Journal of Web Engineering, Vol. 11, No.2 (2012) 131-145

MAXLCA: A NEW QUERY SEMANTIC MODEL FOR XML KEYWORD SEARCH

NING GAO, ZHI-HONG DENG*, JIA-JIAN JIANG, HANG YU

Key Laboratory of Machine Perception (Ministry of Education)
School of Electronic Engineering and Computer Science, Peking University
{ninggao, zhdeng, jiangjiajian, yh_030603 }@pku.edu.cn

Received February 16, 2011
Revised January 16, 2012

Keyword search enables web users to easily access XML data without understanding the complex data schemas. However, the ambiguity of keyword search makes it arduous to select qualified data nodes matching keywords. To address this challenge in XML datasets whose documents have a relatively low average size, we present a new keyword query semantic model: MAXimal Lowest Common Ancestor (MAXLCA). MAXLCA can effectively avoid false negative problem observed in ELCA, SLCA and XSeek. Furthermore, we construct an algorithm GMAX for MAXLCA-based queries that is proved efficient in evaluations. Experiments on INEX show that the search engine using MAXLCA and GMAX outperforms in all three comparative criteria: effective, efficient and processing scalability.

Key words: Keyword Search, XML, MAXLCA, GMAX
Communicated by: M. Gaedke & P. Fraternali

1 Introduction

As a large number of corpuses are represented, stored and published in XML format, how to find useful information from XML documents has become an increasingly important issue. Keyword search has been proved to be a user friendly way of querying HTML documents in the World Wide Web and is a widely accepted way for users to find information. As a result, keyword search over XML documents [1, 2, 3, 4, 5] has been extensively studied in recent years.

Although keyword search is user-friendly, the inherent ambiguity of query keywords and XML documents makes it hard to define relevance between answers and keyword search. To identify relevant results for a certain query, researchers have proposed many semantic models [6, 7, 8, 9, 10, 11, 12], such as the highly cited ELCA [6], SLCA [8], and XSeek [9]. These approaches are based on different underlying principles and criteria.

* Corresponding author.

However, due to the diversity of XML datasets, it is hard, if not impossible, to identify one semantic search model best appropriate under every circumstance. For a given XML dataset, therefore, we define the following three criteria to locate the *best-matched* query model:

- **Effectiveness**. A query semantic model is used to identify which elements are relevant and should be returned to the user. The definition of result elements according to certain semantic model should meet the user's search intention as accurately as possible. The evaluation standard of effectiveness is precision and recall. In other words, the search result returned to user should: (i) minimize irrelevant information; (ii) maximize relevant information.
- **Efficiency**. Results should be returned to user in a short response time, in which relevant results retrieval takes up a high proportion. Thus, an applied semantic model should adopt an efficient searching algorithm to ensure the efficiency of the search engine.
- **Processing Scalability**. The process of getting relevant results should show finer scalability as the increase of the number of query keywords. That is, search efficiency should be stable as the number of keywords gets larger.

In this paper, we focused on searching such XML datasets with relatively low average size of documents. For instance, table 1 shows the average document length of the Wikipedia dataset we used in this paper. The average document length of the dataset is 10302.3 characters. Furthermore, by given a query, the documents in Wikipedia are recognized as "relevant" or "irrelevant" by users in the evaluation system of INEX. The average document length of relevant documents and irrelevant documents are also shown in table 1. By further observing the dataset, these documents mostly concentrate only on one topic, suggesting that the user prefers a more intact result with high probability. According to our observation, the existing dominant semantic models, such as SLCA, ELCA and XSeek, are likely to ignore some relevant parts of the result. We consider it as false negative[a] problem, which will be further analyzed in section 3. Intuitively, a semantic query model will achieve better performance on effectiveness by avoiding the false negative problem, verified by the search effectiveness test in section 5.2.

Table 1. Average Document Length

	relevant	**irrelevant**	**dataset**
document length (chars)	12087.7	11310.8	11342.3

To retrieve such datasets with *short* documents, we introduced a new query semantic model called MAXimal Lowest Common Ancestor (MAXLCA). Furthermore, we developed an up-down non-iterative search algorithm GMAX, to obtain MAXLCA results efficiently. Experiments show that the search engine incorporating MAXLCA and GMAX as semantic query model and retrieve algorithm matches the above three principles better than three other dominant models, ELCA, SLCA and XSeek.

The main contributions of this paper are as follows:

[a] An element in XML is relevant but false recognized as irrelevant.

1. We reason that XML query strategy should be designed according to the properties of certain corpuses. To the best of our knowledge, this is the first work focusing on retrieving XML corpuses of *short* documents.
2. We introduce a new semantic XML search model: MAXLCA. It avoids the false negative problem caused by the native ambiguity of keywords search.
3. We design and implement a searching algorithm GMAX for searching MAXLCA result nodes. GMAX is an up-down and non-iterative algorithm with fairly high efficiency and processing scalability.
4. Experiments verify that the search engine with MAXLCA and GMAX is more appropriate to XML dataset with *short* documents, by outperforming in all the three criteria.

The remainder of the paper is organized as follows. In section 2 we review the related work. In section 3, we analyze the false negative problem of the other three comparison semantic models and introduce the MAXLCA and its formal definition. The homologous high effectiveness and efficient searching algorithm GMAX is described in section 4. The experimental results and analysis are discussed in section 5. In section 6, we make a conclusion and point out some future research issues.

2 Related Work

To cope with the query semantic problem, several approaches have been pro-posed for identifying relevant results for XML keyword search [6, 8, 9, 21, 22, 23]. *Meet* operator is proposed in [21], operating on multiple sets where all nodes in the same set are required to have the same schema, which is a special case of the all LCAs problem. The *meet* operator of two nodes o1 and o2 is implemented using joins on relations, where the number of joins is the number of edges on the path from o1 to their Lowest Common Ancestor (LCA).

ELCA was presented by Guo, et.al. [6]. In ELCA, a node in XML tree is designated as a result node only if it contains at least one occurrence of each keyword in its subtree, after excluding the occurrences of the keywords in its descendants that already contain all the keywords. The related definitions are as follow:

Definition 2.1. Given a keyword query $Q = \{ k_1, \ldots, k_m\}$, an XML tree X_{tree}, $ELCA(Q, X_{tree})$ is defined as follows:

$ELCA(Q, X_{tree}) = \{n \mid n^* \in LCA(Q, X_{tree}) \}$, where n^* is node n after exclud-ing all its child nodes which belong to LCA node set.

[6] introduced a Dewey Inverted List (DIL) algorithm. [25] followed the framework of [6], but proposed a more efficiency method of defining candidate nodes and computing the results. To be specific, [25] firstly defines the instance of a node as the number of matches within the node for each keyword. For example, there is an example tree in figure1 and a query {Albert Einstein}. The instances of node 0.3.2.0.0 and 0.3.1.1 are {1,1} and {0,1} respectively. Then the algorithm compares the number of keyword instances that a candidate node *n* has covered with the number of instances that *n*'s children who contain all keywords have covered. If n contains an instance solely matching for each keyword, it will be recognized as an ELCA node.

SLCA was put forward by Xu, et.al. [8]. For a query, a node in XML tree is considered as a result node only if it contains at least one occurrence of each keyword in its subtree, and none of its descendants does. The formal definition of SLCA is showed as follow:

Definition 2.2. Given a keyword query $Q = \{ k_1, \ldots, k_m\}$, an XML tree X_{tree}, $SLCA(Q, X_{tree})$ is defined as follows:

$SLCA(Q, X_{tree}) = \{n \mid n \in LCA(Q, X_{tree}) \wedge \neg(\exists\, n' \in LCA(Q, X_{tree}), n \phi n')\}$, where $n \phi n'$ means n is an ancestor of n'.

[8] proposed the searching algorithm Scan Eager (SE) and Indexed Lookup Eager (ILE) algorithms. [24] followed the work of [8], and proposed a novel approach for processing SLCA results. The processing of searching for SLCA nodes in [8] is considered as binary-SLCA approach(BS), while the processing in [24] is called multiway-SLCA approach(MS). To compute a potential SLCA node, BS approach will pick up a node from the keyword list with the lowest frequency. On the other hand, MS will pick an "anchor" node from among the m keyword data lists to drive the computation. By doing so, MS can optimize the processing by skipping redundant computations.

XSeek was introduced by Liu et al [9]. XSeek adopts the approach proposed in [8]. Moreover, according to XSeek, nodes in XML tree are grouped into three categories: entity node, attribute node and connection node. Then, XSeek also analyzes keyword patterns to determine return information. The related definitions are as follow:

Definition 2.3. *Entity Node*: If a node has siblings under the same name, then this indicates a many-to-one relationship with its parent node, and is considered to represent an entity. E.g. <workshop>, <paper> in Figure 1 are considered as entity nodes. The result nodes presented by XSeek are all entity nodes.

Definition 2.4. *Attribute Node*: If a node does not have siblings of the same name, and only one child is a value, then it is considered to represent an attribute. E.g. <date>, <title>, <editors>, <author> are defined as attribute nodes.

Definition 2.5. *Connection Node*: A node is a connection node if it represents neither an entity nor an attribute. E.g. <proceedings> in Fig. 1 is a connection node.

3 MAXLCA: MAXimal Lowest Common Ancestor

As aforementioned, ELCA, SLCA and XSeek are prestigious semantic models to define query semantic for XML keyword search. Similarly, these semantic models all adopt a variant of lowest common ancestor concepts. However, they are likely to ignore some candidate nodes that also satisfy users' interests. In this section, we analyze the false negative problem of these semantic models and present the MAXimal Lowest Common Ancestor query semantic model.

Consider an example shown in Figure 1. This sample XML tree presents the web page of *Albert Einstein* in Wikipedia. Suppose there is a student who wants to search for the information of Albert Einstein for his physical paper, then {Albert Einstein} will be a submitted query. According to the definition of ELCA, SLCA and XSeek, the nodes 0.3.1.0 and 0.3.2.0.0 will be recognized as results. On the other hand, the nodes 0.3.1.1 and 0.3.2.0.1 are defined as irrelevant nodes. The aforementioned four nodes are the presentations of four paragraphs in the web page. If evaluated by the user, all these

four nodes will probably be decided as relevant since they are all introductions of Albert Einstein. However, in the nodes 0.3.1.1 and 0.3.2.0.1, the last name *Einstein* is used as the abbreviation of *Albert Einstein*, which is common in expressing a person's name. These two nodes are excluded from the returned results since they failed to contain all keywords in the query.

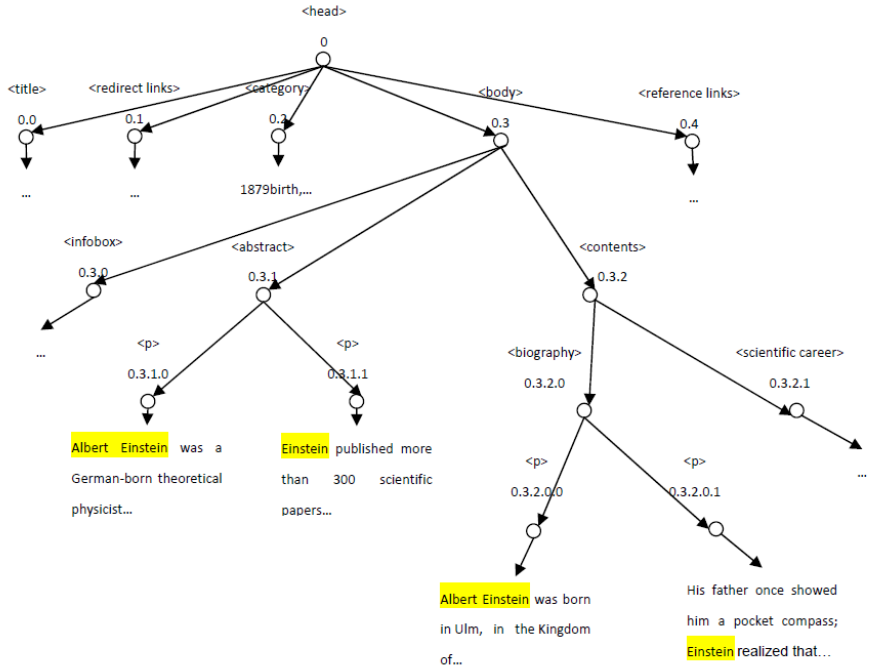


Figure 1 Sample XML tree

From this example, we can conclude that the semantic models of ELCA, SLCA and XSeek are not proper definitions when the query is a person name. The same problem can be extended to a product name. For example, in the web page of Mazda G6, G6 will be used as the abbreviation in the passage. In [21], real query logs from a commercial search engine are used to analyze the search intention of the web queries. Table 2 shows the distribution of the query domain. According to the statistic data, the proportion of people and product query is about 24.12% in all web queries.

Table 2. Distribution of Query Domain of All Queries

Query Domain	Percent
Other	49.05%
Navigational	14.44%
Product	14.31%
People	9.81%
Medical	3.13%
All others	<3.00%

Thus, ELCA, SLCA and XSeek might not be ideal query semantic models for XML keyword search according to the aforementioned false negative problem. To this end, we put forward a new query semantic model called MAXLCA. Note that we adapt XML tree as the data model of XML documents.

Definition 3.1. Given a keyword query $Q = \{ k_1, \ldots, k_m\}$, an XML tree X_{tree}, we define $MAXLCA(Q, X_{tree})$ as follows:

$MAXLCA(Q, X_{tree}) = \{n \mid n \in LCA(Q, X_{tree}) \wedge \neg(\exists\ n' \in LCA(Q, X_{tree}), n' \phi\ n)\}$, where $n' \phi\ n$ means n' is an ancestor of n.

For example, let {Albert Einstein} be a keyword query, the XML tree shown in Figure 1 represents a relevant document. Node 0.3 is the MAXLCA node. Here we note that node 0.3 is the best result, since it is neither too sketchy nor too redundant. The definition of MAXLCA meets user's intention more precisely, and avoids the false negative problem mentioned above. It ensures that the search engine based on MAXLCA can reach the satisfied performance effectively. Plus, information retrieval is an online service, hence the response time is another significant criterion. Therefore, we propose an efficient algorithm GMAX for getting relevant results of MAXLCA in section 4.

4 GMAX: An Efficient Algorithm for Searching MAXLCA

Though the MS approach of computing SLCA results in [24] can improve the efficiency of BS approach in [8] by skipping the redundant results, it still suffers from the framework of iterative. The procedure can only afford computing the SLCA nodes once for two keywords. For example, given a query with three keywords, we have to compute the SLCA result nodes for the first two keywords, and then compute the SLCA result for the third keyword together with the node set generated in the first step. Even worse, the response efficiency decreases rapidly with the increase of the number of query keywords. These make the algorithm time-consuming unsalable. The algorithm of getting XSeek nodes adopts the approach in [8], so that the complexity of getting XSeek nodes is at least the same as getting SLCA nodes.

The algorithm for getting ELCA results in [25] is a non-iterative process. More specifically, no matter how many keywords are contained in the query, the program can compute the ELCA result nodes in one procedure. However, [25] preserves the stack framework of [6] to store and compute father-children relationships. Note that the consumed time of one stack operation in ELCA takes up much more than one comparison operation in getting SLCA nodes. It will be verified by experiment in Section 5.3. Thus, the efficiency of getting ELCA method is much lower than getting SLCA method.

Moreover, both methods for getting SLCA and ELCA result nodes are down-up strategies. The procedure focuses on the leaf nodes at the very beginning, and then looks for the result nodes along the path from the leaf nodes to the roots. As the tree depth increases, the number of nodes in one layer increases evidently. Intuitively, the down-up strategy will take much more time in computing results than up-down strategy since there will be more nodes in lower levels.

Considering the advantages and disadvantages of getting ELCA, SLCA and XSeek policies, we devise an up-down and non-iterative algorithm GMAX for getting MAXLCA result nodes with fairly high efficiency. The detailed algorithm is elaborated in Algorithm 1.

Algorithm 1: Getting_MAXLCA()

```
1: Result = {};
2: getIndex();
 //for each document which matches all the keywords;
3: for (each relevant document){
4: u = 0; // u = root initially;
5: while (u is not a leaf node) {
    //determine whether only one sub-node X of u contains //keywords;
6:   bool isMAXLCA = false; //denote whether u is an MAXLCA node;
7:   int depth = the depth of u; //known from the deweyID of u;
8:   for(i=1 → kw.size()){
      //check if all the keywords appeared in X, the deweyID //of X is the first
depth+1 ID of kw[0];
9:    if (kw[i].deweyID[depth+1] !=kw[0].deweyID[depth+1])
10:   then {isMAXLCA = true; break;}
11:  }
12:  if (isMAXLCA==true) // the MAXLCA node is found
13:  then {break;}
14:  else {u = X}; //continue scanning the sub-node of X;
15:  }
16:  Result = Result + {u};
17:}
getIndex{
```

```
/*for each document contains keyword, there is an adapted struct document.

struct document {

bool relevant; // whether the document matches all keywords;

vector <deweyID> kw; // deweyID of keywords appeared in the documents}*/

read in the location of each keyword, record the corresponding  deweyID per
document;

if(a document matches all the keywords)

then {relevant = true;}

else {relevant = false;}

}
```

Example 4.1. Consider the query {Albert Einstein} and the relevant document shown in Fig. 1. As shown in table 3, there are four leaf elements in the relevant documents containing keyword, which are signed as node 0.3.1.0, 0.3.1.1, 0.3.2.0.0 and 0.3.2.0.1. At the very beginning, u is initialized to root node 0. Then the procedure enters the while loop. Test the sub-nodes of u. By scanning the second row of the candidate result list vertically, we know that only one sub-node 0.3 contains keyword. Then u is assigned as node 0.3. Continue testing the sub-nodes of 0.3, by scanning the third row of the candidate result list, two child nodes 0.3.1 and 0.3.2 contains keyword. Therefore, the procedure will break out from the while loop. A new result node 0.3 is found.

Table 3. A list of candidate answers

Dewey ID	Row 1	Row 2	Row 3	Row 4	Row 5
Albert Einstein	0	3	1	**0**	
Einstein	0	3	1	1	
Albert Einstein	0	3	2	0	0
Einstein	0	3	2	0	1

5 Experimental Study

As stated above, effectiveness, efficiency and scalability are the most crucial criteria in keyword search. In this section, we evaluate the three criteria and con-duct a comparison with ELCA, SLCA and XSeek. In each criterion, MAXLCA outperforms the other models significantly. Furthermore, to explore the features of MAXLCA, we count the number and the average length of the result elements returned by each model.

5.1 Experimental Setup

The experiments are performed on a 2.66GHz Intel (R) Core (TM) CPU run-ning Microsoft Windows XP operating system with 2.0GB memory. The system is implemented in C++. The XML documents

consists of in total 659,388 English files from wiki, which are used in *INEX Ad Hoc Track* in 2008 (AHT)[14]. The total size of these files is 4.6G. In the process of reading and analyzing these XML files, we remove all the stop words from a standard stop word list before stemming. The query set consists of 65 different topics from the topics of AHT. The numbers of keywords in these search topics vary from 1 to 5. As table 4 has shown, most topics in query set contain 2 or 3 keywords.

Table 4. Keywords Number

Keywords	**1**	**2**	**3**	**4**	**5**
Topics	1	23	29	10	2

5.2 Search Effectiveness

Initiative for the Evaluation of XML retrieval (INEX) is a global evaluation platform launched in 2002. The main goal of INEX is to promote the evaluation of XML retrieval by providing a large test collection of XML documents, uniform scoring procedures, and a forum for organizations from Information Retrieval, Database and other relative research fields to compare their results [14]. Both the enactment of the query set and the evaluation criterion target at simulating queries as close to real ones as possible. Each query in the evaluation system is bound to a standard set of "relevant content", which is recognized manually by the participants of AHT. AHT assesses a search result by comparing it with the "relevant content".

Evaluation System of AHT is selected as the standard experiment platform in effectiveness comparison. Here in this paper, the evaluation of effectiveness follows the AHT official measurement, interpolated precision at 1% recall (iP[0.01]), as the evaluation criteria for effectiveness. Considering the impact of different ranking methods in effectiveness, we use two different ranking methods *tf*idf* [15] and BM25 [16]. Both ranking methods are highly cited and reputational.

Based on these two ranking methods, we define the following symbols. *tf(t,e)* is the frequency of keyword *t* appeared in element *e*; *Nd* is the number of files in the collection; *n(t)* is the frequency of keyword *t* appearing in the collection; *len(e)* is the length of element *e*; *avel* is average length of elements in the collection; *Q* is a set of keywords; *score(e,Q)* is the score of element *e* corresponding to query *Q*.

Definition 5.1. *tf*idf*:

$$score(e,Q) = \sum_{t \in Q} tf(t,e) \cdot \log \frac{Nd}{n(t)}$$

This formula is used to calculate the score of leaf nodes. Then we transfer the score recursively from leaf nodes to the root. The score of an inside node is de-fined as the sum of all its child nodes with 0.3 decay.

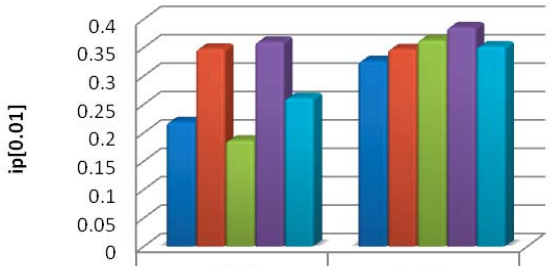
Definition 5.2. *BM25*:

$$score(e,Q)=\sum_{t\in Q} W_t \cdot \frac{(k_1+1)\cdot tf(t,e)}{k_1\cdot(1-b+b\cdot\frac{len(e)}{avel})+tf(t,e)}$$

$$W_t = k_2\cdot\log\frac{Nd}{Nd-n(t)} - k_3\cdot\log\frac{n(t)}{Nd-n(t)}$$

The score of all nodes can be calculated by these formulas. Where the content parameters are: k_1=3, b=0.8, k_2=1, k_3=1.

Figure 2 illustrates the effectiveness of SLCA, ELCA, XSeek, MAXLCA and the baseline Document. The baseline Document takes the whole document as a re-turned result. Confined in both *tf*idf* and *BM25*, MAXLCA performs the best. The definitions of SLCA and ELCA suggest that these two models focus on finding core relevant parts and rejecting the less relevant and irrelevant parts, which show benefit when the searching dataset is formed by long documents. However, for the datasets with short documents, if the content of each document concentrates on one certain topic, the work of finding the relevant parts completely is more important than rejecting the less relevant parts. On the other hand, the definitions of entity, attribute and connection node in XSeek strictly depend on the labels of the XML documents. However, the labels in Wiki dataset are all structural tags, body, template and section for example, which are meaningless in distinguishing the nodes by their semantic differences. Therefore, all three comparison models show some disadvantages.



	tf*idf	BM25
SLCA	0.218815	0.326229
ELCA	0.347921	0.347227
XSeek	0.186291	0.363007
MAXLCA	0.360176	0.385727
Document	0.260985	0.35211

Figure 2 Comparison of Effectiveness

5.3 Processing Efficiency and Scalability

For efficiency, we record the average running time of retrieving results of the above 65 topics. Table 5 shows the details. We find that the time for obtaining MAXLCA results is significantly shorter than that for computing ELCA, SLCA and XSeek results. Additionally, we test the time consuming of topics with different number of keywords and data corpus size. Figure 3 illustrates the experiment

result. Figure 4 is the time consuming under different dataset size. MAXLCA also shows processing scalability with the increase of keywords number and dataset size.

As stated in Section 4, results of SLCA are obtained by an iterative policy. Distinctly, the time consuming of the algorithm increases rapidly when more keywords are added, supported by the experiment result shown in Figure 3. The time consuming of computing SLCA nodes changes under two conditions: (1) the addition of keywords will increase the time consuming; (2) the decreasing of documents containing all keywords will decline the time consuming. As is known, the more keywords a query contains, the less documents in the dataset contains all keywords. Therefore, there is a balance between the number of keywords and the candidate documents. According to the query set and dataset we used in this paper, the evaluation result turns out that the increasing of the keywords affects more than the decreasing of candidate result documents.

The algorithm for getting ELCA results is non-iterative so that there is no conspicuous increasing time cost with the ascent of keywords number. However, when there are only 2 keywords in the query, without the impact of iterative operation, i.e. only basic operations process in the procedure, the time cost of getting SLCA, ELCA, and MAXLCA nodes are 43.3ms, 177.7ms and 44.8ms respectively. This comparison shows that the atom stack operation in algorithm for getting ELCA will take much more time than the atom comparing operation used in getting SLCA or MAXLCA. Thus, the response time of getting ELCA is obviously the highest, also verified by Fig. 3.

Table 5. Comparison of efficiency

	SLCA	**ELCA**	**XSeek**	**MAXLCA**
time consuming (ms)	88.4	218.5	89.3	21.1

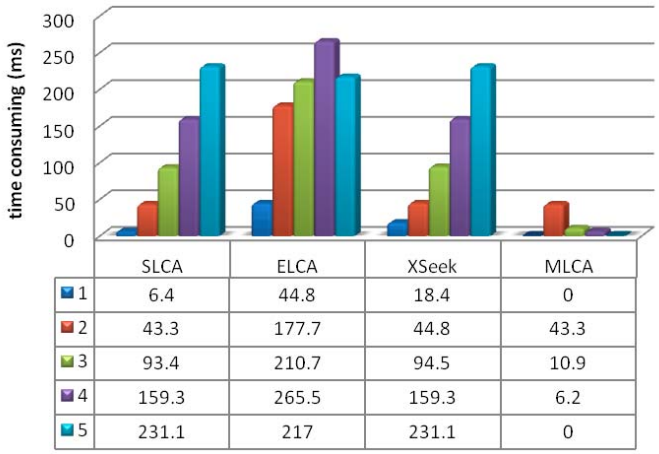
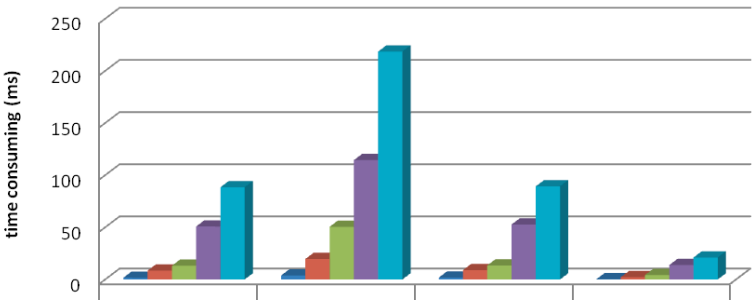


Figure 3 Processing Scalability with Different Keywords Number



	SLCA	ELCA	Xseek	MAXLCA
10,000	1.7	3.8	1.8	0.3
50,000	8.5	19.6	9	2.2
100,000	13.2	50.5	13.5	4.3
300,000	50.8	114.4	52.7	13.8
659,388	88.4	218.5	89.3	21.1

Figure 4 Processing Scalability with Different Data Corpus Size

GMAX satisfies the following three features. The first two features ensure that the efficiency of GMAX is the best in the comparison test, while the third feature explains why MAXLCA shows extraordinary processing scalability with the increase of keywords number.

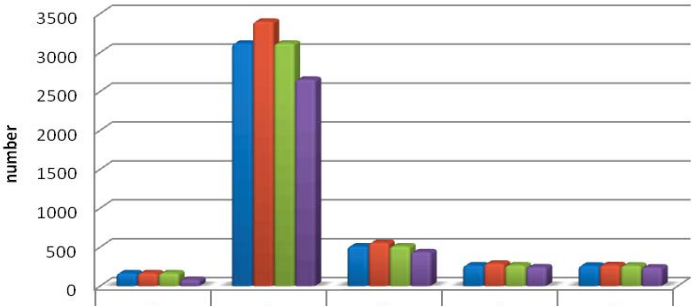
- The atom operation in the procedure of GMAX is comparison operation, taking much less time than stack operation used in ELCA.
- GMAX is an up-down method while both methods of getting SLCA or ELCA result nodes use down-up strategy. As the tree depth increases, the number of nodes in one layer increases markedly. Mostly, in the process of finding the returned nodes, the down-up strategy will scan much more nodes in XML tree than up-down strategy does.
- GMAX is a non-iterative method without increasing time-consuming when keywords are added up. Additionally, since the number of relevant documents (documents that contain all keywords) decreases, the time consuming of GMAX demonstrates a significant drop.

5.4 Features of Result Elements

In the above analysis, we find that MAXLCA outperforms ELCA, SLCA and XSeek. In order to analyze the reason, we summarize the average number of elements in the result set, shown in Fig. 5, and the average length of result elements, shown in Fig. 6 according to different set of topics with various numbers of keywords.

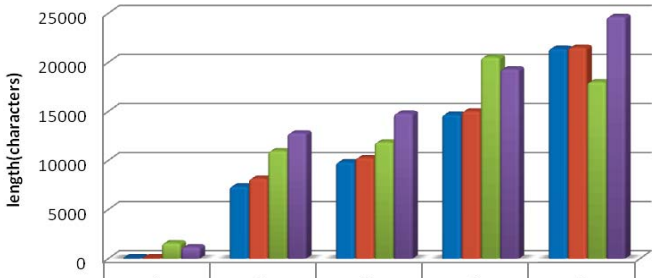
We notice that the average element length (characters an element contained) of MAXLCA is the largest in majority queries and thus the result nodes include more relevant contents. That is to say, the result nodes of MAXLCA are more complete than the result of other semantic models. ELCA, SLCA and XSeek intend to find core relevant elements, ignoring periphery relevant information. However, the goal of MAXLCA is to return the relevant parts of the document as completely as possible. As supported by the effectiveness experiment in section 5.2, users prefer to see the results as complete as possible. All the four semantic models reach the demand of high precision. Furthermore, MAXLCA

achieves a higher recall. This advantage makes MAXLCA the most reasonable choice especially when the search object is a corpus of *short* documents.



	1	2	3	4	5
SLCA	167	3119	512	268	265
ELCA	167	3401	557	291	272
XSeek	167	3119	512	268	265
MLCA	83	2654	438	245	242

Figure 5 Number of Result Nodes



	1	2	3	4	5
SLCA	64.3	7394.7	9852.9	14721.1	21427.5
ELCA	64.3	8176.1	10275.3	15035.7	21526.6
XSeek	1605.44	10990	11872	20529.8	18013.8
MLCA	1192.6	12798.4	14808.8	19324.6	24667.1

Figure 6 Average Element Length

6 Conclusions

In this paper, we investigated the problem of keyword search over short XML documents. We proposed MAXLCA, a new query semantic model, to define the query result. In order to achieve efficiency, we developed an up-down non-recursive search algorithm GMAX. We also conducted extensive experiments to evaluate the performance of MAXLCA. The experiment results show that compared with the other three highly cited semantic models ELCA, SLCA and XSeek, MAXLCA performs better in effectiveness, efficiency and scalability,.

For the future work, there are a variety of interesting research questions. Firstly, we could improve our search engine with some other learn to rank methods [16, 17]. Secondly, we could incorporate top-k search [18, 19, 20] methods into our model and focus on how to find top-k MAXLCA efficiently.

Acknowledgements

This work is partially supported by Project 61170091 supported by National Natural Science Foundation of China and Project 2009AA01Z136 supported by the National High Technology Research and Development Program of China (863 Program).

References

1. N. Gao, Zh. Deng and Y. Xiang. Peking University at INEX 2009: Ad Hoc Track. In INEX, 2009.
2. H. Yu, Zh. Deng and Y. Xiang and N. Gao. Adaptive Top-k Algorithm in SLCA-Based XML Keyword Search. In APWeb, 2010.
3. Z. Liu and Y. Chen. Answering Keyword Queries on XML Using Materialized Views. In ICDE, 2008.
4. V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava. Keyword Proximity Search in XML trees. IEEE Transactions on Knowledge and Data Engineering, 18(4), 2006.
5. Y. Huang, Z. Liu and Y. Chen. eXtract: A Snippet Generation System for XML Search. In VLDB 2008.
6. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In SIGMOD, 2003.
7. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In VLDB, 2003.
8. Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Shortest LCAs in XML Databases. In SIGMOD, 2005.
9. Z. Liu and Y. Chen. Identifying Meaningful Return Information for XML Keyword Search. In SIGMOD, 2007.
10. G. Li, J. Feng, J. Wang, and L. Zhou. Effective Keyword Search for Valuable LCAs over XML Documents. In CIKM, 2007.
11. Z. Liu and Y. Chen. Reasoning and Identifying Relevant Matches for XML Keyword Search. In VLDB, 2008
12. Z. Bao, T. W. Ling, B. Chen, and J. Lu. Effective XML Keyword Search with Relevance Oriented Ranking. In ICDE, 2009
13. B. Schieber and U. Vishkin. On finding lowest common ancestors: Simplification and parallelization. SIAM J. Computing, 1988.
14. INEX. http://www.inex.otago.ac.nz/.
15. D. Carmel, Y.S. Maarek, M. Mandelbrod, et al. Searching XML documents via XML frag-ments. In SIGIR, 2003.
16. Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, Hang Li. Learning to Rank: From Pair-wise Approach to Listwise Approach. Microsoft technique report.
17. Y. Yue, T. Finley, F. Radlinski and T. Joachims. A Support Vector Method for Optimizing Average Precision. In SIGIR 2007.
18. M. Theobald, H. Bast, D. Majumdar, R. Schenkel, and G. Weikum. TopX: Efficient and Versatile Top-k Query Processing for Semistructured Data. In VLDB 2008.
19. H. Yu, Z. Deng, Y. Xiang, N. Gao, Z. Ming, S. Tang. Adaptive Top-k Algorithm in SLCA-Based XML Keyword Search. In APWeb'10.

20. L. Chen, Y. Papakonstantinou. Supporting top-K keyword search in XML databases. In IEEE ICDE'10
21. J. Pound, P. Mika and H. Zaragoza, Ad-Hoc Object Ranking in the Web of Data. In WWW 2010.
22. A. Trotman and B. Sigurbj¨ornsson. NEXI, Now and Next. In INEX 2004.
23. Y. Li, C. Yu, H. V. Jagadish. Schema-Free XQuery. In VLDB 2004.
24. C. Sun, C.Y. Chan and A.k. Geonka. Multiway SLCA-based Keyword Search in XML Data. In WWW 2007.
25. R. Zhou, C. Liu, J. Li: Fast ELCA computation for keyword queries on XML data. InEDBT 2010.