Context-based Diversification for Keyword Queries over XML Data

Jianxin Li, Chengfei Liu *Member, IEEE* and Jeffrey Xu Yu *Member, IEEE*

Abstract—While keyword query empowers ordinary users to search vast amount of data, the ambiguity of keyword query makes it difficult to effectively answer keyword queries, especially for short and vague keyword queries. To address this challenging problem, in this paper we propose an approach that automatically diversifies XML keyword search based on its different contexts in the XML data. Given a short and vague keyword query and XML data to be searched, we firstly derive keyword search candidates of the query by a simple feature selection model. And then, we design an effective XML keyword search diversification model to measure the quality of each candidate. After that, two efficient algorithms are proposed to incrementally compute top-k qualified query candidates as the diversified search intentions. Two selection criteria are targeted: the k selected query candidates are most relevant to the given query while they have to cover maximal number of distinct results. At last, a comprehensive evaluation on real and synthetic datasets demonstrates the effectiveness of our proposed diversification model and the efficiency of our algorithms.

Index Terms—XML Keyword Search, Context-based Diversification.

1 Introduction

Keyword search on structured and semi-structured data has attracted much research interest recently, as it enables users to retrieve information without the need to learn sophisticated query languages and database structure [1]. Compared with keyword search methods in Information Retrieval (IR) that prefer to find a list of relevant documents, keyword search approaches in structured and semi-structured data (denoted as DB&IR) concentrate more on specific information contents, e.g., fragments rooted at the smallest lowest common ancestor (SLCA) nodes of a given keyword query in XML. Given a keyword query, a node v is regarded as an SLCA if (1) the subtree rooted at the node v contains all the keywords, and (2) there does not exist a descendant node v' of v such that the subtree rooted at v' contains all the keywords. In other words, if a node is an SLCA, then its ancestors will be definitely excluded from being SLCAs, by which the minimal information content with SLCA semantics can be used to represent the specific results in XML keyword search. In this paper, we adopt the well-accepted SLCA semantics [2], [3], [4], [5] as a result metric of keyword query over XML data.

In general, the more keywords a user's query contains, the easier the user's search intention with regards to the query can be identified. However, when the given keyword query only contains a small number of vague keywords, it would become a very challenging problem to derive the user's search intention due to the high ambiguity of this type of keyword queries. Although sometimes user involvement is helpful to identify search intentions of keyword queries, a user's interactive process may be time-consuming when the size of relevant result set is large. To address this, we will develop a method of providing diverse keyword query suggestions to users based on the context of the given keywords in the data to be searched. By doing this, users may choose their preferred queries or modify their original queries based on the returned diverse query suggestions.

Example 1: Consider a query q={database, query} over the DBLP dataset. There are 21,260 publications or venues containing the keyword "database", and 9,896 publications or venues containing the keyword "query", which contributes 2040 results that contain the two given keywords together. If we directly read the keyword search results, it would be time consuming and not user-friendly due to the huge number of results. It takes 54.22 seconds for just computing all the SLCA results of q by using XRank [2]. Even if the system processing time is acceptable by accelerating the keyword query evaluation with efficient algorithms [3], [4], the unclear and repeated search intentions in the large set of retrieved results will make users frustrated. To address the problem, we derive different search semantics of the original query from the different contexts of the XML data, which can be used to explore different search intentions of the original query. In this work, the contexts can be modeled by extracting some relevant feature terms of

- *Jianxin Li and Chengfei Liu are with the Faculty of Science, Engineering and Technology, Swinburne University of Technology, Australia. {jianxinli, cliu}@swin.edu.au*
- *Jeffrey Xu Yu is with the Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong, China. yu@se.cuhk.edu.hk*

TABLE 1
Top 10 selected feature terms of q

keyword	features
database	*systems; relational; protein; distributed; oriented; image; sequence; search; model; large.*
query	*language; expansion; optimization; evaluation; complexity; log; efficient; distributed; semantic; translation.*

TABLE 2
Part of statistic information for q

	database *systems* query +				
	language	*expansion*	*optimization*	*evaluation*	*complexity*
#results	71	5	68	13	1
	log	*efficient*	*distributed*	*semantic*	*translation*
#results	12	17	50	14	8
	relational database query +				
	language	*expansion*	*optimization*	*evaluation*	*complexity*
#results	40	0	20	8	0
	log	*efficient*	*distributed*	*semantic*	*translation*
#results	2	11	5	7	5
...	...				

the query keywords from the XML data, as shown in Table 1. And then, we can compute the keyword search results for each search intention. Table 2 shows part of statistic information of the answers related to the keyword query q, which classifies each ambiguous keyword query into different search intentions.

The problem of diversifying keyword search is firstly studied in IR community [6], [7], [8], [9], [10]. Most of them perform diversification as a post-processing or re-ranking step of document retrieval based on the analysis of result set and/or the query logs. In IR, keyword search diversification is designed at the topic or document level. E.g., [7] models user intents at the topical level of the taxonomy and [11] obtains the possible query intents by mining query logs. However, it is not always easy to get these useful taxonomy and query logs. In addition, the diversified results in IR are often modeled at document levels. To improve the precision of query diversification in structured databases or semi structured data, it is desirable to consider both structure and content of data in diversification model. So the problem of keyword search diversification is necessary to be reconsidered in structured databases or semi structured data. [12] is the first work to measure the difference of XML keyword search results by comparing their feature sets. However, the selection of feature set in [12] is limited to metadata in XML and it is also a method of post-process search result analysis.

Different from the above post-process methods, another type of works addresses the problem of intent-based keyword query diversification through constructing structured query candidates [13], [14]. Their brief idea is to first map each keyword to a set of attributes (metadata), and then construct a large number of structured query candidates by merging the attribute-keyword pairs. They assume that each structured query candidate represents a type of search intention, i.e., a query interpretation. However, these works are not easy to be applied in real application due to the following three limitations:

- A large number of structured XML queries may be generated and evaluated;
- There is no guarantee that the structured queries to be evaluated can find matched results due to the structural constraints;
- Similar to [12], the process of constructing structured queries has to rely on the metadata information in XML data.

To address the above limitations and challenges, we initiate a formal study of the diversification problem in XML keyword search, which can directly compute the diversified results without retrieving all the relevant candidates. Towards this goal, given a keyword query, we first derive the co-related feature terms for each query keyword from XML data based on mutual information in the probability theory, which has been used as a criterion for feature selection [15], [16]. The selection of our feature terms is not limited to the labels of XML elements. Each combination of the feature terms and the original query keywords may represent one of diversified contexts (also denoted as specific search intentions). And then, we evaluate each derived search intention by measuring its relevance to the original keyword query and the novelty of its produced results. To efficiently compute diversified keyword search, we propose one baseline algorithm and two improved algorithms based on the observed properties of diversified keyword search results.

The remainder of this paper is organized as follows. In Section 2, we introduce a feature selection model and define the problem of diversifying XML keyword search. In Section 3, we describe the procedure of extracting the relevant feature terms for a keyword query based on the explored feature selection model. In Section 4, we propose three efficient algorithms to identify a set of qualified and diversified keyword query candidates and evaluate them based on our proposed pruning properties. In Section 5, we provide extensive experimental results to show the effectiveness of our diversification model and the performance of our proposed algorithms. We describe the related work in Section 6 and conclude in Section 7.

2 Problem Definition

Given a keyword query q and an XML data T, our target is to derive top-k expanded query candidates in terms of high relevance and maximal diversification for q in T. Here, each query candidate represents a context or a search intention of q in T.

2.1 Feature Selection Model

Consider an XML data T and its relevance-based term-pair dictionary W. The composition method of

W depends on the application context and will not affect our subsequent discussion. As an example, it can simply be the full or a subset of the terms comprising the text in T or a well-specified set of term-pairs relevant to some applications.

In this work, the distinct term-pairs are selected based on their mutual information as [15], [16]. Mutual information has been used as a criterion for feature selection and feature transformation in machine learning. It can be used to characterize both the relevance and redundancy of variables, such as the minimum redundancy feature selection. Assume we have an XML tree T and its sample result set $R(T)$. Let $Prob(x, T)$ be the probability of term x appearing in $R(T)$, i.e., $Prob(x, T) = \frac{|R(x,T)|}{|R(T)|}$ where $|R(x, T)|$ is the number of results containing x. Let $Prob(x, y, T)$ be the probability of terms x and y co-occurring in $R(T)$, i.e., $Prob(x, y, T) = \frac{|R(x,y,T)|}{|R(T)|}$. If terms x and y are independent, then knowing x does not give any information about y and vice versa, so their mutual information is zero. At the other extreme, if terms x and y are identical, then knowing x determines the value of y and vice versa. Therefore, the simple measure can be used to quantify how much the observed word co-occurrences maximize the dependency of feature terms while reduce the redundancy of feature terms. In this work, we use the popularly-accepted mutual information model as follows.

$$MI(x, y, T) = Prob(x, y, T) * log\frac{Prob(x,y,T)}{Prob(x,T)*Prob(y,T)} \quad (1)$$

For each term in the XML data, we need to find a set of feature terms where the feature terms can be selected in any way, e.g., top-m feature terms or the feature terms with their mutual values higher than a given threshold based on domain applications or data administrators. The feature terms can be pre-computed and stored before the procedure of query evaluation. Thus, given a keyword query, we can obtain a matrix of features for the query keywords using the term-pairs in W. The matrix represents a space of search intentions (i.e., query candidates) of the original query w.r.t. the XML data. Therefore, our first problem is to select a subset of query candidates, which has the highest probability of interpreting the contexts of original query. In this work, the selection of query candidates are based on an approximate sample at the entity level of XML data.

Consider query q ={database, query} over DBLP XML dataset again. Its corresponding matrix can be constructed from Table 1. Table 3 shows the mutual information score for the query keywords in q. Each combination of the feature terms in matrix may represent a search intention with the specific semantics. For example, the combination "query *expansion* database *systems* " targets to search the publications discussing the problem of query expansion in the area of database systems, e.g., one of the works, "*query expansion for information retrieval*" published in *Encyclopedia of Database Systems* in 2009, will be returned. If we replace the feature term "systems" with "relational", then the generated query will be changed to search specific publications of query expansion over relational database, in which the returned results are empty because no work is reported to the problem over relational database in DBLP dataset.

TABLE 3
Mutual information score w.r.t. terms in q

database	*system*	*relational*	*protein*	*distributed*	*oriented*
	7.06	3.84	2.79	2.25	2.06
Mutual score (10^{-4})	*image*	*sequence*	*search*	*model*	*large*
	1.73	1.31	1.1	1.04	1.02
query	*language*	*expansion*	*optimization*	*evaluation*	*complexity*
	3.63	2.97	2.3	1.71	1.41
Mutual score (10^{-4})	*log*	*efficient*	*distributed*	*semantic*	*translation*
	1.17	1.03	0.99	0.86	0.70

2.2 Keyword Search Diversification Model

In our model, we not only consider the probability of new generated queries, i.e., *relevance*, we also take into account their new and distinct results, i.e., *novelty*. To embody the relevance and novelty of keyword search together, two criteria should be satisfied: (1) the generated query q_{new} has the maximal probability to interpret the contexts of original query q with regards to the data to be searched; and (2) the generated query q_{new} has a maximal difference from the previously generated query set Q. Therefore, we have the aggregated scoring function.

$$score(q_{new}) = Prob(q_{new}|q, T) * DIF(q_{new}, Q, T) \quad (2)$$

where $Prob(q_{new}|q, T)$ represents the probability that q_{new} is the search intention when the original query q is issued over the data T; $DIF(q_{new}, Q, T)$ represents the percentage of results that are produced by q_{new}, but not by any previously generated query in Q.

2.2.1 Evaluating the Probabilistic Relevance of an Intended Query Suggestion w.r.t. the Original Query

Based on the Bayes Theorem, we have

$$Prob(q_{new}|q, T) = \frac{Prob(q|q_{new},T)*Prob(q_{new}|T)}{Prob(q|T)} \quad (3)$$

where $Prob(q|q_{new}, T)$ models the likelihood of generating the observed query q while the intended query is actually q_{new}; and $Prob(q_{new}|T)$ is the query generation probability given the XML data T.

The likelihood value $Prob(q|q_{new}, T)$ can be measured by computing the probability of the original query q that is observed in the context of the features in q_{new}. Given an original query $q = \{k_i\}_{1\le i\le n}$ and a generated new query candidate $q_{new} = \{s_i\}_{1\le i\le n}$

where k_i is a query keyword in q, s_i is a segment that consists of the query keyword k_i and one of its features f_{ij_i} in W, and $1 \leq j_i \leq m$. Here, we assume that for each query keyword, only top m most relevant features will be retrieved from W to generate new query candidates. To deal with multi-keyword queries, we make the independence assumption on the probability that f_{ij_i} is the intended feature of the query keyword k_i. That is,

$$Prob(q|q_{new},T) = \prod_{k_i \in q, f_{ij_i} \in q_{new}} Prob(k_i|f_{ij_i},T) \tag{4}$$

According to the statistical sample information, the intent of a keyword can be inferred from the occurrences of the keyword and its correlated terms in the data. Thus, we can compute the probability $Prob(k_i|f_{ij_i},T)$ of interpreting a keyword k_i into a search intent f_{ij_i} as follows.

$$\begin{aligned} Prob(k_i|f_{ij_i},T) &= \frac{Prob(f_{ij_i}|k_i,T) * Prob(k_i,T)}{Prob(f_{ij_i},T)} \\ &= \frac{|R(\{k_i,f_{ij_i}\},T)|/|R(T)|}{|R(f_{ij_i},T)|/|R(T)|} \\ &= \frac{|R(s_i,T)|}{|R(f_{ij_i},T)|} \end{aligned} \tag{5}$$

where $s_i = \{k_i, f_{ij_i}\}$.

Consider a query q ={database, query} and a query candidate q_{new}={database system; query expansion}. $Prob$(q|q_{new}, T) shows the probability of a publication that addresses the problem of "database query" regarding the context of "system and expansion", which can be computed by $\frac{|R(\{\text{database system}\},T)|}{|R(\text{system},T)|} * \frac{|R(\{\text{query expansion}\},T)|}{|R(\text{expansion},T)|}$. Here, $|R(\{\text{database system}\},T)|$ represents the number of keyword search results of query {*database system*} over the data T. $|R(\text{system},T)|$ represents the number of keyword search results of running query *system* over the data T, but the number can be obtained without running query *system* because it is equal to the size of keyword node list of "system" over T. Similarly, we can calculate the values of $|R(\{\text{query expansion}\},T)|$ and $|R(\text{expansion},T)|$, respectively. In this work, we adopt the widely accepted semantics - *Smallest Lowest Common Ancestor* SLCA to model XML keyword search results.

Given the XML data T, the query generation probability of q_{new} can be calculated by the following equation.

$$Prob(q_{new}|T) = \frac{|R(q_{new},T)|}{|R(T)|} = \frac{|\bigcap_{s_i \in q_{new}} R(s_i,T)|}{|R(T)|} \tag{6}$$

where $\bigcap_{s_i \in q_{new}} R(s_i,T)$ represents the set of SLCA results by merging the node lists $R(s_i,T)$ for $s_i \in q_{new}$ using the XRank algorithm in [4] that is a popular method to compute the SLCA results by traversing the XML data only once.

Given an original query and the data, the value $\frac{1}{Prob(q|T)}$ is a relatively unchanged value with regards to different generated query candidates. Therefore, Equation 3 can be rewritten as follows.

$$Prob(q_{new}|q,T) = \gamma * (\prod \frac{R(s_i,T)}{R(f_{ij_i},T)}) * \frac{|\bigcap R(s_i,T)|}{|R(T)|} \tag{7}$$

where $k_i \in q$, $s_i \in q_{new}$, $f_{ij_i} \in s_i$ and $\gamma = \frac{1}{Prob(q|T)}$ can be any value in (0,1] because it does not affect the expanded query candidates w.r.t. an original keyword query q and data T.

Although the above equation can model the probabilities of generated query candidates (i.e., the relevance between these query candidates and the original query w.r.t. the data), different query candidates may have overlapped result sets. Therefore, we should also take into account the novelty of results of these query candidates.

2.2.2 *Evaluating the Probabilistic Novelty of an Intended Query w.r.t. the Original Query*

As we know, an important property of SLCA semantics is exclusivity, i.e., if a node is taken as an SLCA result, then its ancestor nodes cannot become SLCA results. Due to this exclusive property, the process of evaluating the novelty for a newly generated query candidate q_{new} depends on the evaluation of the other previously generated query candidates Q. As such, the novelty $DIF(q_{new},Q,T)$ of q_{new} against Q can be calculated as follows:

$$DIF(q_{new},Q,T) = \frac{|\{v_x | v_x \in R(q_{new},T) \wedge \nexists v_y \in \{\bigcup_{q' \in Q} R(q',T)\} \wedge v_x \leq v_y\}|}{|R(q_{new},T) \bigcup \{\bigcup_{q' \in Q} R(q',T)\}|} \tag{8}$$

where $R(q_{new},T)$ represents the set of SLCA results generated by q_{new}; $\bigcup_{q' \in Q} R(q',T)$ represents the set of SLCA results generated by queries in Q, which excludes the duplicate and ancestor nodes; $v_x \leq v_y$ means that v_x is a duplicate of v_y for "=", or v_x is an ancestor of v_y for "$<$"; $R(q_{new},T) \bigcup \{\bigcup_{q' \in Q} R(q',T)\}$ is an SLCA result set that satisfies with the exclusive property.

By doing this, we can avoid presenting repeated or overlapped SLCA results to users. In other words, the consideration of novelty allows us to incrementally refine the diversified results into more specific ones when we incrementally deal with more query candidates.

As we know our problem is to find top k qualified query candidates and their relevant SLCA results. To do this, we can compute the absolute score of the search intention for each generated query candidate. However, for reducing the computational cost, an alternative way is to calculate the relative scores of queries. Therefore, we have the following equation transformation. After we substitute Equation 7 and Equation 8 into Equation 2, we have the final equation.

$$\begin{aligned} score(q_{new}) &= \gamma * \prod(\tfrac{R(s_i,T)}{R(f_{ij_i},T)}) * \tfrac{|\bigcap R(s_i,T)|}{|R(T)|} * \\ &\tfrac{|\{v_x|v_x\in R(q_{new},T)\wedge\nexists v_y\in\{\bigcup_{q'\in Q}R(q',T)\}\wedge v_x\leq v_y\}|}{|R(q_{new},T)\bigcup\{\bigcup_{q'\in Q}R(q',T)\}|} \\ &= \tfrac{\gamma}{|R(T)|} * \prod(\tfrac{R(s_i,T)}{R(f_{ij_i},T)}) * |\bigcap R(s_i,T)| * \\ &\tfrac{|\{v_x|v_x\in R(q_{new},T)\wedge\nexists v_y\in\{\bigcup_{q'\in Q}R(q',T)\}\wedge v_x\leq v_y\}|}{|R(q_{new},T)\bigcup\{\bigcup_{q'\in Q}R(q',T)\}|} \\ &\mapsto \prod(\tfrac{R(s_i,T)}{R(f_{ij_i},T)}) * |\bigcap R(s_i,T)| * \\ &\tfrac{|\{v_x|v_x\in R(q_{new},T)\wedge\nexists v_y\in\{\bigcup_{q'\in Q}R(q',T)\}\wedge v_x\leq v_y\}|}{|R(q_{new},T)\bigcup\{\bigcup_{q'\in Q}R(q',T)\}|} \end{aligned} \tag{9}$$

where $s_i \in q_{new}$, $s_i = k_i \cup f_{ij_i}$, $k_i \in q$, $q' \in Q$ and the symbol $\mapsto$ represents the left side of the equation *depends on* the right side of the equation because the value $\frac{\gamma}{|R(T)|}$ keeps unchanged for calculating the diversification scores of different search intentions.

Property 1: (Upper Bound of Top-k Query Suggestions) Suppose we have found k query suggestions $Q = \{q_1, ..., q_k\}$ for an original keyword query q, and for any $q_x \in Q$. If we have $score(q_x) \geq \prod(\frac{R(s_i,T)}{R(f_{ij_i},T)}) * min\{|L_{k_i}|\}$ where $R(s_i,T)$ (or $R(f_{ij_i},T)$) is the result set of evaluating s_i (or f_{ij_i}) in the entity sample space of T and $min\{|L_{k_i}|\}$ is the size of the shortest keyword node list for keywords k_i in q, then the evaluation algorithm can be safely terminated, i.e., we can guarantee the current k query suggestions will be the top-k qualified ones w.r.t. q in T.

Proof: From Equation 9, we can see that Property 1 can be proved if the following inequation holds: $min\{|L_{k_i}|\} \geq |\bigcap R(s_i,T)| * \frac{|\{v_x|v_x\in R(q_{new},T)\wedge\nexists v_y\in\{\bigcup_{q'\in Q}R(q',T)\}\wedge v_x\leq v_y\}|}{|R(q_{new},T)\bigcup\{\bigcup_{q'\in Q}R(q',T)\}|}$.

Since $s_i \supseteq k_i \cup f_{ij_i}$, it must have $min\{L_{s_i}\} \leq min\{|L_{k_i}|\}$. In addition, we also know $|\bigcap R(s_i,T)|$ should be bounded by the minimal size ($min\{L_{s_i}\}$) of keyword nodes that contain s_i in T, i.e., $|\bigcap R(s_i,T)| \leq min\{L_{s_i}\}$. As such, we have $|\bigcap R(s_i,T)| \leq min\{|L_{k_i}|\}$. Because of $\frac{|\{v_x|v_x\in R(q_{new},T)\wedge\nexists v_y\in\{\bigcup_{q'\in Q}R(q',T)\}\wedge v_x\leq v_y\}|}{|R(q_{new},T)\bigcup\{\bigcup_{q'\in Q}R(q',T)\}|} \leq 1$, so we can derive that $|\bigcap R(s_i,T)| * \frac{|\{v_x|v_x\in R(q_{new},T)\wedge\nexists v_y\in\{\bigcup_{q'\in Q}R(q',T)\}\wedge v_x\leq v_y\}|}{|R(q_{new},T)\bigcup\{\bigcup_{q'\in Q}R(q',T)\}|} \leq min\{|L_{k_i}|\}$. Therefore, the property can be proved. □

Since we incrementally generate and assess the intended query suggestions with the high probabilistic relevance values, Property 1 guarantees we can skip the unqualified query suggestions and terminate our algorithms as early as possible when the top-k qualified ones have been identified.

3 Extracting Feature Terms

To address the problem of extracting meaningful feature terms w.r.t. an original keyword query, there are two relevant works [17], [18]. In [17], the authors proposed a solution of producing top-k interesting and meaningful expansions to a keyword query by extracting k additional words with high "interestingness" values. The expanded queries can be used to search more specific documents. The interestingness is formalized with the notion of *surprise* [19], [20], [21]. In [18], the authors proposed efficient algorithms to identify keyword clusters in large collections of blog posts for specific temporal intervals. Our work integrates both of their ideas together: we first measure the correlation of each pair of terms using our mutual information model in Equation 1, which is a simple surprise metric; and then we build term correlated graph that maintains all the terms and their correlation values. Different from [17], [18], our work utilizes entity-based sample information to build a correlated graph with high precision for XML data.

In order to efficiently measure the correlation of a pair of terms, we use a statistic method to measure how much the co-occurrences of a pair of terms deviate from the independence assumption where the entity nodes (e.g., the nodes with the "*" node types in XML DTD) are taken as a sample space. For instance, given a pair of terms x and y, their mutual information score can be calculated based on Equation 1 where $Prob(x,T)$ (or $Prob(y,T)$) is the value of dividing the number of entities containing x (or y) by the total entity size of the sample space; $Prob(\{x,y\},T)$ is the value of dividing the number of entities containing both x and y by the total entity size of the sample space.

In this work, we build a term correlated graph offline, that is we precompute it before processing queries. The correlation values among terms are also recorded in the graph, which is used to generate the term-feature dictionary W. During the XML data tree traversal, we first extract the meaningful text information from the entity nodes in XML data. Here, we would like to filter out the stop words. And then we produce a set of term-pairs by scanning the extracted text. After that, all the generated term-pairs will be recorded in the term correlated graph. In the procedure of building correlation graph, we also record the count of each term-pair to be generated from different entity nodes. As such, after the XML data tree is traversed completely, we can compute the mutual information score for each term-pair based on Equation 1. To reduce the size of correlation graph, the term-pairs with their correlation lower than a threshold can be filtered out. Based on the off-line built graph, we can on-the-fly select the top-m distinct terms as its features for each given query keyword.

4 Keyword Search Diversification Algorithms

In this section, we first propose a baseline algorithm to retrieve the diversified keyword search results. And then, two anchor-based pruning algorithms are designed to improve the efficiency of the keyword search diversification by utilizing the intermediate results.

Algorithm 1 Baseline Algorithm

```
input: a query $q$ with $n$ keywords, XML data $T$ and its term correlated graph $G$
output: Top-$k$ search intentions $Q$ and the whole result set $\Phi$
1: $M_{m \times n}$ = getFeatureTerms($q$, $G$);
2: while ($q_{new}$ = GenerateNewQuery($M_{m \times n}$)) $\neq$ null do
3:     $\phi$ = null and $prob\_s\_k$ = 1;
4:     $l_{i_x j_y}$ = getNodeList($s_{i_x j_y}$, $T$) for $s_{i_x j_y} \in q_{new} \wedge 1 \leq i_x \leq m \wedge 1 \leq j_y \leq n$;
5:     $prob\_s\_k = \prod_{f_{i_x j_y} \in s_{i_x j_y} \in q_{new}} (\frac{|l_{i_x j_y}|}{getNodeSize(f_{i_x j_y}, T)})$;
6:     $\phi$ = ComputeSLCA($\{l_{i_x j_y}\}$);
7:     $prob\_q\_new = prob\_s\_k * |\phi|$;
8:     if $\Phi$ is empty then
9:         $score(q_{new}) = prob\_q\_new$;
10:    else
11:        for all Result candidates $r_x \in \phi$ do
12:            for all Result candidates $r_y \in \Phi$ do
13:                if $r_x == r_y$ or $r_x$ is an ancestor of $r_y$ then
14:                    $\phi.remove(r_x)$;
15:                else if $r_x$ is a descendant of $r_y$ then
16:                    $\Phi.remove(r_y)$;
17:        $score(q_{new}) = prob\_q\_new * |\phi| * \frac{|\phi|}{|\phi|+|\Phi|}$;
18:    if $|Q| < k$ then
19:        put $q_{new} : score(q_{new})$ into $Q$;
20:        put $q_{new} : \phi$ into $\Phi$;
21:    else if $score(q_{new}) > score(\{q'_{new} \in Q\})$ then
22:        replace $q'_{new} : score(q'_{new})$ with $q_{new} : score(q_{new})$;
23:        $\Phi.remove(q'_{new})$;
24: return $Q$ and result set $\Phi$;
```

4.1 Baseline Solution

Given a keyword query, the intuitive idea of the baseline algorithm is to first retrieve the relevant feature terms with high mutual scores from the term correlated graph of the XML data T; then generate list of query candidates that are sorted in the descending order of total mutual scores; and finally compute the SLCAs as keyword search results for each query candidate and measure its diversification score. As such, the top-k diversified query candidates and their corresponding results can be chosen and returned.

Different from traditional XML keyword search, our work needs to evaluate multiple intended query candidates and generate a whole result set, in which the results should be diversified and distinct from each other. Therefore, we have to detect and remove the duplicated or ancestor SLCA results that have been seen when we obtain new generated results.

The detailed procedure has been shown in Algorithm 1. Given a keyword query q with n keywords, we first load its pre-computed relevant feature terms from the term correlated graph G of XML data T, which is used to construct a matrix $M_{m \times n}$ as shown in Line 1. And then, we generate a new query candidate q_{new} from the matrix $M_{m \times n}$ by calling the function GenerateNewQuery() as shown in Line 2. The generation of new query candidates are in the descending order of their mutual information scores. Line 3-Line 7 show the procedure of computing $Prob(q|q_{new}, T)$. To compute the SLCA results of q_{new}, we need to retrieve the precomputed node lists of the keyword-feature term pairs in q_{new} from T by $getNodeList(s_{i_x j_y}, T)$. Based on the retrieved node lists, we can compute the likelihood of generating the observed query q while the intended query is actually q_{new}, i.e., $Prob(q|q_{new}, T) = \prod_{f_{i_x j_y} \in s_{i_x j_y} \in q_{new}} (|l_{i_x j_y}| / getNodeSize\ (f_{i_x j_y},\ T))$ using Equation 5 where $getNodeSize(f_{i_x j_y}, T)$ can be quickly obtained based on the precomputed statistic information of T. After that, we can call for the function ComputeSLCA() that can be implemented using any existing XML keyword search method. In Line 8 - Line 16, we compare the SLCA results of the current query and the previous queries in order to obtain the distinct and diversified SLCA results. At Line 17, we compute the final score of q_{new} as a diversified query candidate w.r.t. the previously generated query candidates in Q. At last, we compare the new query and the previously generated query candidates and replace the unqualified ones in Q, which is shown in Line 18 - Line 23. After processing all the possible query candidates, we can return the top k generated query candidates with their SLCA results.

In the worst case, all the possibe queries in the matrix may have the possibility of being chosen as the top-k qualified query candidates. In this worst case, the complexity of the algorithm is O($m^{|q|} * L_1 \sum_{i=2}^{|q|} logL_i$) where L_1 is the shortest node list of any generated query, $|q|$ is the number of original query keywords and m is the size of selected features for each query keyword. In practice, the complexity of the algorithm can be reduced by reducing the number m of feature terms, which can be used to bound the number (i.e., reducing the value of $m^{|q|}$) of generated query candidates.

4.2 Anchor-based Pruning Solution

By analyzing the baseline solution, we can find that the main cost of this solution is spent on computing SLCA results and removing unqualified SLCA results from the newly and previously generated result sets. To reduce the computational cost, we are motivated to design an anchor-based pruning solution, which can avoid the unnecessary computational cost of unqualified SLCA results (i.e., duplicates and ancestors). In this subsection, we first analyze the interrelationships between the intermediate SLCA candidates that have been already computed for the generated query candidates Q and the nodes that will be merged for answering the newly generated query candidate q_{new}. And then, we will propose the detailed description and algorithm of the anchor-based pruning solution.

4.2.1 *Properties of Computing diversified SLCAs*

Definition 1: (Anchor nodes) Given a set of query candidates Q that have been already processed and a new query candidate q_{new}, the generated SLCA results Φ of Q can be taken as the anchors for efficiently computing the SLCA results of q_{new} by partitioning the keyword nodes of q_{new}.

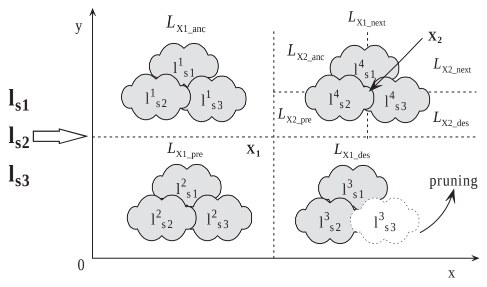


Fig. 1. The usability of anchor nodes

Example 2: Figure 1 shows the usability of anchor nodes for computing SLCA results. Consider two SLCA results X_1 and X_2 (assume X_1 precedes X_2) for the current query set Q. For the next query $q_{new} = \{s_1, s_2, s_3\}$ and its keyword instance lists L = $\{l_{s1},\ l_{s2},\ l_{s3}\}$, the keyword instances in L will be divided into four areas by the anchor X_1: (1) $L_{X_1\_anc}$ in which all the keyword nodes are the ancestor of X_1 so $L_{X_1\_anc}$ cannot generate new and distinct SLCA results; (2) $L_{X_1\_pre}$ in which all the keyword nodes are the previous siblings of X_1 so we may generate new SLCA results if the results are still bounded in the area; (3) $L_{X_1\_des}$ in which all the keyword nodes are the descendant of X_1 so it may produce new SLCA results that will replace X_1; and (4) $L_{X_1\_next}$ in which all the keyword nodes are the next siblings of X_1 so it may produce new results, but it may be further divided by the anchor X_2. If there is no intersection of all keyword node lists in an area, then the nodes in this area can be pruned directly, e.g., l^3_{s1} and l^3_{s2} can be pruned without computation if l^3_{s3} is empty in L_{x1_des}. Similarly, we can process $L_{X_2\_pre}$, $L_{X_2\_des}$ and $L_{X_2\_next}$. After that, a set of new and distinct SLCA results can be obtained with regards to the new query set $Q \bigcup q_{new}$.

Theorem 1: (Single Anchor) Given an anchor node v_a and a new query candidate $q_{new} = \{s_1, s_2, ..., s_n\}$, its keyword node lists $L = \{l_{s_1}, l_{s_2}, ...,\ l_{s_n}\}$ can be divided into four areas to be anchored by v_a, i.e., the keyword nodes that are the ancestors of v_a, denoted as $L_{v_a\_anc}$; the keyword nodes that are the previous siblings of v_a, denoted as $L_{v_a\_pre}$; the keyword nodes that are the descendants of v_a, denoted as $L_{v_a\_des}$; and the keyword nodes that are the next siblings of v_a, denoted as $L_{v_a\_next}$. We have that $L_{v_a\_anc}$ does not generate any new result; each of the other three areas may generate new and distinct SLCA results individually; no new and distinct SLCA results can be generated accross the areas.

Proof: For the keyword nodes in the ancestor area, all of them are the ancestors of the anchor node v_a that is an SLCA node w.r.t. Q. All these keyword nodes in the area can be directly discarded due to the exclusive property of SLCA semantics. For the keyword nodes in the area $L_{v_a\_pre}$ (or $L_{v_a\_next}$), we need to compute their SLCA candidates. But the computation of SLCA candidates can be bounded by the preorder (or postorder) traversal number of v_a where we assume the range encoding scheme is used in this work. For the keyword nodes in $L_{v_a\_des}$, if they can produce SLCA candidates that are bounded in the range of v_a, then these candidates will be taken as the new and distinct results while v_a will be removed from result set.

The last is to certify that the keyword nodes across any two areas cannot produce new and distinct SLCA results. This is because the only possible SLCA candidates they may produce are the node v_a and its ancestor nodes. But these nodes cannot become new and distinct SLCA results due to the anchor node v_a. For instance, If we can select some keyword nodes from $L_{v_a\_pre}$ and some keyword nodes from $L_{v_a\_des}$, then their corresponding SLCA candidate must be the ancestor of v_a, which cannot become the new and distinct SLCA result due to the exclusive property of SLCA semantics. Similarly, no new result can be generated if we select some keyword nodes from the other two areas $L_{v_a\_pre}$ and $L_{v_a\_next}$ (or $L_{v_a\_des}$ and $L_{v_a\_next}$).

□

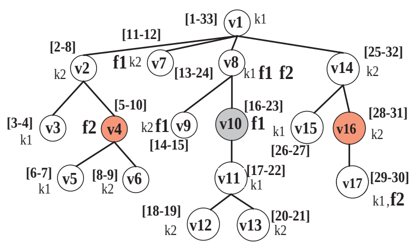


Fig. 2. Demonstrate usability of anchor nodes in Tree

Example 3: Figure 2 gives a small tree that contains four different terms k_1, k_2, f_1 and f_2. And the encoding ranges are attached to all the nodes. Assume f_1 and f_2 are two feature terms of a given keyword query $\{k_1, k_2\}$; and f_1 has higher correlation with the query than f_2. After the first intended query $\{k_1, k_2, f_1\}$ is evaluated, the node v_{10} will be the SLCA result. When we process the next intended query $\{k_1, k_2, f_2\}$, the node v_{10} will be an anchor node to partition the keyword nodes into four parts by using its encoding range $[16 - 23]$: L_{pre}=$\{v_2, v_3, v_4, v_5, v_6, v_7, v_9\}$ in which each node's postorder number is less than 16; L_{next}=$\{v_{14}, v_{15}, v_{16}, v_{17}\}$ in which each node's preorder number is larger than 23; L_{anc}=$\{v_1, v_8\}$ in which each node's preorder number is less than 16 while its postorder number is larger than 23; and the rest nodes belong to L_{des}=$\{v_{11}, v_{12}, v_{13}\}$. Although v_8 contains f_2, it cannot contribute more specific results than the anchor node v_{10} due to exclusive property of SLCA semantics. So we only need to evaluate $\{k_1, k_2, f_2\}$ over the three areas L_{pre}, L_{next} and L_{des} individually. Since no node contains f_2 in L_{des}, all the keyword nodes in L_{des} can be skipped. At last, the nodes v_4 and v_{16} will be two additional SLCA results. If we have the following intended queries to be processed, then their relevant keyword nodes can be partitioned by the current three anchor nodes v_4, v_{10}, v_{16}. Generally, the number of keyword nodes to be skipped increases with the increase of number of anchor nodes to be selected, which can significantly reduce the time cost of algorithms as shown in experiments.

Theorem 2: (Multiple Anchors) Given multiple anchor nodes V_a and a new query candidate $q_{new} = \{s_1, s_2, ..., s_n\}$, its keyword node lists $L = \{l_{s_1}, l_{s_2}, ..., l_{s_n}\}$ can be maximally divided into $(3 * |V_a| + 1)$ areas to be anchored by the nodes in V_a. Only the nodes in $(2 * |V_a| + 1)$ areas can generate new SLCA candidates individually and the nodes in the $(2 * |V_a| + 1)$ areas are independent to compute SLCA candidates.

Proof: Let V_a = (x_1, ..., x_i, ..., x_j, ..., $x_{|V_a|}$), where x_i precedes x_j for $i < j$. We first partition the space into four areas using x_1, i.e., $L_{x_1\_anc}$, $L_{x_1\_pre}$, $L_{x_1\_des}$ and $L_{x_1\_next}$. For $L_{x_1\_next}$, we partition it further using x_2 into four areas. We repeat this process until we partition the area $x_{(|V_a|-1)\_next}$ into four areas $x_{|V_a|\_anc}$, $L_{x_{|V_a|}\_pre}$, $L_{x_{|V_a|}\_des}$ and $x_{|v_a|\_next}$ by using $x_{|V_a|}$. Obviously the total number of partitioned areas is $3 * |V_a| + 1$. As the ancestor areas do not contribute to SLCA results, we end up with $2 * |V_a| + 1$ areas that need to be processed.

Theorem 1 guarantees that any keyword nodes across different areas cannot contribute new and distinct SLCA results. This is because any possible result candidate to be generated by these keyword nodes must be the ancestor of one current SLCA result at least. Due to the exclusive property of SLCA semantics, no new and distinct SLCA results can be produced across different areas. □

Property 2: Consider the $(2 * |V_a| + 1)$ effective areas to be anchored by the nodes in V_a. If $\exists s_i \in q_{new}$ and none of its instances appear in an area, then the area can be pruned because it cannot generate SLCA candidates of q_{new}.

The reason is that any area that can possibly generate new results should contain at least one keyword matched instance for each keyword in the query based on the SLCA semantics. Therefore, if an area contains no keyword instance, then the area can be removed definitely without any computation.

Following Example 3, if we have the third intended query that needs to be evaluated and v_4, v_{10} and v_{16} have been selected as anchor nodes, then we have ten areas of keyword nodes: $L_{v4-pre} = \{v_3\}$; $L_{v4-anc} = \{v_1, v_2\}$; $L_{v4-des} = \{v_5, v_6\}$; $L_{v4-next/v10-pre} = \{v_7, v_9\}$; $L_{v10-anc} = \{v_1, v_8\}$; $L_{v10-des} = \{v_{11}, v_{12}, v13\}$; $L_{v10-next/v16-pre} = \{v_{15}\}$; $L_{v16-anc} = \{v_1, v_{14}\}$; $L_{v16-des} = \{v_{17}\}$; and $L_{v16-next} = NIL$; Three of them can be directly discarded due to exclusive property of SLCA semantics, i.e., L_{v4-anc}, $L_{v10-anc}$ and $L_{v16-anc}$. $L_{v16-next}$ can also be removed due to empty set. The rest can be first filtered if they do not contain all the keywords in the third intended query before computing the possible new and distinct SLCA results.

Algorithm 2 Anchor-based Pruning Algorithm

```
input: a query $q$ with $n$ keywords, XML data $T$ and its term correlated graph $G$
output: Top-$k$ query intentions $Q$ and the whole result set $\Phi$
1: $M_{m \times n}$ = getFeatureTerms($q$, $G$);
2: while $q_{new}$ = GenerateNewQuery($M_{m \times n}$) $\neq$ null do
3:   Line 3-Line 5 in Algorithm 1;
4:   if $\Phi$ is not empty then
5:     for all $v_{anchor} \in \Phi$ do
6:       get $l_{i_x j_y\_pre}$, $l_{i_x j_y\_des}$, and $l_{i_x j_y\_next}$ by calling for Partition($l_{i_x j_y}$, $v_{anchor}$);
7:       if $\forall l_{i_x j_y\_pre} \neq$ null then
8:         $\phi'$ = ComputeSLCA($\{l_{i_x j_y\_pre}\}$, $v_{anchor}$);
9:       if $\forall l_{i_x j_y\_des} \neq$ null then
10:        $\phi''$ = ComputeSLCA($\{l_{i_x j_y\_des}\}$, $v_{anchor}$);
11:      $\phi$ += $\phi'$ + $\phi''$;
12:      if $\phi'' \neq$ null then
13:        $\Phi.remove(v_{anchor})$;
14:      if $\exists l_{i_x j_y\_next}$ =null then
15:        Break the FOR-Loop;
16:      $l_{i_x j_y} = l_{i_x j_y\_next}$ for $1 \leq i_x \leq m \wedge 1 \leq j_y \leq n$;
17:  else
18:    $\phi$ = ComputeSLCA($\{l_{i_x j_y}\}$);
19:  $score(q_{new})$ = $prob\_q\_new$ * $|\phi|$* $\frac{|\phi|}{|\Phi|+|\phi|}$;
20:  Line 18-Line 23 in Algorithm 1;
21: return $Q$ and result set $\Phi$;
```

4.2.2 The Anchor-based Pruning Algorithm

Motivated by the properties of computing diversified SLCAs, we design the anchor-based pruning algorithm. The basic idea is described as follows. We generate the first new query and compute its corresponding SLCA candidates as a start point. When the next new query is generated, we can use the intermediate results of the previously generated queries to prune the unnecessary nodes according to the above theorems and property. By doing this, we only generate the distinct SLCA candidates every time. That is to say, unlike the baseline algorithm, the diversified results can be computed directly without further comparison.

The detailed procedure is shown in Algorithm 2. Similar to the baseline algorithm, we need to construct the matrix of feature terms, retrieve their corresponding node lists where the node lists can be maintained using R-tree index. And then, we can calculate the likelihood of generating the observed query q when the issued query is q_{new}. Different from the baseline algorithm, we utilize the intermediate SLCA results of previously generated queries as the anchors to efficiently compute the new SLCA results for the following queries. For the first generated query, we can compute the SLCA results using any existing XML keyword search method as the baseline algorithm does, shown in Line 18. Here, we use stack-based method to implement the function ComputeSLCA().

The results of the first query will be taken as anchors to prune the node lists of the next query for reducing its evaluation cost in Line 5 - Line 16. Given an anchor node v_{anchor}, for each node list $l_{i_xj_y}$ of a query keyword in the current new query, we may get three effective node lists $l_{i_xj_y\_pre}$, $l_{i_xj_y\_des}$ and $l_{i_xj_y\_next}$ using R-tree index by calling for the function Partition(). If a node list is empty, e.g., $l_{i_xj_y\_pre}$ =null, then we don't need to get the node lists for the other query keywords in the same area, e.g., in the left-bottom area of v_{anchor}. This is because it cannot produce any SLCA candidates at all. Consequently, the nodes in this area cannot generate new and distinct SLCA results. If all the node lists have at least one node in the same area, then we compute the SLCA results by the function ComputeSLCA(), e.g., ComputeSLCA($\{l_{i_xj_y\_des}\}$, v_{anchor}) that merges the nodes in $\{l_{i_xj_y\_des}\}$. If the SLCA results are the descendant of v_{anchor}, then they will be recorded as new distinct results and v_{anchor} will be removed from the temporary result set. Through Line 19, we can obtain the final score of the current query without comparing the SLCA results of ϕ with that of Φ. At last, we need to record the score and the results of the new query into Q and Φ, respectively. After all the necessary queries are computed, the top-k diversified queries and their results will be returned.

4.3 Anchor-based Parallel Sharing Solution

Although the anchor-based pruning algorithm can avoid unnecessary computation cost of the baseline algorithm, it can be further improved by exploiting the parallelism of keyword search diversification and reducing the repeated scanning of the same node lists.

4.3.1 Observations

According to the semantics of keyword search diversification, only the distinct SLCA results need to be returned to users. We have the following two observations.

Observation 1: Anchored by the SLCA result set V_a of previously processed query candidates Q, the keyword nodes of the next query candidate q_{new} can be classified into $2 * |V_a| + 1$ areas. According to Theorem 2, no new and distinct SLCA results can be generated across areas. Therefore, the $2*|V_a|+1$ areas of nodes can be processed independently, i.e., we can compute the SLCA results area by area. It can make the parallel keyword search diversification efficient without communication cost among processors.

Observation 2: Because there are term overlaps between the generated query candidates, the intermediate partial results of the previously processed query candidates may be reused for evaluating the following queries, by which the repeated computations can be avoided.

4.3.2 Anchor-based Parallel Sharing Algorithm

To make the parallel computing efficiently, we utilize the SLCA results of previous queries as the anchors to partition the node lists that need to be computed. By assigning areas to processors, no communication cost among the processors is required. Our proposed algorithm guarantees that the results generated by each processor are the SLCA results of the current query. In addition, we also take into account the shared partial matches among the generated query candidates, by which we can further improve the efficiency of the algorithm.

Different from the above two proposed algorithms, we first generate and analyse all the possible query candidates Q_{new}. Here, we use a vector V to maintain the shared query segments among the generated queries in Q_{new}. And then, we begin to process the first query like the above two algorithms. When the next query is coming, we will check its shared query segments and explore parallel computing to evaluate the query candidate.

To do this, we first check if the new query candidate q_{new} contains shared parts ψ in V. For each shared part in ψ, we need to check its processing status. If the status has already been set as "processed", then it says that the partial matches of the shared part have been computed before. In this condition, we only need to retrieve the partial matches from previously cached results. Otherwise, it says that we haven't computed the partial matches of the shared part before. We have to compute the partial matches from the original node lists of the shared segments. After that, the processing status will be set as "processed". And then, the node lists of the rest query segments will be processed. In this algorithm, we also explore parallel computing to improve the performance of query evaluation. At the beginning, we specify the maximal number of processors and the current processor's id (denoted as PID). And then, we distribute the nodes that need to be computed to the processor with PID in a round. When all the required nodes are reached at a processor, the processor will be activated to compute the SLCA results of q_{new} or the partial matches for the shared segments in q_{new}. After all active processors complete their SLCA computations, we get the final SLCA results of q_{new}. At last, we can calculate the score of q_{new} and compare its score with the previous ones. If its score is larger than that of one of the query candidates in Q, then the query candidate q_{new}, its score $score(q_{new})$ and its SLCA results will be recorded. After all the necessary queries are processed, we can return the top-k qualified and diversified query candidates and their corresponding SLCA results. The detailed algorithm is not provided due to the limited space.

5 Experiments

In this section, we show the extensive experimental results for evaluating the performance of our baseline algorithm (denoted as *b*aseline *e*valuation *BE*) and anchor-based algorithm (denoted as *a*nchor-based *e*valuation *AE*), which were implemented in Java and run on a 3.0GHz Intel Pentium 4 machine with 2GB RAM running Windows XP. For our anchor-based parallel sharing algorithm (denoted as *ASPE*), it was implemented using six computers, which can serve as six processors for parallel computation.

5.1 Dataset and Queries

We use a real dataset, DBLP [22] and a synthetic XML benchmark dataset XMark [23] for testing the proposed XML keyword search diversification model and our designed algorithms. The size of DBLP dataset is 971MB and the size of generated XMark dataset is 697MB. Compared with DBLP dataset, the synthetic XMark dataset has varied depths and complex data structures, but as we know, it does not contain clear semantic information due to its synthetic data. Therefore, we only use DBLP dataset to measure the effectiveness of our diversification model in this work.

For each XML dataset used, we selected some terms based on the following criteria: (1) a selected term should often appear in user-typed keyword queries; (2) a selected term should highlight different semantics when it co-occurs with feature terms in different contexts. Based on the criteria of selection, we chose some terms for each dataset as follows. For DBLP dataset, we selected *"database, query, programming, semantic, structure, network, domain, dynamic, parallel"*. And we chose *"brother, gentleman, look, free, king, gender, iron, purpose, honest, sense, metals, petty, shakespeare, weak, opposite"* for XMark dataset. By randomly combining the selected terms, we generated 14 sets of keyword queries for each dataset.

In addition, we also designed 10 specific DBLP keyword queries with clear search intentions as the ground truth query set. And then, we modified the clear keyword queries into vague queries by masking some keywords, e.g., *"$\underline{\text{query optimization}}$ parallel database"*, *"$\underline{\text{query language}}$ database"* and *"dynamic $\underline{\text{programming}}$ networks"* where the keywords with underlines are masked.

5.2 Effectiveness of Diversification Model

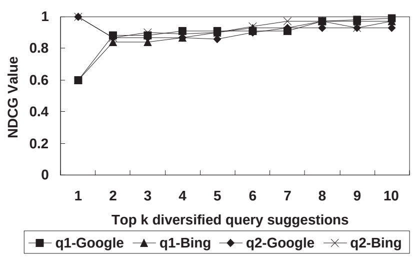
Three metrics will be applied to evaluate the effectiveness of our diversification model by assessing its relevance, diversity and usefulness.

5.2.1 Test Adapted-nDCG

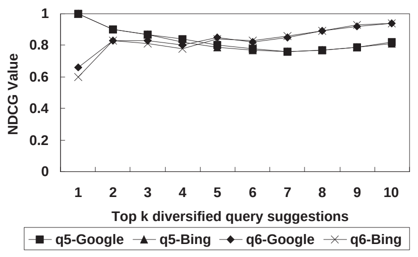
The normalized Discounted Cumulative Gain (nDCG) has established itself as the standard evaluation measure when graded relevance values are available [9], [24], [13]. The first step in the nDCG computation is the creation of a gain vector G. The gain $G[k]$ at rank k can be computed as the relevance of the result at this rank to the user's keyword query. The gain is discounted logarithmically with increasing rank, to penalize documents lower in the ranking, reflecting the additional user effort required to reach them. The discounted gain is accumulated over k to obtain the DCG value and normalized using the ideal gain at rank k to finally obtain the nDCG value. For the relevance-based retrieval, the ideal gain may be considered as an identical one for some users. However, for the diversification-based retrieval, the ideal gain cannot be considered as an identical one for users because users may prefer to see different results for a query. Therefore, we need adapt nDCG measure to test the effectiveness of query diversification model.

Since it is difficult to find the ground truth for query diversification, we carefully selected the ground truth for our tested keyword queries. Here, we employed human assessors to manually mark the ground truth. We invited three groups of users to do user study where each group has ten undergraduate volunteers and all these users share the similar scoring criteria, i.e., interestingness, meaningfulness, and novelty. In this paper, Google and Bing search engines are used to explore diversified search intentions as a standard diverse suggestion set for a given keyword query. To do this, the volunteers in the 1^{st} and 2^{nd} groups evaluated the original query using Google and Bing search engines in the domain of DBLP "informatics.unitrier.de/-ley/db/". By checking at most 100 relevant results, each user u_i can identify 10 diversified queries based on her/his satisfaction and give each diversified query q_{div} a satisfaction credit $Credit(u_i, q_{div})$ in the range [0-10]. By doing this, each group can produce a set of different diversified queries where the credit of each query is marked by its average score: $\frac{\sum_{i=1}^{10}(Credit(u_i,q_{div}))}{10}$ where $Credit(u_i, q_{div}) = 0$ if q_{div} does not appear in the list of 10 diversified queries of u_i. After that, the top-10 diversified queries with the highest scores can be selected for each group. As such, the 1^{st} and 2^{nd} groups can produce two independent ideal gain for a keyword query using Google and Bing search engines. Similarly, the 3^{rd} group of users marked the diversified query suggestions produced by our work, which is used to calculate the gain $G[k]$. Based on the formula $nDCG_k = \frac{DCG_k}{idealDCG_k}$, we can calculate the nDCG of each keyword query based on the statistical information supported by the three groups of users.

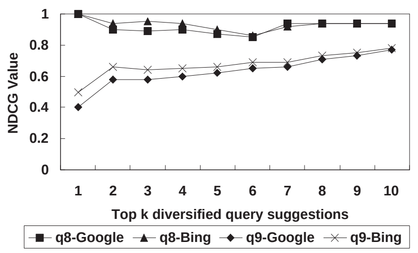
Figure 3(a)-Figure 3(d) show the nDCG values of keyword queries over DBLP dataset where only the



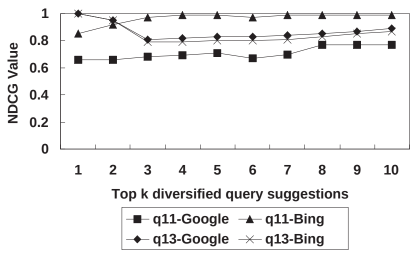
(a) q_1 and q_2 over DBLP



(b) q_5 and q_6 over DBLP



(c) q_8 and q_9 over DBLP



(d) q_{11} and q_{13} over DBLP

Fig. 3. nDCG Values vs. top-k query suggestions

top-10 diversified query suggestions are taken into account. From Figure 3(a) and Figure 3(b), we can see that nDCG values are no less than 0.8 when k $\geq$ 2, i.e., our proposed diversification model has 80% of the chance to approach the ground truth of ideal diversified query suggestion. Figure 3(c) shows for query set q_9, our diversification model only has 60% of chance to approach the ground truth of ideal diversified query suggestion. Figure 3(d) also shows the average nDCG value of the 11^{th} keyword query set q_{11} based on Google engine is near to 0.6. For this kind of queries, we need recommend more diversified query suggestions to improve the effectiveness of our diversification model.

5.2.2 Test Differentiation of Diversified Query Results

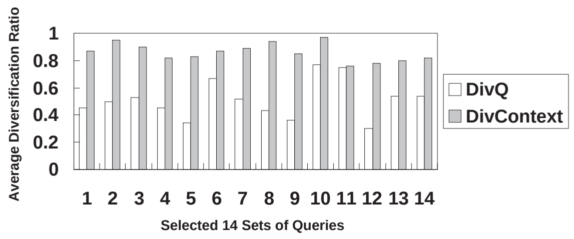


Fig. 4. Differentiation of Diversified Query Results

To show the effectiveness of diversification model, we also tested the differentiation between result sets by evaluating our suggested query candidates. This assessment is used to measure the possibility of diversified results based on the suggested diversified query candidates. To do this, we first selected the top-5 diversified query suggestions for each original keyword query. And then, we evaluated each diversified query suggestion over the DBLP dataset, respectively. From each list of relevant results, we chose its top-10 most relevant results. As such, we can have 50 results as a sample for each original keyword query to test its diversified possibility. The assessing metric can be summarized by the following equation:

$$\frac{\sum_{q_i,q_j \in top-kDS(q)}(1-\frac{\sum_{r_i \in R(q_i), r_j \in R(q_j)} Sim(r_i,r_j)}{|R(q_i)|*|R(q_j)|})}{C_k^2} \tag{10}$$

where $top-kDS(q)$ represents top-k diversified query suggestions; $R(q_i)$ represents the number of selected most relevant results for the diversified query suggestion q_i; $Sim(r_i, r_j)$ represents the similarity of two results r_i and r_j to be measured by n-gram model.

Similarly, we also tested the differentiation of diversified query results using DivQ in [13] and compared their experimental results with ours (denoted as DivContext). Figure 4 demonstrates the possibility of the different results with regards to the 14 selected sets of queries over DBLP dataset. The experimental results show that our diversification model provides significantly better differentiated results than DivQ. This is because structural difference in structured queries generated by DivQ cannot exactly reflect the differentiation of results to be generated by these structured queries. For the 6th, 10th and 11th sets of queries, DivQ can reach up to 80% because the three sets of queries contain a few specific keywords and their produced structured queries (i.e., query suggestions in DivQ) are with big different structures. From the experiments, we can see our context sensitive diversification model outperforms the DivQ probabilistic model because both the word context (i.e., term correlated graph) and structure (i.e., exclusive property of SLCA) are considered in our model.

5.2.3 Test Adapted Keyword Queries

This part of experiment builds on the general assumption that users often pick up their interested queries ranked at the top in the list of query suggestions. Therefore, in this section, we first selected 10 specific keyword queries with clear semantics and manually generated 10 corresponding vague keyword queries by removing some keywords from each specific keyword query. And then, we computed a list of diversified query suggestions for each vague keyword query. By checking the positions of the specific queries appearing in their corresponding query suggestion lists, we can test the usefulness of our diversification model. Here, we utilized a simple metric to quantify the value of usefulness. For instance, if the specific keyword query occurs at the top 1-5 positions in its corresponding suggestion list, then the usefulness of the diversification model is marked by 1. If it is located at the top 6-10 positions, then the usefulness is marked by 0.5. If it is located at the position range 11-20, then the usefulness is marked as 0.25. Otherwise, the suggestion is unusual for users because it is seldom for users to check more than 20 suggestions in a sequence, i.e., the usefulness is labeled as zero.

We also compared our work with the diversification model in [13]. Since [13] uses structured queries to represent query suggestions while ours uses a set of terms as query suggestions, we manually transformed their structured queries into the corresponding term sets based on the context of the structured queries. To make the comparison be fair, a structured query is considered as a specific keyword query if its corresponding term set contains all the keywords in the specific keyword query.

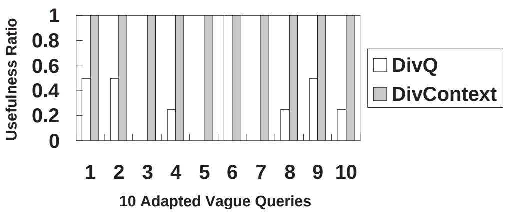
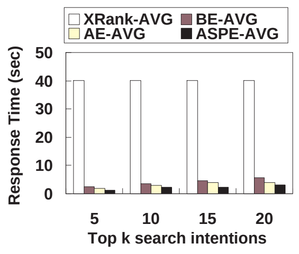


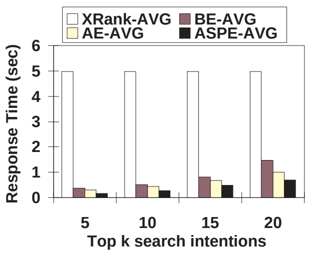
Fig. 5. Usefulness of Diversification Model

From Figure 5, we can see the specific keyword queries within top 5 query suggestions for the ten adapted vague queries by our method (DivContext). But for DivQ method, only one of them can be observed within the top 5 query suggestions, three of them found within top 10 suggestions, another three within top 20 suggestions, and the rest cannot get into the top 20 suggestions. From the experimental results, we can see the usefulness of a qualified suggestion model should consider the relevance of the query suggestions, as well as the novelty of the results to be generated by these query suggestions.

5.3 Efficiency of Diversification Approaches



(a) DBLP



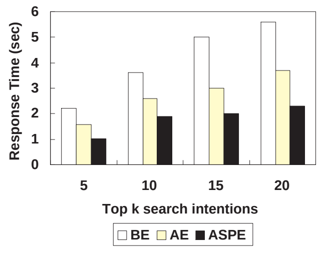
(b) XMark

Fig. 6. Average Time Comparison of Original Keyword Queries and Their Suggested Query Candidates

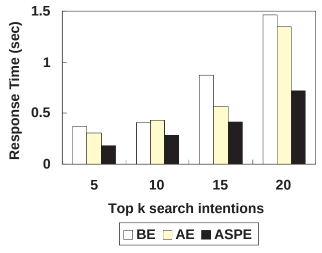
Figure 6(a) and Figure 6(b) show the average response time of computing all the SLCA results for the selected original keyword queries by using XRank [2], and directly computing their diversifications and corresponding SLCA results by using our proposed BE, AE and ASPE. According to the experimental results, we can see that our diversified algorithms just consumed 20% time of XRank to obtain top-k qualified diversified query suggestions and their diverse results where k is 5, 10, and 15, respectively. When k = 20, our algorithms consumed at most 30% time of XRank. Here, the time of XRank just counted the cost of computing SLCA results for the original keyword queries. If we further took into account the time of selecting diverse results, then our algorithms will be performed much better than XRank.

Especially, when the users are only interested in a few of search contexts, our context-based keyword query diversification algorithms can outperform the post-processing diversification approaches greatly. The main reason is that the computational cost is bounded by the minimal keyword node list. The specific feature terms with short node lists can be used to skip lots of keyword nodes in the other node lists. So the efficiency can be improved greatly.

From Figure 7(a), we can see BE consumed more time to answer a vague query with the increase of diversified query suggestions over DBLP dataset, i.e., take up to 5.7 seconds to answer a vague query with diversified results when the number of qualified suggestions is set as 20. However, AE and ASPE can do



(a) DBLP



(b) XMark

Fig. 7. Average Time Cost of Queries

that in about 3.5 seconds and 2 seconds, respectively. This is because lots of nodes can be skipped by anchor nodes without computation. Another reason is that when the number of suggestions is small, e.g., 5, we can quickly identify the qualified suggestions and safely terminate the evaluation with the guarantee of the upper bound. As such, the qualified suggestions and their diverse results can be output.

Figure 7(b) shows the time cost of evaluating the keyword queries over XMark dataset. Although the efficiency of BE is slower than that AE and ASPE, it can still finish each query evaluation in about 1.5 seconds. This is because (1) the size of keyword nodes is not as large as that of DBLP;(2) the keyword nodes are distributed evenly in the synthetic XMark dataset, which limits the performance of AE and ASPE.

6 Related Work

Diversifying results of document retrieval has been introduced [6], [7], [8], [9]. Most of the techniques perform diversification as a post-processing or re-ranking step of document retrieval. Related work on result diversification in IR also includes [25], [26], [27]. The authors in [25] used probabilistic framework to diversify document ranking, by which web search result diversification is addressed. They also applied the similar model to discuss search result diversification through sub-queries in [26]. The authors in [27] proposed a set of natural axioms that a diversification system is expected to satisfy, by which it can improve user satisfaction with the diversified results. Different from the above relevant works, in this paper, our diversification model was designed to process keyword queries over structured data. We have to consider the structures of data in our model and algorithms, not limited to pure text data like the above methods. In addition, our algorithms can incrementally generate query suggestions and evaluate them. The diversified search results can be returned with the qualified query suggestions without depending on the whole result set of the original keyword query.

Recently, there are some relevant work to discuss the problem of result diversification in structured data. For instance, [28] conducted thorough experimental evaluation of the various diversification techniques implemented in a common framework and proposed a threshold-based method to control the tradeoff between relevance and diversity features in their diversification metric. But it is a big challenge for users to set the threshold value. The authors in [29] developed efficient algorithms to find top-k most diverse set of results for structured queries over semi-structured data. As we know, a structured query can be used to express much more clear search intention of a user. Therefore, diversifying structured query results is less significant than that of keyword search results. In [30], the authors focused on the selection of diverse item set, not considering structural relationships of the items to be selected. The most relevant work to ours is the approach DivQ in [13] where the authors first identified the attribute-keyword pairs for an original keyword query and then constructed a large number of structured queries by connecting the attribute-keyword pairs using the data schema (the attributes can be mapped to corresponding labels in the schema). The challenging problem in [13] is that two generated structured queries with slightly different structures may still be considered as different types of search intentions, which may hurt the effectiveness of diversification as shown in our experiments. However, our diversification model in this work utilized mutually co-occurring feature term sets as contexts to represent different query suggestions and the feature terms are selected based on their mutual correlation and the distinct result sets together. The structure of data are considered by satisfying the exclusive property of SLCA semantics.

7 Conclusions

In this paper, we first presented an approach to search diversified results of keyword query from XML data based on the contexts of the query keywords in the data. The diversification of the contexts was measured by exploring their relevance to the original query and the novelty of their results. Furthermore, we designed three efficient algorithms based on the observed properties of XML keyword search results. Finally, we verified the effectiveness of our diversification model by analyzing the returned search intentions for the given keyword queries over DBLP dataset based on the nDCG measure and the possibility of diversified query suggestions. Meanwhile, we also demonstrated the efficiency of our proposed algorithms by running substantial number of queries over both DBLP and XMark datasets. From the experimental results, we get that our proposed diversification algorithms can return qualified search intentions and results to users in a short time.

8 Acknowledgments

This work was supported by the ARC Discovery Projects under Grant No. DP110102407 and DP120102627, and partially supported by grant of the RGC of the Hong Kong SAR, No.418512.

References

[1] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword search on structured and semi-structured data," in *SIGMOD Conference*, 2009, pp. 1005–1010.

[2] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "Xrank: Ranked keyword search over xml documents," in *SIGMOD Conference*, 2003, pp. 16–27.

[3] C. Sun, C. Y. Chan, and A. K. Goenka, "Multiway slca-based keyword search in xml data," in *WWW*, 2007, pp. 1043–1052.

[4] Y. Xu and Y. Papakonstantinou, "Efficient keyword search for smallest lcas in xml databases," in *SIGMOD Conference*, 2005, pp. 537–538.

[5] J. Li, C. Liu, R. Zhou, and W. Wang, "Top-k keyword search over probabilistic xml data," in *ICDE*, 2011, pp. 673–684.

[6] J. G. Carbonell and J. Goldstein, "The use of mmr, diversity-based reranking for reordering documents and producing summaries," in *SIGIR*, 1998, pp. 335–336.

[7] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, "Diversifying search results," in *WSDM*, 2009, pp. 5–14.

[8] H. Chen and D. R. Karger, "Less is more: probabilistic models for retrieving fewer relevant documents," in *SIGIR*, 2006, pp. 429–436.

[9] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon, "Novelty and diversity in information retrieval evaluation," in *SIGIR*, 2008, pp. 659–666.

[10] A. Angel and N. Koudas, "Efficient diversity-aware search," in *SIGMOD Conference*, 2011, pp. 781–792.

[11] F. Radlinski and S. T. Dumais, "Improving personalized web search using result diversification," in *SIGIR*, 2006, pp. 691–692.

[12] Z. Liu, P. Sun, and Y. Chen, "Structured search result differentiation," *PVLDB*, vol. 2, no. 1, pp. 313–324, 2009.

[13] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl, "*DivQ*: diversification for keyword search over structured databases," in *SIGIR*, 2010, pp. 331–338.

[14] J. Li, C. Liu, R. Zhou, and B. Ning, "Processing xml keyword search by constructing effective structured queries," in *APWeb/WAIM*, 2009, pp. 88–99.

[15] H. Peng, F. Long, and C. H. Q. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, 2005.

[16] C. O. Sakar and O. Kursun, "A hybrid method for feature selection based on mutual information and canonical correlation analysis," in *ICPR*, 2010, pp. 4360–4363.

[17] N. Sarkas, N. Bansal, G. Das, and N. Koudas, "Measure-driven keyword-query expansion," *PVLDB*, vol. 2, no. 1, pp. 121–132, 2009.

[18] N. Bansal, F. Chiang, N. Koudas, and F. W. Tompa, "Seeking stable clusters in the blogosphere," in *VLDB*, 2007, pp. 806–817.

[19] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *SIGMOD Conference*, 1997, pp. 265–276.

[20] W. DuMouchel and D. Pregibon, "Empirical bayes screening for multi-item associations," in *KDD*, 2001, pp. 67–76.

[21] A. Silberschatz and A. Tuzhilin, "On subjective measures of interestingness in knowledge discovery," in *KDD*, 1995, pp. 275–281.

[22] "http://dblp.uni-trier.de/xml/."

[23] "http://monetdb.cwi.nl/xml/."

[24] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[25] R. L. T. Santos, C. Macdonald, and I. Ounis, "Exploiting query reformulations for web search result diversification," in *WWW*, 2010, pp. 881–890.

[26] R. L. T. Santos, J. Peng, C. Macdonald, and I. Ounis, "Explicit search result diversification through sub-queries," in *ECIR*, 2010, pp. 87–99.

[27] S. Gollapudi and A. Sharma, "An axiomatic approach for result diversification," in *WWW*, 2009, pp. 381–390.

[28] M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. T. Jr., and V. J. Tsotras, "On query result diversification," in *ICDE*, 2011, pp. 1163–1174.

[29] M. Hasan, A. Mueen, V. J. Tsotras, and E. J. Keogh, "Diversifying query results on semi-structured data," in *CIKM*, 2012, pp. 2099–2103.

[30] D. Panigrahi, A. D. Sarma, G. Aggarwal, and A. Tomkins, "Online selection of diverse results," in *WSDM*, 2012, pp. 263–272.

Jianxin Li received his BE and ME degrees in computer science, from the Northeastern University, China, in 2002 and 2005, respectively. He received his PhD degree in computer science, from the Swinburne University of Technology, Australia, in 2009. He is currently a postdoctoral research fellow at Swinburne University of Technology. His research interests include database query processing & optimization, and social network analytics.

Chengfei Liu received the BS, MS and PhD degrees in Computer Science from Nanjing University, China in 1983, 1985 and 1988, respectively. Currently he is a Professor in the Faculty of Science, Engineering and Technology, Swinburne University of Technology, Australia. His current research interests include keywords search on structured data, query processing and refinement for advanced database applications, query processing on uncertain data and big data, and data-centric workflows. He is a member of IEEE, and a member of ACM.

Jeffrey Xu Yu received the BE, ME, and PhD degrees in computer science, from the University of Tsukuba, Japan, in 1985, 1987, and 1990, respectively. Currently he is a Professor in the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong. His major research interests include graph mining, graph database, social networks, keyword search, and query processing and optimization. He is a senior member of IEEE, a member of the IEEE Computer Society, and a member of ACM.