Finding Top-*k* Semantically Related Terms in Relational Keyword Search

ABSTRACT

Due to the insufficient knowledge of users about the database schema and content, most of them cannot easy to find appropriate keywords to express their query intentions. This paper proposes a novel approach, which can provide a list of keywords that semantically related to the set of given keywords by analyzing the correlations between terms in database. The suggestion would broaden the view of users and help them to formulate more efficient keyword queries. To capture the correlations between terms in database, a coupling relationship measuring method is proposed to model both the intra- and inter-term couplings, which can reveal the explicit and implicit relationships between terms. For a given keyword query, based on the coupling relationships between terms, an order of all terms in database is created for each query keywords and then the threshold algorithm (TA) is to expeditiously generate top-*k* ranked semantically related terms. The experiments demonstrate that our term coupling relationship measuring method can efficiently capture the semantic correlations between terms. The performance of top-*k* term selection algorithm is also demonstrated.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems – *Query processing*;
H.2.8 [**Database Management**]: Database applications– *Data mining*

General Terms

Algorithms, Performance, Design, Experimentation.

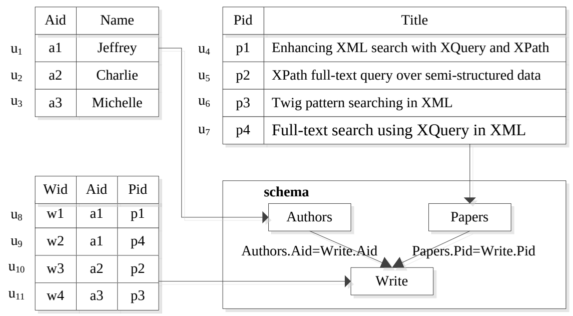
Keywords

Relational database, keyword search, term coupling relationship, top-*k* selection.

1 INTRODUCTION

Keyword query is becoming a very popular way to obtain the information from the relational database along with its wide spread use on the Internet. In reality, however, most of common users usually have insufficient knowledge about the database content and schema, and they are also lack of keywords related to the search domain. Thus, it is not easy for them to find appropriate keywords to express their query intentions. In real applications, to explore the database, the user may issue a query with a few general keywords at first, and then gradually refines the query through observing the query results. In such an iteration, the user needs to check each result to identify whether it is related to his interest or not, which is a time-consuming and tedious work.

Consider a DBLP database consisting of 3 relations connected through primary-foreign-key relationships showed in Figure 1.



	Aid	Name
u_1	a1	Jeffrey
u_2	a2	Charlie
u_3	a3	Michelle

	Pid	Title
u_4	p1	Enhancing XML search with XQuery and XPath
u_5	p2	XPath full-text query over semi-structured data
u_6	p3	Twig pattern searching in XML
u_7	p4	Full-text search using XQuery in XML

	Wid	Aid	Pid
u_8	w1	a1	p1
u_9	w2	a1	p4
u_{10}	w3	a2	p2
u_{11}	w4	a3	p3

Figure 1. An example of DBLP database

Suppose a master student who is a XML beginner just knows a few keywords about XML research field and wants to find papers about the XML search techniques from DBLP website. Based on the DBLP database, he/she would issue a query Q containing keywords "XML, search". On receiving the query Q, the traditional keyword search approach will return a set of minimal total joint networks (MTJNTs), each of which (i) is obtained from a single relation or by joining several relations, and (ii) contains all the query keywords. Since there are too many papers containing keywords "XML" and "search" in DBLP dataset, there are too many MTJNTs in the query results. In such a case, the user would like the system suggest a list of keywords that are semantically related to Q in order to reduce the searching scope. From Figure 1, it is clearly that the author "Jeffrey" and keywords "XPath", "XQuery", and "twig pattern" are very relevant to Q. That means these terms can refine Q to formulate a more selective query. As an example, the user would execute a query Q'=[Jeffrey, XML, search] to retrieve only the papers of author Jeffrey on XML searching and the query results are "$a_1 \bowtie w_1 \bowtie p_1$" and "$a_1 \bowtie w_2 \bowtie p_4$". Additionally, the tuples $p2$ and $p3$ containing "full-text", "semi-structured data", and "twig pattern" are also related to the query Q. While, these tuples would not be returned by the system due to the terms they contained are not specified explicitly by the user query. If the user is also interested in these topics, he/she can choose the keyword "full-text", "semi-structured data", and/or "twig pattern" to explore the database. Hence, it is necessary to provide a list of semantically related terms to the given query and then the user can refine or reformulate his/her query according to the terms in the list.

The challenge in selecting semantically related keywords is to understand the semantics of the original query and to measure the semantic relationships between query keywords and database terms. Several approaches have been proposed to deal with the issue of keyword search over relational databases [1-5]. The basic idea of the approaches is to assume the query keywords are independent to each other and leverage full text matching to find all connected tuples explicitly contain all the query keywords. However, in the real world, there are various coupling relationships [6] between objects, which have been shown valuable to be incorporated into analysis such as document term semantic analysis [7], clustering [8] and classification [9]. Similarly, terms contained in tuples are coupled in terms of co-occurrences and inter-related relationships. If the query keywords can be mapped into database terms, then the semantic relationships between query keywords and database terms can be estimated by the coupling relationships between database terms. The coupling relationship of terms is composed of *intra-coupling* and *inter-coupling*, where *intra-coupling* denotes the explicit relationship between terms (such as two terms co-occurred in same tuples) and *inter-coupling* represents the implicit relationship between terms (such as two terms occurred separately in different tuples are inter-related through at least one common term). On top of this idea, in this paper, we propose a new approach which incorporates the term coupling relationships to provide a list of relevant terms rather than the MTJNTs. Given a set of keywords *Q*, and an integer *k*, our approach returns the *k* most semantically related terms from the database to *Q*.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 gives a formal definition of the problem and outlines an overview of our solution framework. Section 4 proposes the term coupling relationship measuring method while Section 5 presents a top-*k* related term selection method. The experiment results are presented in Section 6. The paper is concluded in Section 7.

2 Related Work

Several methods have been proposed to handle keyword search on relational database, and the popularity of keyword search is ongoing [11]. The previous work can be classified into two main categories, depending on whether they retrieve MTJNTs based on candidate networks (CN) [2, 3, 13] or data graph [1, 5, 12]. The CN-based approaches, such as DBXplorer [2], DISCOVER [3], and SPARK [13], generate all possible candidate networks following the database schema, and then identify the MTJNTs based on CNs. A CN is a joining network of tuples, in which the tuples are inter-connected through primary-foreign-key constraints. The data graph-based methods, such as BANKS [1] and its extensions [5, 12], firstly model the database as a directed data graph, where nodes are tuples and the directed edges are foreign key references between tuples. A keyword query is then processed by traversing graph for searching MTJNTs containing the query keywords. In summary, the existing approaches mainly focus on searching MTJNTs explicitly containing the specified keywords and lack of considering the semantic relevance between answers and queries. As a result, they cannot identify the results from which some MTJNTs may also be very relevant to a query in semantic terms, even though they do not explicitly contain the query keywords.

Our approach has a fundamental difference from the conventional keyword search (KS) approach: our approach extract terms, while KS approach fetches joint tree of tuples. More specifically, given a set of query keywords *Q* and an integer *k*, a top-*k* KS approach aims to find the *k* MTJNTs most relevant to *Q* and the MTJNTs are ranked according to their content relevance or tree size. In contrast, our approach selects the *k* terms most relevant to *Q* by measuring the coupling relationships between query keywords and database terms. Note that, the *k* terms produced by our approach do not necessarily appear in the *k* MTJNTs fetched by top-*k* KS approach. The reason is that, some of MTJNTs in results may not real relevant to the user intentions even it contains the query keywords, while some tuples do not contain the query keywords may very relevant to the user need in terms of semantic, these tuples would not be retrieved by the existing KS approach.

Recently, tentative work on keyword semantic understanding and approximate query has been undertaken. In [17], the transformation rules are manually defined used for keyword query integration and the local results are analyzed used for finding relevant answers. In [18], the metadata of database is used for translating keyword queries into meaningful SQL queries that describe the intended query semantics. In [11], the data structural semantics are exploited and employed to reformulate the initial query. Although keyword/term semantics have been taken into consideration, most of the existing approaches usually assume that keywords in a query (resp. terms in database) are independent of one another, but in reality coupling relationships exist between objects such as keywords and terms as shown in [6, 7].

The work that is most similar to ours is the FCT (frequent co-occurring term) in [19], which address the problem of how to find the top-*k* frequent co-occurring term from the keyword query results by using the fast STAR algorithm. Our approach differs from that in [19] in the following aspects:

1. Given a set of query keywords *Q*, and an integer *k*, a FCT query returns the *k* most frequent terms in the results of a keyword query with the same *Q* while our approach finds the *k* most relevant terms in the overall database rather than the query results of *Q*. The advantage of our approach is providing a global perspective of correlations between query keywords and database terms for users and can lead users find more appropriate keywords or concepts to refine/reformulate their queries.

2. FCT only takes the frequency of occurrence of terms in query results into consideration for measuring the related concepts/terms to the original query. In contrast, our approach considers both the co-occurrence and inter-relation between terms for measuring the coupling relationships between query keywords and database terms. Thus, our approach can reveal both the explicit and implicit relationships between query keywords and database terms.

3 Problem Definition and Solution

In this section, we first present the problem definition and then introduce our solution.

3.1 Problem Definition

Definition 1 (Schema graph). Consider an relational database *D* as a collection of relations $D=(r(R_1), r(R_2), \ldots, r(R_n))$, where each relation $r(R_i)$ in *D* contains n_i tuples with the schema R_i. A *schema graph* of relational database *D* is a directed graph $G_S(V, E)$, where *V* is the set of nodes and each of which represents a relation $r(R_i)$ in *D*, *E* is the set of edges and each of which represents a foreign key reference between a pair of relations in *D*. Given two relation schemas R_i and R_j, there exists an edge in the schema graph G_S, from R_j to R_i, denoted $e(R_i \rightarrow R_j)$, if the foreign key defined on R_j references to the primary key defined on R_i. Figure 1 illustrates the schema graph of the sample DBLP database.

In this paper, we suppose any two relations are connected in the schema graph. If some relations are not connected, it should be decomposed into several groups of connected relations and apply our method on the decomposed groups. A relation R_i is called a *link relation* if there is no relation R_j, such that $R_j \rightarrow R_i$. That is, R_i only contains foreign keys to reference other relations but there is no primary key be defined on it. For example, the relation *Write* in DBLP is a link relation because relation *Write* has no primary key and there exists *Write*→*Authors* and *Write* →*Papers* in the schema graph.

A relational database can be modeled as a database graph $G_D(V, E)$ on the schema graph G_S, where V represents the set of tuples in database, and E represents the set of connections between tuples. There is a connection between two tuples, t_i and t_j in G_D, if there exists at least on foreign key references from t_i to t_j (or t_j to t_i) in the database. Figure 2 illustrates the database graph G_D for the sample DBLP database showed in Figure 1.

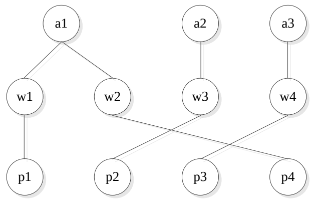


Figure 2. Tuple connections of the sample DBLP database

Definition 2 (Minimal Total Joining Network of Tuples, *MTJNT*). Given a l-keyword query Q and a relational database D with schema graph G_S, a joining network of tuples (*JNT*) is a connected tree of tuples where two adjacent tuples, $t_i \in r(R_i)$ and $t_j \in r(R_j)$, can be joined according to the foreign key references defined on relational schema R_i and R_j in G_S. An MTJNT is a JNT that satisfy the following two conditions:

(i) Total: each keyword in query Q must be contained in at least one tuple of the JNT.

(2) Minimal: no tuple of the JNT can be removed such that the remaining tuples is still a JNT contains all the keywords in Q. In other words, a JNT is not total if any tuple is removed.

Definition 3 (*l*-Keyword query). A l-keyword query Q over database D is an ordered list of distinct keywords of size l, i.e., $Q=\{k_1, k_2, \ldots, k_l\}$, and searches inter-connected tuples that contain the given keywords in their text attributes. A l-keyword query returns a set of answers, each of them is a minimal total joining network of tuples (*MTJNT*).

Problem 1 (Top-*k* semantically related term selection). Let Q be a set of query keywords over a relational database D. The top-k semantically related term selection problem is defined as,

$$\Gamma_k = argmax_{\Gamma} \sum_{i=1}^{k(k<n)} \delta_{SR}(t_i, Q) \quad (1)$$

where, Γ_k is a list of k terms, n is the number of all distinct terms in D, and $\delta_{SR}(t_i, Q)$ represents the semantic relationship between a term t_i and the set of query keywords Q. The objective of the problem is to find a set of number k terms in D that semantically related closely as possible to the set of given query keywords.

3.2 Solution

This paper proposes a two-step processing solution to address this problem. The first step occurs offline. It firstly extracted all the distinct terms from the database, each of which takes the form of <attribute, keyword>, where *attribute* refers to the attribute name in the relation, *keyword* is a word or topical phrase in the values corresponding to the attribute. Here, we suppose the attributes belong to different relations have different names, so that each <attribute, keyword> is unique and would not be duplicated. In addition, the standard word-stemming technique should be applied, so that words like “obtain” and “obtaining” can be regarded as the same word. And then, a data view is generated by connecting all the relations in the database according to their primary-foreign-key references, following which the intra- and inter-couplings between different pairs of terms can be calculated by leveraging the correlation analysis method on data view. Consequently, the term intra- and inter-coupling can be combined into a coupling relationship to reflect the semantic relevance between terms.

The second step occurs online when a user makes a query. It first decomposes the input query into several distinct keywords. Based on coupling relationships between terms, it then creates orders of terms for each query keyword. Each order corresponds to a query keyword and the terms in each order are ranked according to their coupling relationships to that keyword. After this, the top-k related terms can be quickly captured by using Threshold Algorithm (TA) on the orders.

4 Term Coupling Relationship Analysis

In this Section, we first generate the term relationship graph, and then describe how to measure the weights of nodes and edges in the term relationship graph.

4.1 Term Relationship Graph

We use term relationship graph to model the relationships between terms in database. Figure 3 illustrates the relationships of the terms extracted from the example DBLP database in Figure 1. The set of nodes is the set of all terms. There is an edge between two nodes corresponding terms t_i and t_j, if (i) t_i and t_j are in the same tuple of a relation, or (ii) t_i and t_j exist in tuples u_x and u_y from different relations that can be connected through a sequence of primary-foreign key references.

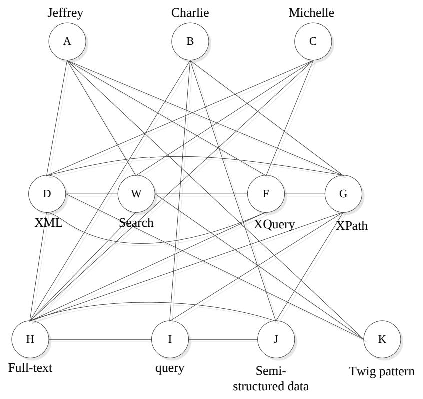


Figure 3. Term relationship graph for the DBLP database

As showed in Figure 3, the relationships between terms can be divided into explicit and implicit relationships. Two nodes are explicitly related if there is an edge between them such as nodes A and D. Two nodes are implicitly related if they can be inter-connected through at least one link/common nodes but there is no edge between them. For example, the nodes H and K are inter-connected through the nodes D and W. To measure the relationships (including explicit and implicit relationships) between two terms in the graph, the weights for nodes and edges should be computed based on the database

4.2 Weight of a Node

A straightforward way to weight the node of graph is to employ the TF·IDF-based method. Each tuple in the relation can be modeled as a document and the keywords in the tuple are treated as terms, and accordingly the technique of TF·IDF weighting function in IR and database literatures [20, 21] can be borrowed and incorporated to weight the node in the graph.

Given a database *D* and a term t_i appearing in a tuple *u* ($u \in D$), we use $f(t_i, u)$ to denote the number of occurrences of the term t_i in tuple *u*. *N* is the cardinality of tuples that include terms in database *D* (we do not count the number of tuples contained in link tables because of which containing no term), and N_i is the number of tuples containing term t_i in *D*. We then use the TF-IDF metric

The normalized term frequency of term t_i in *u*, $ntf(t_i, u)$, can be defined as Equation (2),

$$ntf(t_i, u) = 1 + \ln(1 + f(t_i, u)) \tag{2}$$

The inverse tuple frequency $itf(t_i)$ is defined as Equation (3),

$$itf(t_i) = ln\frac{N}{N_i+1} \tag{3}$$

in which, the itf is normalized by dividing the total number of tuples in *D* over $(N_i + 1)$ and then applying the *ln* function.

Since the length of each tuple (i.e., the number of terms contained in a tuple) may different from each other, it needs to be normalized. The normalized tuple length (*ntl*), which is used to reduce the term weights in long tuples, is defined as Equation (4),

$$ntl(u) = (1 - s) + s * \frac{|u|}{\frac{\sum_{u' \in D}|u'|}{N}} \tag{4}$$

where, $|u|$ represents the number of terms in tuple *u*, and *s* is a constant that is usually set to 0.2. Normalized tuple length means the ratio of the number of terms in tuple *u* to the average of terms in the set of tuples in *D*.

After this, the weight of term t_i in tuple *u* can be defined as,

$$w(t_i, u) = \frac{ntf(t_i,u)}{ntl} * itf(t_i) \tag{5}$$

Since the term t_i may appear in several tuples, the weight of the node representing term t_i is normalized as follows,

$$w(t_i) = \frac{\sum_{u=1}^{N_i} w(t_i,u)}{N_i} \tag{6}$$

It is clearly that $w(t_i, u)$ captures the importance of term t_i in a specific tuple *u* while $w(t_i)$ reflects the average importance of term t_i among all tuples containing it. For example, the sample DBLP database of Figure 1 contains *N*=7 tuples and $\sum_{u' \in D}|u'| =$ 18 terms (here, we count the total number of terms showed in Figure 6 appearing in the tuples). The term "XML" appeared in N_i=3 tuples (u_4, u_6, u_7), and the times of term "XML" appeared in each of these tuples are both 1, i.e., $f(\text{XML}, u_4)$=1, $f(\text{XML}, u_6)$=1, and $f(\text{XML}, u_7)$=1, respectively. Also, the length of these tuples are $|u_4|$=4, $|u_6|$=3, and $|u_7|$=4, respectively. Consequently, the weight of the term "XML" in these tuples are, $w(XML, u_4) = \frac{1+\ln(1+1)}{0.8+0.2*4/(11/7)} * \ln\left(\frac{7}{4}\right) = 0.73$, $w(XML, u_6) = \frac{1+\ln(1+1)}{0.8+0.2*3/(11/7)} * \ln\left(\frac{7}{4}\right) = 0.80$, and $w(XML, u_7) = \frac{1+\ln(1+1)}{0.8+0.2*4/(11/7)} * \ln\left(\frac{7}{4}\right) = 0.73$, respectively. As a result, the weight of node corresponding the term "XML' in graph is $w(XML) = \text{average}(0.73 + 0.80 + 0.73) = 0.75$.

After this, all the nodes are finally normalized by dividing the maximum weight of the node in graph. The normalized weight of nodet_i, $nw(t_i)$, is defined as,

$$nw(t_i) = \frac{w(t_i)}{MaxWeight} \tag{7}$$

4.3 Weight of Edges

To compute the weights of edges, we first construct a data view that is the collection of all connected tuples based on the database schema graph, and then capture the weight (composed of intra and inter coupling relationships) of edge between two connected nodes of term relationship graph based on data view.

4.3.1 Data view

Definition 4 (*Data view*): Given a database *D* with *n* connected relations, $r(R_1)$, $r(R_2)$, …, $r(R_n)$, the data view *V* is formed by joining all connected relations in *D* through the primary-foreign-key relationships according to the schema graph, that is, $V = r(R_1) \bowtie r(R_2) \bowtie \ldots \bowtie r(R_n)$, where each tuple *u* in *V* is a combination of connected tuples and represents a meaningful and integral unit.

For example, we can join the 3 relations in Figure 1 to create a data view as shown in Table 1.

Table 1 An instance of data view for sample DBLP database

AID	PID	Name	Title
a1	p1	Jeffrey	Enhancing XML search with XQuery and XPath
a1	p4	Jeffrey	An efficient full-text search using XQuery in XML
a2	p2	Charlie	XPath full-text query over semi-structured data
a3	p3	Michelle	Twig pattern searching in XML

The data view contains much richer structural information than the text document. Based on data view, we next present how to compute the term coupling relationship, which is inspired by the term coupled modeling in document analysis [7].

4.3.1 Term-intra Couplings in the Tuples

In Information Retrieval, two terms are considered semantically related if they frequently co-occur in the same document. Similarly, each tuple in a data view is considered as a document so that we can mimic this idea to measure the intra-coupling between terms in database.

The frequency of co-occurrence of a pair of terms (t_i, t_j) appearing in the same tuple can be measured by Jaccard coefficient as follows,

$$J(t_i, t_j) = \frac{|V(t_i) \cap V(t_j)|}{|V(t_i) \cup V(t_j)|} \tag{8}$$

in which, $V(t_i)$ and $V(t_j)$ represents the subset of tuples in view *V* containing terms t_i and t_j, respectively.

Given a term, such as <Author, Jeffrey>, it can be visualized as a selection query "Author=Jeffrey" that binds only a single attribute. By issuing a term query over the view, a set of tuples all containing the keyword of term can be identified.

It should be pointed out that, given a tuple *u* and any two terms in *u*, t_i and t_j, they can be classified into two cases according to their relationships in the tuple *u* as follows:

(i) t_i and t_j bind the same attribute;
(ii) t_i and t_j correspond to different attribute.

It is clearly that t_i and t_j in case (i) are more relevant than those in case (ii). Thus, we set the distance between two terms t_i and t_j in a tuple u, denoted as $d_r(t_i, t_j)$, as showed in Equation (9).

$$d(t_i, t_j) = \begin{cases} 0 & t_i \text{ and } t_j \text{ correspond to case } (i) \\ 1 & t_i \text{ and } t_j \text{ correspond to case } (ii) \end{cases} \quad (9)$$

After this, we can define the *term intra-coupling* by considering both of the Jaccard coefficient and term distance.

Definition 5 (*Intra-coupling of terms*): Given a view V and any two terms t_i and t_j, there exists a intra-coupling relationship between t_i and t_j if they co-occur in at least one tuple u of V, the intra-coupling between t_i and t_j in V is defined as,

$$\delta_{IaR}(t_i, t_j|V) = \frac{J(t_i,t_j)}{d_r(t_i,t_j)+1} \quad (10)$$

where, $J(t_i, t_j)$ is defined as Equation (8).

Since term t_i may also co-occur with other terms in the same tuple, it should be normalized by dividing the total number of intra-couplings between t_i and all other terms. Thus, the intra-coupling between t_i and t_j can be finally computed as follows,

$$\delta_{IaR}(t_i, t_j) = \begin{cases} 1 & i = j \\ \frac{\delta_{IaR}(t_i,t_j|V)}{\sum_{a=1,a\neq i}^{n} \delta_{IaR}(t_i,t_a|V)} & i \neq j \end{cases} \quad (11)$$

in which, n is the number of all distinct terms extracted from database D.

For instance, given two terms "Jeffrey" and "XML" (for simple, we only use keyword to denote its corresponding term). The Jaccard coefficient of them in the view of Table 1 is $J(Jeffrey, XML) = \frac{2}{3}$ and the distance is $d(Jeffrey, XML) = 1$, respectively. Consequently, the intra-coupling between them is $\delta_{IaR}(Jeffrey, XML|V) = \frac{1}{3}$. Since the term "Jeffrey" also co-occurs with terms "search", "XQuery", "XPath", and "full-text" in the view and the intra-couplings between them are $\delta_{IaR}(Jeffrey, search|V) = \frac{1}{3}$, $\delta_{IaR}(Jeffrey, XQuery|V) = \frac{1}{2}$, $\delta_{IaR}(Jeffrey, XPath|V) = \frac{1}{6}$, and $\delta_{IaR}(Jeffrey, full-text|V) = \frac{1}{6}$, respectively. Finally, the normalized intra-coupling between "Jeffrey" and "XML" is $\delta_{IaR}(Jeffrey\ , XML) = \frac{1/3}{1/3+1/3+1/2+1/6+1/6} = \frac{2}{9}$.

It is clearly to conclude that from the example above, for each pair of terms t_i and t_j, we have δ_{IaR} (t_i, t_j $|V$)≥0 and $\sum_{j=1,j\neq i}^{n} \delta_{IaR}(t_i, t_j|V) = 1$. Note that, the values of $\delta_{IaR}(t_i, t_j)$ and $\delta_{IaR}(t_j, t_i)$ may not be equal to each other due to the different dominators. While, the matrix of $\delta_{IaR}(t_i, t_j|V)$ in Equation (10) is symmetric because of $J(t_i, t_j)$= $J(t_j, t_i)$, therefore we need to only compute the upper-half of the matrix of $\delta_{IaR}(t_i, t_j|V)$. The term intra-coupling relationship calculating algorithm is shown in Algorithm 1.

Algorithm 1. Term intra-coupling calculating algorithm

```
Input: data view V, set of all distinct terms T in D, number of terms n.
Output: IaRMatrix
 1. IaRMatrix=null.
 2. for i=1 to n-1 do
 3.   for k=i+1 to n do
 4.     IaRMatrix[i][k]=J(T[i], T[k])/d(T[i], T[k]).
 5.     IaRMatrix[k][i]=IaRMatrix[i][k].
 6.   end for
 7.   for m=1 to n do
 8.     if (m≠i) then
 9.       Sum=Sum+IaRMatrix[i][m].
10.   end for
11.   for j=1 to n do
12.     if (j≠i) then
13.       IaRMatrix[i][j]= $\frac{\text{IaRMatrix}[i][j]}{\text{Sum}}$.
14.   end for
15.end for
16.returnIaRMatrix.
```

Using algorithm 1, an intra-coupling matrix for each pair of terms can be obtained. Table 2 shows the intra-coupling matrix of terms extracted from the sample DBLP database. For simple, we use *A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, *I*, *J*, and *K* to denote the extracted terms *Jeffrey*, *Charlie*, *Michelle*, *XML*, *Search*, *XQuery*, *XPath*, *full-text*, *query*, *semi-structured data*, and *twig pattern*, respectively.

Table 2. Example of intra-coupling matrix of terms

	A	B	C	D	E	F	G	H	I	J	K
A	1.00	0.00	0.00	0.22	0.22	0.33	0.11	0.11	0.00	0.00	0.00
B	0.00	1.00	0.00	0.00	0.00	0.00	0.17	0.17	0.33	0.33	0.00
C	0.00	0.00	1.00	0.20	0.20	0.00	0.00	0.00	0.00	0.00	0.60
D	0.11	0.00	0.06	1.00	0.33	0.22	0.08	0.08	0.00	0.00	0.11
E	0.11	0.00	0.06	0.33	1.00	0.22	0.08	0.08	0.00	0.00	0.11
F	0.20	0.00	0.00	0.27	0.27	1.00	0.13	0.13	0.00	0.00	0.00
G	0.06	0.10	0.00	0.10	0.10	0.13	1.00	0.13	0.19	0.19	0.00
H	0.06	0.10	0.00	0.10	0.10	0.13	0.13	1.00	0.19	0.19	0.00
I	0.00	0.20	0.00	0.00	0.00	0.00	0.20	0.20	1.00	0.40	0.00
J	0.00	0.20	0.00	0.00	0.00	0.00	0.20	0.20	0.40	1.00	0.00
K	0.00	0.00	0.43	0.29	0.29	0.00	0.00	0.00	0.00	0.00	1.00

The intra-coupling reflects the explicit relationship between two co-occurred terms. Specifically, if there are two terms co-occur in the view, there must exist an edge between the two nodes in graph that the terms correspond to.

However, besides the intra-coupling, some co-occurred terms may also appear separately in different tuples and they are probably inter-related through their *common terms*. In this paper, we call such implicit correlation between terms is *inter-coupling relationship*, which would enhance the relationships between the co-occurred terms. For example, given two terms "XPath" and "XQuery", from the Table 1 we can see that they co-occur in tuple 1 and appear separately in tuple 2 and tuple 3, respectively. Clearly, the common term between "XPath" and "XQuery" is "full-text", which appears together with "XPath" and "XQuery" in tuple 2 and tuple 3 of Table 1, respectively. Additionally, the terms have never co-occurred in the same tuples, may also inter-related via their common terms. For example, the terms "semi-structured data" and "XML" are inter related by their common terms "XPath" and "full-text". Next, we will propose the term inter-coupling measuring method below to capture the implicit relationships between inter-related terms.

4.3.2 Term Inter-coupling across Tuples

Given a data view V and a term t_i, all the terms co-occurred with t_i in V can be seen as the relevant terms associated with t_i. For any two terms t_i and t_j that appear in different tuples, the inter-coupling between them can be estimated by the commonality in the relevant terms associated with them. For example, given a term <Title, XML> in Table 1, a set of terms <Author, Jeffrey >, <Author, Michelle>, <Title, search>, <Title, XQuery>, <Title, XPath>, <Title, twig pattern>, and <Title, full-text> is associated with it; while, a set of terms <Author, Charlie>, <Title, XPath>, <Title, full-text>, and <Title, query> is associated with the term <Title, semi-structured data>. Clearly, the overlapped terms between two sets are <Title, XPath> and <Title, full-text>. In this paper, we call these terms are *common terms/common nodes* of the compared terms, which mean that two terms appearing in different tuples are inter-related through their common terms. According to this, the inter-coupling between terms t_i and t_j through their common term t_c can be defined as follows.

Definition 6 (*Inter-coupling of terms*): Given a data view V and any two terms t_i and t_j, they are inter-related if there is at least one common term t_c such that $\delta_{IaR}(t_i, t_c)>0$ and $\delta_{IaR}(t_j, t_c)>0$ hold but terms t_i and t_j appear in different tuples. The inter-coupling between term t_i and t_j via common term t_c is defined as follows,

$$\delta_{IeR}(t_i, t_j|t_c) = \min\{\delta_{IaR}(t_i, t_c), \delta_{IaR}(t_j, t_c)\} \tag{12}$$

where, $\delta_{IaR}(t_i, t_c)$ and $\delta_{IaR}(t_j, t_c)$ are the intra-coupling between terms t_i and t_c, t_j and t_c, respectively.

Since there may be more than one common term between t_i and t_j and each one have different weight in the term relationship graph, we use the following method to normalize the term inter-couplings. Suppose S be the set of common terms of t_i and t_j, that is, $S=\{t_c|(\delta_{IaR}(t_i, t_c)>0 \wedge \delta_{IaR}(t_j, t_c)>0)\}$. Then, the inter-coupling between term t_i and t_j, inter-related by all the common terms in S, can be formalized as,

$$\delta_{IeR}(t_i, t_j) = \begin{cases} 1 & i = j \\ \frac{\sum_{\forall t_c \in S} nw(t_c) * \delta_{IeR}(t_i, t_j|t_c)}{|S|} & i \neq j \end{cases} \tag{13}$$

where, $nw(t_c)$ represents the weight of term t_c which is computed by Equation (7), $|S|$ denotes the number of common terms in S, and $\delta_{IeR}(t_i, t_j|t_c)$ is the inter-coupling between t_i and t_j inter-connected via their common term t_c. Equation (13) means that the inter-coupling between term t_i and t_j is measured by the average strength of all the weights of edges between them. If $S=\Phi$, then $\delta_{IeR}(t_i, t_j)$ is zero. The term inter-coupling calculating algorithm is shown in Algorithm 2.

Algorithm 2. Term inter-coupling calculating algorithm

```
Input: set of terms $T$, number of terms $n$, IaRMatrix, weights of terms
Output: IeRMatrix
 1. IeRMatrix=null.
 2. for $i$=1 to $n$-1 do
 3.   for $j$=1 to $n$do
 4.     $S$←the set of common terms between T[$i$] and T[$j$].
 5.     $m$=|S|.
 6.     if ($S$=$\Phi$) then
 7.       IeRMatrix[$i$][$j$]=0.
 8.     else
 9.       for $k$=1 to $m$ do
10.         IeRMatrix[$i$][$k$]=min(IaRMatrix[$i$][$k$], IaRMatrix[$j$][$k$]).
11.         IeRMatrix[$i$][$j$]+=$\frac{\text{IeRMatrix}[i][k]*nw(\text{S}[k])}{m}$.
12.       end for
13.   end for
14. end for
15. returnIeRMatrix.
```

Using Algorithm 2, an inter-coupling matrix for each pair of terms can be obtained. Table 3 shows the inter-coupling matrix of terms extracted from the sample DBLP database showed in Figure 1.

Table 3. Example of inter-coupling matrix of terms

	A	B	C	D	E	F	G	H	I	J	K
A	1.00	0.05	0.07	0.07	0.07	0.06	0.04	0.04	0.05	0.05	0.07
B	0.05	1.00	0.00	0.04	0.04	0.06	0.11	0.11	0.13	0.13	0.00
C	0.07	0.00	1.00	0.08	0.08	0.07	0.03	0.03	0.00	0.00	0.07
D	0.07	0.04	0.08	1.00	0.07	0.07	0.05	0.05	0.04	0.04	0.07
E	0.07	0.04	0.08	0.07	1.00	0.07	0.05	0.05	0.04	0.04	0.07
F	0.06	0.06	0.07	0.07	0.07	1.00	0.05	0.05	0.06	0.06	0.09
G	0.04	0.11	0.03	0.05	0.05	0.05	1.00	0.08	0.10	0.10	0.03
H	0.04	0.11	0.03	0.05	0.05	0.05	0.08	1.00	0.10	0.10	0.03
I	0.05	0.13	0.00	0.04	0.04	0.06	0.10	0.10	1.00	0.13	0.00
J	0.05	0.13	0.00	0.04	0.04	0.06	0.10	0.10	0.13	1.00	0.00
K	0.07	0.00	0.07	0.07	0.07	0.09	0.03	0.03	0.00	0.00	1.00

4.3.3 Term Coupling Relationship

The coupling relationship between two terms t_i and t_j is composed of intra- and inter-coupling of them, which is defined as follows,

$$\delta_{SR}(t_i, t_j) = \begin{cases} 1 & i = j \\ (1-\alpha) \cdot \delta_{IaR}(t_i, t_j) + \alpha \cdot \delta_{IeR}(t_i, t_j) & i \neq j \end{cases} \tag{14}$$

where, $\alpha \in [0, 1]$ is the parameter to determine the weight of intra- and inter-coupling. The Equation (14) would be intra-coupling if α=0 while it would be inter-coupling if α=1, which means the intra- and inter-coupling are the special cases of the term coupling relationship. Given two terms t_i and t_j, it is clearly that the higher the coupling relationship between t_i and t_j, the more the t_i semantically related to t_j, and the larger the weight of edge $w(t_i \rightarrow t_j)$ in graph; and vice versa. Note that there are two weights on the edge between any pair of connected nodes in the term relationship graph. More specifically, given two connected nodes t_i and t_j in graph, the weights on the edge between t_i and t_j are, $w(t_i \rightarrow t_j)$ and $w(t_i \rightarrow t_j)$, which represent the coupling relationship from t_i to t_j, and t_j to t_i, respectively.

Table 4 shows the coupling relationship matrix of all terms extracted from sample DBLP database. Here, we set α to 0.5, which means the intra- and inter-coupling have the same ratio in measuring the term coupling relationship.

Table 4. Example of coupling relationship matrix of terms

	A	B	C	D	E	F	G	H	I	J	K
A	1.00	0.03	0.03	0.14	0.14	0.20	0.08	0.08	0.03	0.03	0.04
B	0.03	1.00	0.00	0.02	0.02	0.03	0.14	0.14	0.23	0.23	0.00
C	0.03	0.00	1.00	0.14	0.14	0.03	0.02	0.02	0.00	0.00	0.33
D	0.09	0.02	0.07	1.00	0.20	0.14	0.06	0.06	0.02	0.02	0.09
E	0.09	0.02	0.07	0.20	1.00	0.14	0.06	0.06	0.02	0.02	0.09
F	0.13	0.03	0.03	0.17	0.17	1.00	0.09	0.09	0.03	0.03	0.04
G	0.05	0.10	0.02	0.07	0.07	0.09	1.00	0.11	0.14	0.14	0.02
H	0.05	0.10	0.02	0.07	0.07	0.09	0.10	1.00	0.15	0.15	0.02
I	0.03	0.17	0.00	0.02	0.02	0.03	0.15	0.15	1.00	0.27	0.00
J	0.03	0.17	0.00	0.02	0.02	0.03	0.15	0.15	0.27	1.00	0.00
K	0.04	0.00	0.25	0.18	0.18	0.04	0.02	0.02	0.00	0.00	1.00

From Table 4, we can see that the coupling relationship between terms considering both of intra- and inter-coupling is more reasonable than that of only considering either intra-coupling or inter-coupling of terms. For example, we consider a pair of terms "<Title, XML>" (denoted by D) and "<Title, semi-structured data>" (denoted by J) in Table 4. If we only consider their intra-coupling, there is no relationship between them as showed in Table 2. But in reality, "XML" and "semi-structured data" is related to each other in semantic and the relationship between them can be captured by our inter-coupling calculating algorithm. As a result, the coupling relationship between them would not be zero as showed in Table 4.

5 Top-*k* Semantically Related Term Selection

To find the top-k related terms, a simple way is to compute the sum of coupling relationship between a term and each query keyword. This section describes an alternative algorithm to facilitate the top-k related term selection. The algorithm consists of two steps. The first step is to create the order (i.e., ranking list) of all terms in database for each query keyword according to their coupling relationships to the query keyword. In this paper, we assume each query keyword can be mapped into a specific database term, and then we can leverage the coupling relationships between terms to create the orders for each query keyword. The second step is to use threshold algorithm (TA) to select the top-k related terms based on these orders.

Step 1. Create orders for terms. For each query keyword k_i (suppose it corresponds to a database term t_i), create an order τ_i

of all terms (except t_i) in database in descending order, according to their coupling relationships to t_i (i.e., k_i). The terms in order τ_i can be divided into two sets. The one is the relevant set, where the terms have coupling relationships to k_i. The other one is the irrelevant set, where the terms have no coupling relationship to k_i. Firstly, it is natural to assume that an irrelevant term is less important than a relevant one, thus the terms of irrelevant set should be ranked after the terms of relevant set in the order. Secondly, since there is no evidence to show that one of them in irrelevant set is more or less important than the other one, they are positioned randomly in the irrelevant set of the order.

Since there are totally l query keywords, the output of this procedure is a set of l orders. According to the output orders, each term t_j has a score that is associated with the position of t_j in order τ_i. The score of t_j in τ_i that corresponds to keyword k_i is:

$$s(t_j|k_i)=n-\tau_i(t_j)+1 \tag{15}$$

in which, $\tau_i(t_j)$ represents the position of t_j in τ_i.

Step 2. Select top-*k* related terms. For a set of query keywords Q, and the set of all distinct terms T in database D, using the output of Step 1, this step computes the set $Q_k(T) \subseteq T$ with $|Q_k(T)|=k$, such that $\forall t_j \in Q_k(T)$ and $t_j' \in \{T-Q_k(T)\}$ it holds that $score(t_j,Q) > score(t_j',Q)$, with $score(t_j,Q) = \sum_{i=1}^{l} s(t_j|k_i)$, where $s(t_j|k_i)$ is computed by using the Equation (14).

The Threshold Algorithm (TA) [22] is employed to find the top-k relevant terms for a set of given keywords. The TA uses *Sorted* and *Random* modes to access the terms in the orders. The *Sorted access* mode obtains the score of a term in an order by traversing the order of the terms sequentially from the top. The *Random access* mode obtains the score of a term in an order in one access. The *threshold* is set as the sum of the score of last visited term from each order for the current round-robin.

The top-k related term selection algorithm works as follows.

1. Accessing each one of the l orders of the terms in a round-robin. As a term t is seen in some order τ_i that corresponds to query keyword k_i, get the score of term t from every other orders $\{\tau_j|\tau_j \in O_l$ and $j \neq i\}$ by using the random access. The final score of term t for the set of query keywords Q is computed as:

$$score(t,Q) = \sum_{i=1}^{l} s(t|k_i) \tag{16}$$

2. The threshold λ for the j-th round-robin cycle is defined as the sum of the score of last visited term from each order, that is,

$$\lambda = \sum_{i=1}^{l} s(\underline{t_j}|k_i) \tag{17}$$

where, $s(\underline{t_j}|k_i)$ denotes the score of last visited term of order τ_i ($i \in \{1,\ldots,l\}$) by the end of the j-th round-robin cycle. The algorithm terminates when k terms with score values greater or equal to the threshold λ.

3. Output the k terms among the set of all found terms with the highest value for $score(t,Q)$.

Using the Algorithm above, a list of top-k related terms can be returned for a set of given query keywords. Then, user can choose term in the list to explore the database and view the results of the related terms in the list.

6 EXPERIMENTS

6.1 Experimental Settings

The experiments are conducted on a computer running Windows 2007 with Intel P4 3.2-GHz CPU, and 8 GB of RAM. All algorithms are developed in C# and SQL. We use the DBLP dataset to evaluate the performance of our methods. The download DBLP XML file is decomposed into several relations according to the schema showed in Figure 4.

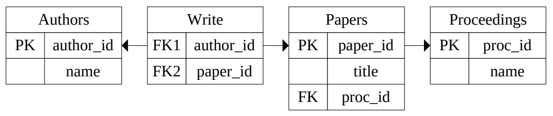


Figure 4.The DBLP schema (PK refers to primary key and FK refers to foreign key)

We select 1000 tuples from the four relations of the DBLP database as our testing dataset, and these tuples can be connected though primary-foreign-key references based on the database schema showed in Figure 4. After the text decomposition and segmentation, there are totally 1674 distinct terms extracted from the dataset. Based on the dataset, we next show the efficiency and performance of our term coupling relationship measuring and top-k related term selection methods.

6.2 Precision of Term Coupling Relationships

This experiment aims to test the precision of our term coupling relationship measuring method (short for TCR) that corresponds to different parameter value of α in Equation (14). To verify the accuracy of the TCR method, we adopt the strategy as follows. We invited 10 people, which are researchers and PhD students, to randomly choose 10 terms from the DBLP dataset. And then, for each term t_i, we obtained top 5 terms by using our TCR method with respect to each value of parameter α in Equation (14) from 0 to 1 at the increments 0.1. After this, these terms are mixed together and a set K_i of 55 terms is generated consequently. We invited 10 people, which are researchers and PhD students. Lastly, we presented the terms with their corresponding K_i's to each user in our study. The task of each user is to mark the top 5 terms that they considered semantically related to t_i. We then measured how closely the 5 terms marked as relevant by the user matched the 5 terms returned by each algorithm. The users were asked to describe whether they considered a term t' related to a given term t based on:

(i) the terms t' and t are same or similar in semantic, such as the term "semi-structured data" and "XML data".

(ii) the terms t' and t are related in semantic, for example, the term "association rules" is usually associated with the term "apriori algorithm", hence they are considered to be related.

Figure 5 illustrates the *precision* in estimating the top 5 terms obtained by using our method with respect to different values of α. The precision is calculated as the number of terms retrieved among the top 5 terms that were marked as relevant, i.e., $Precision = \frac{|relevant\ terms\ retrieved|}{5}$. Note that, the precision for each value of α is averaged over 10 selected terms.

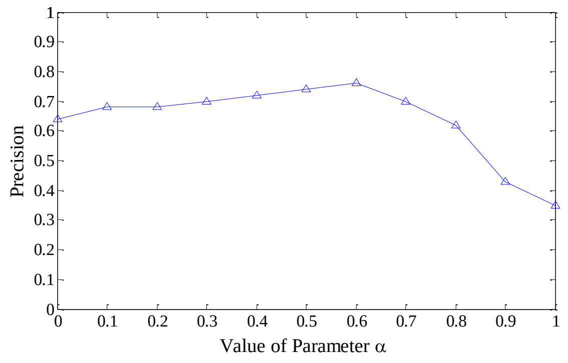


Figure 5. Precision of relevant terms retrieved for different values of α on DBLP dataset

It can be seen that the curve of precision reaches a peak at α=0.5 for DBLP dataset, which demonstrates that our method performs best when α is set to 0.6 (the corresponding accuracy is 0.76). It also can be seen that, the precision raises as the parameter α increases from 0 to 0.6, which indicates the inter-coupling play a positive role on the coupling relationship measuring. While, the precision declines as the parameter α after 0.6, which means the inter-coupling brings negative impact into the coupling relationship measuring. It should be pointed out that the term coupling relationships depend on the dataset, and thus it is essential to optimize the setting of α to achieve the highest precision for different datasets.

As mentioned in Section 2, the work that is most similar to ours is the FCT algorithm in [19], which finds the most frequent term co-occurring with the query keywords in the MTJNTs. In this experiment, the number of query keyword in the query is only one (each selected term is treated as a query), which makes the precision of FCT and TCR (when $\alpha = 0$) equal, i.e., 0.64. It is clearly that the precision of TCR (when $\alpha = 0.6$) is much higher than that of FCT over the DBLP dataset. This is because FCT only discover the relevant terms that frequently co-occur with the query keywords. In contrast, TCR considers both the term co-occurred and inter-related relationships, which can better reveal the explicit and implicit correlations between terms. Hence, the answers of TCR can meet the user's intentions more closely.

6.4 Execution Performance

This experiment aims to verify the execution time of the top-*k* related term selection algorithm. There are two parameters, *l* and *k*, in this algorithm, where *l* represents the number of query keywords and *k* denotes the number of terms needs to be selected. We fix the number of *l* to 2, 4, 6, 8, and 10, respectively and then test the execution time for different *k* values. Figure 7 illustrates the execution time on DBLP dataset for different *k* values when *l*={2, 4, 6, 8, 10}.

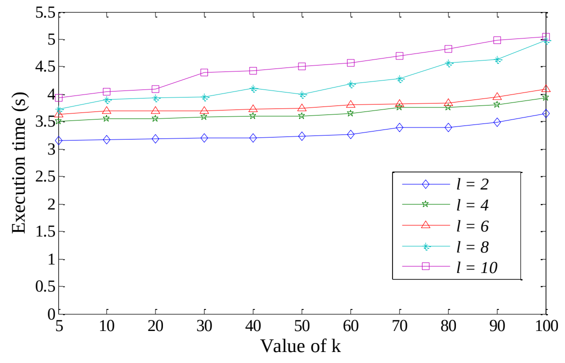


Figure 6. Execution time of top-*k* selection algorithm

From Figure 6, it can be seen that the performance of the algorithm decreases with the increasing of number *l* and *k*. This is because the top-*k* related term selection algorithm needs to deal with more terms in orders as the number *l* and *k* increased. We also computed the time consumption for computing the sum of coupling relationship between a term and each query keyword. It takes approximately 125 seconds for DBLP dataset. Our top-*k* related term selection algorithm clearly outperforms existing methods and demonstrates more efficient performance.

7 CONCLUSIONS

This paper presented a novel approach, which leverages the coupling relationships between terms in database, to find the top-*k* semantically related terms for a set of query keywords. Based on the database schema, a data view is first created by connected tuples of relations through primary-foreign-key references. The term coupling relationship is then estimated by considering both the intra- and inter-coupling between terms within and across the tuples of the view. For a set of given query keywords, the orders of all terms in database is generated for each query keyword according to the coupling relationships between terms and query keyword, and then TA algorithm is used to quickly find the top-*k* related terms based on these orders. The experiments on real dataset identified that the term coupling relationship method can capture the semantic relationships of terms more reasonable and the top-*k* related term selection algorithm also achieves high performance. It would be interesting to investigate how to minimize the updating cost when the database is varied.

REFERENCES

[1] Aditya, B., Bhalotia, G., Chakrabarti, S., and Hulgeri, A. 2002. Banks: browsing and keyword searching in relational databases. In *Proceedings of the VLDB Conference*, 1083-1086.

[2] Agrawal, S., Chaudhuri, S., and G. Das. Dbxplorer: A system forkeyword-based search over relational databases. 2002. In*Proceedings of the ICDE Conference*, 5-16.

[3] Hristidis, V. and Papakonstantinou, Y. 2002. Discover: keyword search in relational databases. In *Proceedings of the VLDB Conference*, 670-681.

[4] Hristidis, V., Gravano, L., and Papakonstantinou, Y. 2003. Efficient IR-style keyword search over relational databases. In *Proceedings of the VLDB Conference*, 850-861.

[5] Tata, S. and Lohman, G. M. 2008. SQAK: doing more with keywords. In *Proceedings of the ACM SIGMOD Conference*, 889-902.

[6] Cao, L. B., Ou, Y. M., Yu, Philip S. 2012. Coupled behavior analysis with applications. *IEEE Trans. Knowl. Data Eng.* 24(8): 1378-1392.

[7] Cheng, X., Miao, D. Q., Wang, C., and Cao, L. B. 2013. Coupled term-term relation analysis for document clustering. *In Proceedings of the IJCNN Conference*, 1-8.

[8] Wang, C., Cao, L. B., Wang, M. C., Li, J. J., Wei, W., and Ou, Y. M. 2011. Coupled nominal similarity in unsupervised learning. *In Proceedings of the CIKM Conference*, 973-978.

[9] Wang, X. and Sukthankar, G. 2013. Multi-label relational neighbor classification using social context features. *In Proceedings of the ACM KDD Conference*, 464-472.

[10] Wang, C., She, Z., and Cao, L. B. 2013. Coupled clustering ensemble: incorporating coupling relationships both between base clusterings and objects. *In Proceedings of the ICDE Conference*, 374-385.

[11] Yao, J. J., Cui, B., and Hua, L. S. 2012. Keyword query reformulation on structured data. *In Proceedings of the ICDE Conference*, 953-964.

[12] Ding, B., Yu, J. X., and Wang, S. 2007. Finding top-k min-cost connected trees in databases. *In Proceedings of the ICDE Conference*, 468-477.

[13] Luo, Y., Lin, X. M., and Wang, W. 2007. SPARK: top-k keyword query in relational databases. *In Proceedings of the ACM SIGMOD Conference*, 305-316.

[14] Zhou, R., Liu, C. F., and Li, J. X. 2010. Fast ELCA computation for keyword queries on XML data. 2010. *In Proceedings of the EDBT Conference*, 549-560.

[15] Kong, L. B., Gilleron, R., and Lemay, A. 2009. Retrieving meaningful relaxed tightest fragments for XML keyword search. *In Proceedings of the EDBT Conference*, 815-826.

[16] Bao, Z. F., Lu, J. H., and Ling, T. W. 2010. XReal: an interactive XML keyword searching. *In Proceedings of the CIKM Conference*, 1933-1934.

[17] Sarkas, N., Bansal, N., and Das, G. 2009. Measure-driven keyword query expansion. *The Proceedings of the VLDB Endowment*, 2(1):121-132.

[18] Bergamaschi, S., Domnori, E., and Guerra, F. 2011. Keyword search over relational databases: a metadata approach. *In Proceedings of the ACM SIGMOD Conference*, 565-576.

[19] Tao, Y. F. and Yu, J. X. Finding frequent co-occurring terms in relational keyword search. 2009. *In Proceedings of the EDBT Conference*, 839-850.

[20] Liu, F., Yu, C., and Meng, W. Y. 2006. Effective keyword search in relational database. *In Proceedings of the ACM SIGMOD Conference*, 563-574.

[21] Li, G. L., Feng, J. Y., and Zhou, L. Z. 2008. Retune: retrieving and materializing tuple units for effective keyword search over relational databases. *In Proceedings of the ER Conference*, 469-483.

[22] Fagin, R., Lotem, A., and Naor, M. 2001. Optimal aggregation algorithms for middleware. *In Proceedings of the PODS Conference*, 102-113.