Q_2\) yet. To solve this problem, next section will analyze semantics of the keyword query, which is another important factor for effective XML keyword search.

### 3.2 Semantics of Keyword Query

Previous approaches treat keyword query just as a set of keywords. We adopt the classification approach proposed by (Liu & Chen 2007) for grouping keywords in a query into two categories: (i) if an input keyword \(k\) matches the value of a node or it matches the label of a node that has got a descendant matching another keyword then \(k\) specifies a predicate, corresponding to the where clause in XQuery; (ii) otherwise, it is treated as a return node, corresponding to the return clause in XQuery.

Their work focuses on identifying return nodes from a given keyword query; each predicate is just a keyword in the query. For example, query \(Q_2 = \{\text{Ziyang Liu}, \text{article title XML Keyword Search}\}\) comprises of all predicates and each keyword is a predicate. All subtrees in which each contains all keywords in the query are returned as relevant results. Following that classification, Figures 3b, 3c, 3d are all relevant results to the query \(Q_2\) because each of them contains all input keywords in the query. However, actually only Figure 3b is relevant to the query \(Q_2\). Thus, an immediate question is how we can group a given keyword query \(Q = \{k_1, k_2, \ldots, k_m\}\) to form predicates and return nodes. We observe that there are relations between keywords in the query and usually the keywords can be combined together to form a predicate or return node. For example, the query \(Q_2\) can be expressed as two predicates (\{\text{Ziyang Liu} and title:XML Keyword Search\}) and a return node (article). From the observation, given a keyword query \(Q = \{k_1, k_2, \ldots, k_m\}\) and an XML tree \(T\), we can group keywords together to form predicates and return nodes based on the algorithm 1.

For example, consider query \(Q_2 = \{\text{Ziyang Liu}, \text{article, title, XML Keyword Search}\}\) on the document \(D_1\), based on above Algorithm 1, we can identify that "article" is a return node, while "Ziyang Liu" and "title:XML Keyword Search" are two predicates. Specifically, keywords "Ziyang Liu" matches the value of nodes \{0.2.0.1.1\}, \{0.2.0.1.3\}, \{0.2.0.2.1\} and \{0.2.0.3.1\} in document \(D_1\). Keyword "article" matches the la-
Algorithm 1 [Keyword Classification]

Input: a keyword query \( Q = \{k_1, k_2, \ldots, k_m\} \) and an XML tree \( T \).

Output: the associated predicates and return nodes

1. if a keyword \( k_i \) matches the label of a node associated with a descendant which has got a value matching keyword \( k_{i+1} \) (or a sequence of keywords \( k_{i+1}, k_{i+2}, \ldots, k_{i+j}, i + j \leq m \)) then
   2. return \( k_i \) and \( k_{i+1} \) (or \( k_{i+1}, k_{i+2}, \ldots, k_{i+j} \)) form a predicate, noted as \( k_i : k_{i+1} \) (or \( k_i : k_{i+1}, k_{i+2}, \ldots, k_{i+j} \)).
3. else if a keyword \( k_i \) (or sequence of keywords \( k_i, k_{i+1}, k_{i+j}, i + j \leq m \)) matches the value of a node which has no ancestor matching another keyword \( k_l \) \((1 \leq l < i)\) then
   4. return \( k_i \) (or \( k_i, k_{i+1}, \ldots, k_{i+j} \)) form a predicate, noted as \( k_i : k_{i+1}, k_{i+2}, \ldots, k_{i+j} \).
5. else
6. return \( k_i \) is a return node. It means \( k_i \) matches the label of a node which has no any descendants having a value matching \( k_{i+1} \) (or a sequence of keywords \( k_{i+1}, k_{i+2}, \ldots, k_{i+j} \)).
7. end if

Definition 3 (RLCA) Given a keyword query \( Q = \{k_1, k_2, \ldots, k_m\} \) which is classified into a set of return nodes \( R \) and a set of predicates \( P \). A RLCA fragment of \( Q \) on document \( D \) is a subtree \( T \) of \( D \), which holds the following conditions:

1. \( T \) satisfies every predicate in \( P \) at least once,
2. \( T \) contains every return node in \( R \) at least once,
3. for any pair of nodes \( u \) and \( v \) in \( T \), either (i) \( u \) is meaningfully related to \( v \) or (ii) there exists a node \( v' \) having the same type with \( v \) while \( u \) is meaningfully related to \( v' \).
4. no proper subtree of \( T \) can hold for the above conditions.

Algorithm 2 [Potential Candidates]

Input: query \( Q \), XML tree \( T \).

Output: a set of potential candidates which each of them contains all predicates and return nodes in \( Q \), XML tree \( T \).

1. Classifying keywords in query \( Q \) into a set of predicates \( P \) and a set of return nodes \( R \) based on Algorithm 1.
2. Identifying all subtrees in \( T \) which each of them satisfies both \( P \) and \( R \) as potential candidates (adapted the Algorithm proposed by (Cohen et al. 2003)).

Stage 2: Relevant LCA. This stage focuses on how to evaluate the meaningfulness of a potential candidate returned from Stage 1. It includes a procedure called Meaningful \((u,v)\) (line 11 of Algorithm 3) which identifies the meaningful connection between a pair of nodes \( u \) and \( v \) as defined in Definition 2. Then, this procedure is used to identify the meaningfulness of a potential candidate. Notice that in Stage 1, we have identified a set of potential candidates already satisfying conditions 1, 2 and 4 of Definition 3. Therefore, in this stage, if a potential candidate in the returned results, only if it satisfies the condition

However, the fragment in Figure 4c is meaningful to query \( Q_5 \) because for every pair of nodes \( u \) and \( v \), either \( u \) is meaningfully related to \( v \) or there exists a node \( v' \) which is the same type with \( u \) while \( v \) and \( v' \) are meaningfully related. Specifically, the pair of nodes name \([0.1.0.0]\) and name \([0.1.1.0]\) (as well as the pair of nodes position \([0.1.0.2]\) and position \([0.1.1.2]\)) are meaningfully related because they have the same type. The pair of nodes name \([0.1.0.0]\) and position \([0.1.0.2]\) (as well as the pair of nodes name \([0.1.1.0]\) and position \([0.1.1.2]\)) are meaningfully related because there do not exist two nodes with the equivalent type on the path between them. Now, considering two nodes name \([0.1.0.0]\) and position \([0.1.1.2]\), we can find another node position \([0.1.0.2]\) having the same type with position \([0.1.1.2]\) which is meaningfully related to the node name \([0.1.0.0]\). It is similar for nodes name \([0.1.1.0]\) and position \([0.1.0.2]\).

Finally, the node name \([0.0]\) is meaningfully related to all other nodes because there do not exist two nodes with equivalent type on the paths between them.

4 RLCA Algorithm

In this section, we propose an algorithm to identify all meaningful fragments of a given keyword query according to Definition 3. We divide our algorithm into two stages. In stage 1, we focus on identifying all fragments which each satisfies condition 1, 2 and 4 of Definition 3 as potential candidates. Stage 2, we identify the meaningfulness and relevancy of the potential results related to the query.

Stage 1: Identifying potential candidates. Given a keyword query and an XML document modeled as a tree \( T \), the aims of this stage are: (i) classifying the keywords into two sets of predicates and return nodes using Algorithm 1; (ii) identifying a set of potential candidates which each candidate satisfies conditions 1, 2 and 4 of Definition 3. Notice that predicates and return nodes in our work have the forms of tag-keyword queries in: (Cohen et al. 2003). Therefore, after classifying the query keywords into predicates and return nodes, we adopted the algorithm in (Cohen et al. 2003) to calculate the set of potential candidates. The details of this stage is demonstrated by Algorithm 2.
algorithm evaluates every pair of nodes in the candidate to evaluate its relevancy (line 2, 3 and 4). At the end of Algorithm 3, a set of results returned will be guaranteed to be meaningful and relevant to the input query.

Algorithm 3 [RLCA]

**Input**: a set of potential candidates identified from Algorithm 2.

**Output**: a set of relevant results.

1: for each potential candidate $F$ do
2: for every pair of two nodes $u$ and $v$ in $F$ do
3: if not Meaningful ($u$, $v$) then
4: if $\exists v' \in F$ | Meaningful ($u$, $v'$) then
5: return false
6: end if
7: end if
8: end for
9: return true
10: end for
11: Meaningful ($u$, $v$)
12: {return true if $u$ and $v$ are meaningful related according to Definition 3, otherwise returns false}
13: if $u$ and $v$ have the same type then
14: return true
15: else
16: $\{\text{find all nodes on the paths from } u \text{ to } v\}$
17: OnPathNodes ← all nodes on the path between $u$ and $v$
18: if $\exists u_1, u_2, u', v' \in \text{nodesOnPaths} \land (u_1 \text{ and } u_2 \text{ are equivalent then} \text{ return false}
19: else
20: return true
21: end if
22: end if

5 Experiments

To evaluate the performance of the proposed RLCA algorithms, we designed and performed a comprehensive set of experiments on real data sets. Our approach is compared against the state-of-art proposals of XSEarch (Cohen et al. 2003) and CVLCA (Li et al. 2007). We employ four metrics, elapsed time, precision, recall and $\mathcal{F}$-measure to evaluate the efficiency and effectiveness of RLCA compared with other approaches. The relevant results are manually obtained from user studies. To obtain a set of relevant result for each keyword query, we manually defined a XQuery query for each keyword query. Those XQuery queries were run on the same set of data and the returned results are used as “ground truth” for our experiments.

5.1 Experiment Setup

The experiments were conducted on a 3.2GHz P4 CPU running Windows XP Professional with 1GB of RAM. The algorithms were implemented in Java. We used Oracle Berkeley DB\(^1\) as a tool for creating indexes.

**Data Set**. We have tested several real data sets: DBLP\(^2\), SIGMOD Record\(^3\) and XMark\(^4\). The DBLP is a data set about bibliographic information on major computer science journals and proceedings. SIGMOD Record is an XML version of SIGMOD Record articles database.

**Query Set**. Our query set consists of 60 queries in total. For each data set, we tested twenty keyword queries. The queries are selected to represent a variety of cases, where both labels and values of data are used.

5.2 Effectiveness Test

This section evaluates the effectiveness of our algorithms, compared with the state-of-art proposals of XSEarch (Cohen et al. 2003) and CVLCA (Li et al. 2007). We conducted twenty keyword queries for each data set. The effectiveness and efficiency was measured by four standard metrics borrowed from information retrieval (IR) literature, precision, recall and $\mathcal{F}$-measure and elapsed time. The improvements of precision and $\mathcal{F}$-measure are demonstrated by Figure. 5a and Figure. 6a respectively.

5.2.1 Search accuracy

We used the standard metric, precision, to measure the search accuracy which indicates the fraction of results in the returned answer that are correct. We can see, in Figure. 5a, RLCA outperforms CVLCA (Li et al. 2007), especially in some tested queries i.e. $Q_2$. In these queries, our keyword classification mechanism mostly identify correctly user’s intentions which in turns successfully pruning irrelevant results. Whilst no such a classification approach employed in CVLCA algorithm, thus the existing approach may include irrelevant results in their answer set. Precision of XSearch (Li et al. 2007) is comparable to our approach in most of data sets, however, in XSEarch (Li et al. 2007) users are required to submit the keyword query in forms of tag-keyword set. It means that the users have to manually specify predicates and return results which have to satisfy. In contrast, our proposed algorithm can handle this task automatically at the processing time. So, it eliminates the burden for users when using the system. Even though XSearch

5.2.2 Search completeness

We employed the recall metric to measure the search completeness which indicates the fraction of all correct results actually captured in the returned answer. Figure. 5b demonstrates that our approach outperforms both XSEarch (Cohen et al. 2003) and CVLCA (Li et al. 2007). We observed that XSEarch (Cohen et al. 2003) and CVLCA (Li et al. 2007) fail to recognize meaningful fragments to a query which has more than one predicate with the same type, such as $Q_2$ in Table 1. The reason, as analyzed in Section 2.2, is that both the existing approaches employed node labels to inferring node interconnections; meanwhile our approach proposed using node types for that aim. Overall, the advantages of our approach over all existing approaches can be demonstrated by employing F-measure for further evaluations.

5.2.3 $\mathcal{F}$-measure

To further evaluate the effectiveness of our approach and the state-of-art algorithms, we employed $\mathcal{F}$-measure which is the weighted harmonic mean of precision and recall. $\mathcal{F}$-measure is calculated as:

$$\mathcal{F} - \text{measure} = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times (\text{precision} + \text{recall})}$$  \hspace{1cm} (1)
In our experiments, recall and precision are evenly weighted ($\beta = 1$). The $F$-measure of our approach and existing approaches are shown in Figure 6a. We can see that the overall effectiveness of our algorithm outperforms the other algorithms. This result strengthens our thoughts that exploiting structural semantics of data in combination with semantics of query can improve the effectiveness and relevancy of XML keyword search.

5.3 Search Efficiency

The execution time of our approach compared with XSEarch (Cohen et al. 2003) and CVLCA (Li et al. 2007) are presented in Figure 6b. Our approach is quite slower than XSEarch (Cohen et al. 2003). This is reasonable because in our approach query keywords are classified into predicates at the processing time. In contrast, the query in XSearch is already in form of predicates before the query is executed. In addition, our algorithm use more complicated evaluation procedures to guarantee that meaningful fragments (like Figure 6b) are not missed. CVLCA (Li et al. 2007) is the fastest one, however it treats a query as a set of words and semantics of query are not analyzed.

6 Discussions

We have reviewed the existing work of XML keyword search on identifying relevant matches in Section 2. In this section, we discuss several orthogonal studies which can be incorporated into our work.

Efficiently Retrieving LCAs. The problem of efficiently identifying the set of LCAs has been extensively studied in literature (Guo et al. 2003, He et al. 2007, Kacholia et al. 2005, Shao et al. 2007, Sun et al. 2007, Xu & Papakonstantinou 2005, 2008). Stack-based algorithm for efficiently evaluating LCAs has been proposed in (Guo et al. 2003). (He et al. 2007). It exploits a bi-level index in conjunction with bi-direction search (Kacholia et al. 2005) for pruning and accelerating the search. To reduce the index space, it partitions a data graph into blocks: The bi-level index stores summary information at the block level to initiate and guide search among blocks, and more detailed information for each block to accelerate the search within blocks. In (Xu & Papakonstantinou 2005, 2008), the characteristics of LCAs have been described to accelerate the evaluation.

Ranking Schemes and Top-K. The issue of ranking the results of keyword search over XML documents based on the relevant scores or user desired references has been studied in many existing work (Amer-Yahia et al. 2005, Guo et al. 2003, Lau & Ng 2008, Li, Feng, Wang, Yu & He 2008, Theobald et al. 2008). (Amer-Yahia et al. 2005, Li, Feng, Wang, Yu & He 2008) propose a scoring methods that are inspired by $tf*idf$ and accounted for both content and structure of returned subtrees. In (Guo et al. 2003), the results of XML keyword search are ranked based on the ranking model adapted from PageRank hyperlink metric. A scoring model proposed in (Theobald et al. 2008) is an extension of the probabilistic-IR Okapi BM25 model. An adaptive ranking model has been studied in (Lau & Ng 2008), which can be adaptive to satisfy various needs and preferences in searching XML data. Whilst (Marian et al. 2005, Kimelfeld & Sagiv 2006) focus on efficiently identifying top-k relevant results for XML keyword search.
This paper focuses on the problem of effectively and accurately identifying the relevant results to XML keyword search. Other work discussed in this section are orthogonal issues and may be incorporated into our work.

7 Conclusions

In this paper, we have investigated the problem of keyword search over XML documents, with the aim of identifying the most relevant and meaningful fragments to XML keyword search. We proposed a novel approach namely Relevant LCA (RLCA) to accurately answer XML keyword queries. We firstly introduce the concept of equivalent types to capture a set of nodes which their types are equivalent, rather than using simple label of a node. Then we classified keywords in the query into conditions (or predicates) and return nodes. A fragment is meaningful only if it satisfies all predicates and return nodes in the query, not just only contains all query keywords. Finally, we introduced the semantics of a meaningful fragment as relevant results to the query. We have implemented our algorithms and the extensive experiments on real data sets have been conducted. The effectiveness of our algorithms, compared with the state-of-art existing proposals was measured carefully by three metrics: precision, recall and F-measure. The experimental results were analyzed in all three metrics and the results show that our approach achieves higher effectiveness than existing approaches.

References


