Semantic Approximate Keyword Query Based on Keyword and Query Coupling Relationship Analysis

Xiangfu Meng
School of Electronic and Information Engineering
Liaoning Technical University
Huludao, China
marxi@126.com

Longbing Cao
Advanced Analytics Institute
University of Technology, Sydney
Sydney, Australia
longbing.cao@uts.edu.au

Jingyu Shao
Advanced Analytics Institute
University of Technology, Sydney
Sydney, Australia
jingyu.shao@student.uts.edu.au

ABSTRACT

Due to imprecise query intention, Web database users often use a limited number of keywords that are not directly related to their precise query to search information. Semantic approximate keyword query is challenging but helpful for specifying such query intent and providing more relevant answers. By extracting the semantic relationships both between keywords and keyword queries, this paper proposes a new keyword query approach which generates semantic approximate answers by identifying a set of keyword queries from the query history whose semantics are related to the given keyword query. To capture the semantic relationships between keywords, a semantic coupling relationship analysis model is introduced to model both the $intra-$ and $inter-keyword\ couplings$. Building on the coupling relationships between keywords, the semantic similarity of different keyword queries is then measured by a semantic matrix. The representative queries in query history are identified and then a priori order of remaining queries corresponding to each representative query in an off-line preprocessing step is created. These representative queries and associated orders are then used to expeditiously generate top-k ranked semantically related keyword queries. We demonstrate that our coupling relationship analysis model can accurately capture the semantic relationships both between keywords and queries. The efficiency of top-k keyword query 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

CIKM'14, November 3–7, 2014, Shanghai, China.

ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2661867 .

Keywords

Web database, keyword query, coupling relationship analysis, top-k selection

1. INTRODUCTION

With the increasing of complexity and size of Web databases[1], keyword search plays an important role in obtaining the information needed from Web databases. In practice, however, it is difficult for lay users to obtain complete and effective information since average people usually have insufficient knowledge about the structure and contents of Web databases. Accordingly, one often has imprecise ideas about what exact keywords he/she should use for searching and finds it hard to formulate an appropriate query by using only a few keywords. As a result, an inadequate answer, or no answer, is often returned when the query is too selective or query keywords are not properly selected. In such a context, a user has to reformulate queries several times before meaningful query results are received, which is often a time-consuming exercise. Therefore, it is important to produce a list of queries that are semantically related to the original query so that a user can select and view the answers to a query by choosing it from a proper list. Additionally, providing related queries is also very helpful for the scientific database users, especially the people who is unfamiliar with a new research field.

The challenge in selecting semantically related queries is to understand the semantics of the original query and to measure the semantic similarity between them. Several approaches have been proposed to deal with the issue of keyword search over relational databases [1, 3, 14, 15, 20]. Their basic idea is to extract a set of joining trees of tuples. A joining tree of tuples is formed by matched tuples, which contain the specified keywords in their text attributes, are interconnected through primary-foreign-key references. However, most of the existing work neglects the coupling relationships between keywords when searching the joining tuple trees, rather treating them independent. As a result, the semantic coupling between keywords is overlooked.

However, in the real world, there are various coupling relationships [8] existing between objects, which have been shown valuable to be incorporated into analysis such as document term semantic analysis [10], clustering [21] and classification [23]. Similarly, keywords embedded in a query are

[1]Web database refers to a non-local online database that can be accessible by a web form based interface.

coupled in terms of co-occurrences and semantic relationships. The meaning of a keyword is often associated with the meaning of the others, we call $intra-couplings$ between keywords in a query. The semantically connected keyword set in a query jointly express the user query intention. In addition, coupling relationships also exist between keywords from different queries, we call $inter-couplings$ of keywords. Such cross-query keyword relationship contributes to the matching between a query and others. On top of this observation, in this paper, we propose a new keyword query approach which incorporates the keyword semantic couplings for approximate query. It incorporates the coupling analysis [22] into keyword coupling relationship and query semantic similarity analysis. It then leverages the query semantic similarities to extract the top-k queries from query history that are related to a given query. We will use the toy example below to motivate and provide an overview of our approach.

Example 1. As shown in Figure 1, a DBLP database consisting of three relations: $Authors$, $Papers$ and $Write$, connected by primary-foreign-key relationships.

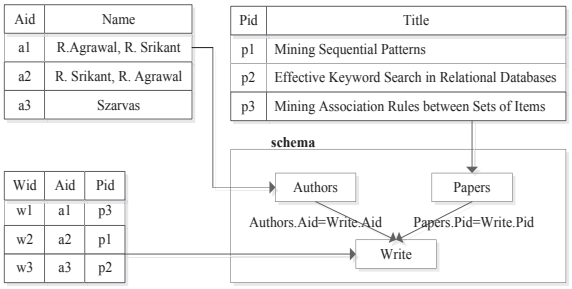

Figure 1: Example of relational database DBLP

A user may issue the following keyword queries:
Q_1:-DBLP(R. Agrawal, sequential patterns)
Q_2:-DBLP(association rules, apriori algorithm)

On receiving the query Q_1, a classic keyword search approach may provide a set of joining tree of tuples containing keywords "R. Agrawal" and "sequential patterns" as the query results. For example, a joining tree of tuples $a_2 \bowtie w_2 \bowtie p_1$ is an answer for query Q_1. It is clear that there exists a relationship between keywords "R. Agrawal" and "sequential patterns" because they appear in the same query. The query mostly probably indicates that the user wants to find the paper related to "sequential patterns" proposed by the author "R. Agrawal". In real applications, the user who submitted query Q_1 may also be interested in tuples containing "association rules" since the "sequential patterns" and "association rules" are proposed by the same author (i.e., R. Agrawal), and they are much related with each other in the data mining research field. However, the traditional keyword search approaches cannot provide the answer containing keywords "association rules" such as $a_1 \bowtie w_1 \bowtie p_3$ returned by Q_2. This example shows coupling relationships exist between keywords in a query and between queries.

This paper proposes a solution for extracting keyword semantic approximate query results by selecting the top-k queries from query history that are semantically related to a given query. We first capture the intra- and inter-couplings between different pairs of keywords extracted from *query history* - log of past users keyword queries. The intra- and inter-couplings are then combined to generate the keyword coupling relationship. Note that, although keyword coupling relationships can be captured from other data sources, in this paper we only rely on the query history that is directly related to user query intentions. With such keyword coupling relationships, we further measure the semantic similarity between different queries by building a semantic matrix, where the coupling relationships between keywords are reserved. As a result, given a query, the system provides a list of k queries from query history that are related to the given query, and a user can view the answers of related query by choosing it in the list.

Our contributions are summarized as follows:

(1). A novel method is proposed to measure the keyword coupling relationships, which considers both the intra- and inter-couplings between different keywords within and across the queries.

(2). A new keyword query similarity metric based on a semantic matrix is proposed, in which the keyword coupling relationships are reserved.

(3). A top-k query selection algorithm, which is used to quickly select top-k related queries from query history, is presented.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 outlines an overview of our framework. Section 4 proposes the keyword coupling relationship measuring method while Section 5 describes the query semantic similarity measuring method. Section 6 presents a top-k query selection algorithm. The experiment results are presented in Section 7. The paper is concluded in Section 8.

2. RELATED WORK

Several methods have been proposed to handle keyword search in relational and XML database systems, and the popularity of keyword search is ongoing [24]. For the relational database, the related work can be classified into two main categories. The first is mainly based on Steiner trees, such as BANKS [1] and its extensions [12, 20]. These approaches 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 minimal joining trees of tuples containing the query keywords. The second leverages Candidate network (CN), such as DBXplorer [3], DISCOVER [15], and SPARK [18], to find the relevant answers. A candidate network is a joining network of tuples(JNTs), in which the tuples are inter-connected through primary-foreign-key relationships. The CN-based approaches generate all possible candidate networks following the database schema, and then identify a set of minimal total joining network of tuples (MTJNTs) based on CNs. For the XML database, the lowest common ancestors (LCAs) [26] and its extensions [5, 17] are used for keyword search. In summary, the existing approaches mainly focus on finding MTJNTs or LCAs 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 or LCAs may also be very relevant to a query in semantic terms, even though they do not explicitly contain the query keywords.

In recent years, tentative work on keyword semantic understanding and approximate query has been undertaken.

In [19], the transformation rules are manually defined used for keyword query integration and the local results are analyzed used for finding relevant answers. In [6], the metadata of database is used for translating keyword queries into meaningful SQL queries that describe the intended query semantics. In [24], the data structural semantics are exploited and employed to reformulate the initial query. Although keyword semantics have been taken into consideration, most of the existing approaches usually assume that keywords in a query are independent of one another, but in reality coupling relationships exist between objects such as keywords and terms as shown in [8, 10].

The concept of coupling relationship has recently been introduced to cater for the interactions within and between attributes. A number of studies [8, 10, 21] have proven to be very effective for capturing the implicit relationships for machine learning and data mining tasks such as clustering and document analysis. In this paper, we incorporate this idea into keyword coupling and query similarity analysis and then leverage the query similarity to find the top-k queries from query history that are related to a given query. The top-k retrieval problem has been studied in several situations, such as the view-based top-k query against the relational database [11] and the top-k preferences retrieval in context of high dimensions [25]. Given a set of objects O, the basic idea of top-k retrieval is to quickly find the k objects in O with the highest scores with respect to a given query by considering the monotonic ranking functions defined on a subset of the attributes of the set O. Note that, keyword search issues are not investigated in this paper. Our focus is on keyword coupling relationship analysis, query semantic similarity measure and top-k related query selection.

3. FRAMEWORK

This paper proposes a two-step processing approach to address this problem. The framework is shown in Figure 2.

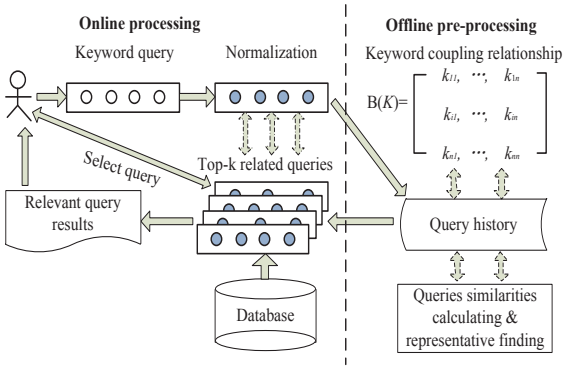

Figure 2: Framework of keyword semantic approximate query

The first step occurs offline. It analyses query history of all users already in the system and then captures the coupling relationships between keywords. Firstly, all distinct keywords in query history are extracted, following which the intra- and inter-couplings between different pairs of keywords can be calculated by leveraging the correlation analysis method. Consequently, the keyword intra- and inter-coupling is combined into a coupling relationship to reflect the semantic relevance between keywords. Based on the keyword coupling relationship, we further use the semantic matrix to measure the similarity between keyword queries. To reduce the online computation time, a few representative queries are selected and the remaining queries in query history are formed into several ordered lists, hereafter called "orders". Each order corresponds to a representative query and the queries in each order are ranked according to their similarities to that representative query.

Table 1: Example of a pruned query history

UID	QID	Keywords
U_1	Q_{13}	classification, clustering, KDD
U_2	Q_{24}	association rules, clustering, data analysis
U_3	Q_{35}	association rules, decision tree, prediction
U_4	Q_{43}	classification, decision tree, KDD

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 keywords, it then computes the semantic similarities between the given query and representative queries. Lastly, a list of top-k related queries is returned using Threshold Algorithm (TA) and priori orders. The user can view the results of the related query by selecting it from the list.

4. KEYWORD COUPLING RELATIONSHIP ANALYSIS

Building on the term coupling analysis in document analysis [10], to evaluate the keyword coupling relationship, the query history is used as the knowledge source in this paper. This section first introduces the query history and its prune strategy, and then presents how to measure the keyword coupling relationships based on the query history.

4.1 Query History

Definition 1. (Keyword query): A input keyword query Q over database D is an ordered list of distinct keywords, i.e., $Q = \{k_1, k_2, ..., k_m\}$, each k_i $(i = 1, ..., m)$ in Q is a word or a topical phrase, depending on the decomposition.

Definition 2. (Query history): A query history W is consisted of $\{(U_1, Q_1, K_1), ..., (U_n, Q_n, K_n)\}$ in chronological order, where U_i is a session ID (a session is a duration started from a user connects to the Web database and ended to the user disconnects), Q_i is a query ID, and K_i is a query keywords list.

To guarantee the quality of query history, the following prune strategy is used: (1) remove the queries with empty results; (2) reserve only the keyword query that is most related to the query intention. A user may issue several queries in one session that progress from being general in nature to being more concrete, finally stopping at a query that returns meaningful results. Therefore, it is the last keyword query in such a refinement sequence that is most related to the query intention and should be reserved [9]; (3) decompose the remaining queries into several distinct keywords and normalize them using text split and analysis tools such as AlchemyAPI and Wikipedia [16] (although it would

be interesting to process the natural language and normalize the keywords, it is beyond the scope of this paper). An example of pruned query history is shown in Table 1.

4.2 Keyword Intra-coupling within a Query

In Information Retrieval, two terms are considered semantically related if they frequently co-occur in the same documents of the document set. Similarly, each keyword query in query history is considered as a document. Then, the frequency of co-occurrence of a pair of keywords (k_i, k_j) appearing in the same queries of query history can be measured by Jaccard coefficient [7] as follows,

$$J(k_i, k_j) = \frac{|W(k_i) \cap W(k_j)|}{|W(k_i) \cup W(k_j)|} \quad (1)$$

where, $W(k_i)$ and $W(k_j)$ represent the queries in query history in which k_i and k_j appears, respectively. Based on Equation (1), we can give the definition of $intra-coupling$ of keywords.

Definition 3. (Intra-coupling of keywords): Keywords k_i and k_j are intra-related if they co-occur in at least one keyword query Q_i ($Q_i \in W$), the intra-coupling between them in W is defined as,

$$\delta_{IaR}(k_i, k_j|W) = J(k_i, k_j) \quad (2)$$

where, $J(k_i, k_j)$ is defined as Equation (1).

Since keyword k_i or k_j may also co-occur with other keywords in the queries, we need to normalize the intra-coupling between k_i and k_j by dividing the total number of intra-couplings between k_i and other keywords. Thus, the intra-coupling between k_i and k_j is finally computed as follows,

$$\delta_{IaR}(k_i, k_j) = \begin{cases} 1 & i = j \\ \frac{\delta_{IaR}(k_i, k_j|W)}{\sum_{a=1, a\neq i}^{n} \delta_{IaR}(k_i, k_a|W)} & i \neq j \end{cases} \quad (3)$$

where, n is the number of all distinct keywords in W.

For each pair of keywords (k_i, k_j), we have $\delta_{IaR}(k_i, k_j) \geq 0$ and $\sum_{j=1, j\neq i}^{n} \delta_{IaR}(k_i, k_j) = 1$. Note that, the values of $\delta_{IaR}(k_i, k_j)$ and $\delta_{IaR}(k_j, k_i)$ may not be equal to each other due to the different dominators. The keyword intra-coupling relationship calculating algorithm is shown in Algorithm 1. Note that, since $J(k_i, k_j) = J(k_j, k_i)$, the matrix of $\delta_{IaR}(k_i, k_j|W)$ is symmetric, therefore we need to only compute the upper-half of the matrix of $\delta_{IaR}(k_i, k_j|W)$ in Algorithm 1 (line 4-5). Table 2 shows the intra-coupling matrix of keywords extracted from Table 1. For simplicity, we use CA, CU, KD, AR, DA, DT, and PR to denote the keywords *classification*, *clustering*, *KDD*, *association rules*, *data analysis*, *decision tree*, and *prediction*, respectively.

The intra-coupling reflects the correlation between the keywords in case of they are co-occurring in the same queries. Besides the intra-coupling, the keywords may also be inter-related via their *common keywords* across queries. In particular, the keywords, which have never co-occurred in the same queries (that means they only appeared in separate queries), may also inter-related in semantic, such as "classification" and "association rules" are inter-related by the keyword "clustering" and "decision tree". In this paper, we say this type of correlation between keywords is $inter-coupling$ of keywords. We next present how to capture the inter-coupling between keywords.

Algorithm 1: Keyword intra-coupling calculation

Input: query history W, set of all distinct keywords K extracted from W, number of keywords 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][j]=J(K[i], K[k]).
5          IaRMatrix[k][i]=IaRMatrix[i][k].
6      for m = 1 to n do
7          if m ≠ i then
8              Sum=Sum+IaRMatrix[i][m].
9      for j = 1 to n do
10         if j ≠ i then
11             IaRMatrix[i][j]=IaRMatrix[i][j]/Sum.
12 Return IaRMatrix.
```

Table 2: Example of keyword intra-coupling matrix

	CA	CU	KD	AR	DA	DT	PR
CA	1.00	0.20	0.60	0.00	0.00	0.20	0.00
CU	0.22	1.00	0.22	0.22	0.33	0.00	0.00
KD	0.60	0.20	1.00	0.00	0.00	0.20	0.00
AR	0.00	0.20	0.00	1.00	0.30	0.20	0.30
DA	0.00	0.50	0.00	0.50	1.00	0.00	0.00
DT	0.22	0.00	0.22	0.22	0.00	1.00	0.33
PR	0.00	0.00	0.00	0.50	0.00	0.50	1.00

4.3 Keyword Inter-coupling across Queries

The basic idea for capturing the inter-coupling between keywords is that if the sets of keywords co-occurred with k_i and k_j are partly overlapped we say k_i and k_j are inter-related.

Given a keyword k_i, all the keywords co-occurred with k_i in query history can be seen as the features associated with k_i. The inter-coupling between two keywords can be estimated by the commonality in the features associated with them. For example, given a keyword "classification" in Table 1, a set of keywords "clustering, KDD, decision tree" is associated with it; while, a set of keywords "clustering, data analysis, decision tree, prediction" is associated with the keyword "association rules". Clearly, the overlapped keywords between two sets are "clustering" and "decision tree". In this paper, we say these are common keywords, which mean that two keywords occurring in different queries are inter-related via their common keywords. According to this, the inter-coupling between k_i and k_j via a common keyword k_c can be defined as follows.

Definition 4. (Inter-coupling of keywords): Keywords k_i and k_j are inter-related if there is at least one common keyword k_c such that $\delta_{IaR}(k_i, k_c) > 0$ and $\delta_{IaR}(k_j, k_c) > 0$ hold but keywords k_i and k_j appear in different queries. The inter-coupling between keywords k_i and k_j via common keyword k_c is defined as,

$$\delta_{IeR}(k_i, k_j|k_c) = min\{\delta_{IaR}(k_i, k_c), \delta_{IaR}(k_j, k_c)\} \quad (4)$$

where, $\delta_{IaR}(k_i, k_c)$ and $\delta_{IaR}(k_j, k_c)$ are the intra-couplings between k_i and k_c, k_j and k_c, respectively.

Algorithm 2: Keyword inter-coupling calculating algorithm

Input: set of all keywords K in W, number of keywords n, IaRMatrix, weight of each keyword in K.
Output: IeRMatrix.

```
IeRMatrix=null.
for i = 1 to n − 1 do
    for j = 1 to n do
        S ← the set of common keywords between K[i] and K[j].
        m = |S|.
        if S = φ then
            IeRMatrix[i][j]=0.
        else
            for k = 1 to m do
                minvalue=min{δ_IaR(K[i], S[k]),
                             δ_IaR(K[j], S[k])}.
                sum+= minvalue*w(S[k]).
            IeRMatrix[i][j]=sum/m.
Return IeRMatrix.
```

It should be pointed out that there is usually more than one common keyword between k_i and k_j and each one may have different importance/weight in query history. Thus, it is necessary to measure the importance of the common keyword. The intuition is that the greater the frequency of the keyword occurring in query history, the greater the number of the users interested in it, and thus the more important the keyword is. A method that leverages this intuition is to count the frequencies of keywords appearing in query history, and then allow important coefficients to depend on these frequencies. Let $QF(k_i)$ represents the frequency of occurrence of keyword k_i in query history and $QFMax$ the frequency of the most frequently occurring keyword. Consequently, the weight of k_i, $w(k_i)$ can be defined as,

$$w(k_i) = \frac{QF(k_i)}{QFMax} \tag{5}$$

We then let S be the set of common keywords for k_i and k_j, that is, $S = \{k_c | (\delta_{IaR}(k_i, k_c) > 0 \wedge \delta_{IaR}(k_j, k_c) > 0)\}$. Then, the inter-coupling between k_i and k_j, inter-related by all the common keywords in S, can be formalized as,

$$\delta_{IeR}(k_i, k_j) = \begin{cases} 1 & i = j \\ \dfrac{\sum_{\forall k_c \in S} w(k_c) * \delta_{IeR}(k_i, k_j | k_c)}{|S|} & i \neq j \end{cases} \tag{6}$$

where, $w(k_c)$ is computed by Equation (5), $\delta_{IeR}(k_i, k_j|k_c)$ is the inter-coupling between k_i and k_j related via the common keyword k_c, and $|S|$ denotes the number of common keywords in S. Equation (6) means the inter-coupling between k_i and k_j is measured by the average inter-couplings between k_i and k_j via all of their's common keywords. If $S = \phi$, then $\delta_{IeR}(k_i, k_j)$ is zero. Note that, the correlation between two co-occurring keywords is also enhanced by their inter-coupling relationship. The keywords inter-coupling calculating algorithm is shown in Algorithm 2. Table 3 shows the inter-coupling matrix of keywords extracted from Table 1.

Table 3: Example of keyword inter-coupling matrix

	CA	CU	KD	AR	DA	DT	PR
CA	1.00	0.00	0.00	0.20	0.20	0.00	0.20
CU	0.00	1.00	0.00	0.00	0.00	0.22	0.22
KD	0.00	0.00	1.00	0.20	0.20	0.00	0.20
AR	0.20	0.00	0.20	1.00	0.00	0.00	0.00
DA	0.20	0.00	0.20	0.00	1.00	0.22	0.50
DT	0.00	0.22	0.00	0.00	0.22	1.00	0.00
PR	0.20	0.22	0.20	0.00	0.50	0.00	1.00

Table 4: Keyword coupling relationship matrix

	CA	CU	KD	AR	DA	DT	PR
CA	1.00	0.10	0.30	0.10	0.10	0.10	0.10
CU	0.11	1.00	0.11	0.11	0.17	0.11	0.11
KD	0.30	0.10	1.00	0.10	0.10	0.10	0.10
AR	0.10	0.10	0.10	1.00	0.15	0.10	0.15
DA	0.10	0.25	0.10	0.25	1.00	0.11	0.25
DT	0.11	0.11	0.11	0.11	0.11	1.00	0.17
PR	0.10	0.11	0.10	0.25	0.25	0.25	1.00

4.4 Keyword Coupling Relationship

The coupling relationship between two keywords k_i and k_j is the combination of intra- and inter-coupling of the two keywords, which is defined keyword coupling as follows,

$$\delta_{SR}(k_i, k_j) = \begin{cases} 1 & i = j \\ (1-\alpha) * \delta_{IaR}(k_i, k_j) + \alpha * \delta_{IeR}(k_i, k_j) & i \neq j \end{cases} \tag{7}$$

where, $\alpha \in [0, 1]$ is the parameter to determine the weight of intra- and inter-coupling. It is clearly that the higher the coupling relationship, the more related is the two keywords. Note that, the Equation (7) would be intra-coupling if $\alpha = 0$ while it would be inter-coupling if $\alpha = 1$, that means the intra- and inter-coupling are the special cases of keyword coupling relationship.

Table 4 shows the coupling relationship matrix of all keywords extracted in Table 1. Here, we set α to 0.5, which means the intra- and inter-coupling play the same important role in measuring the keyword coupling relationship. From Table 4, we can see that the coupling relationship between keywords considering both of intra- and inter- coupling of keywords is more reasonable than that of only considering either intra-coupling or inter- coupling of keywords. For example, we consider a pair of keywords "classification" and "prediction" (or "data analysis") in Table 1. If we only consider their intra-coupling, there is no relationship between them as showed in Table 2. But in reality, "classification" and "prediction" (or "data analysis") is closely 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. KEYWORD QUERY SEMANTIC SIMILARITY ANALYSIS

In information retrieval, cosine similarity is a commonly used similarity measure, defined on Vector Space Model. In this paper, each query can be treated as a document and the keyword can be treated as a term. Thus, we can adopt the

cosine similarity to quantify the semantic similarity between queries. The solution consists of the following 3 steps.

Step 1. Convert the keyword query into vector representation. Given a pair of queries Q_{i1} and Q_{i2}, we assume K be the set of all distinct keywords in Q_{i1} and Q_{i2} and n the number of keywords in K. We also let $n = |K|$ and Δ be an arbitrary but fixed order on the keywords appearing in K. $K[i]$ refers to the i-th keyword of K based on the order Δ. In the context of Q_{i1} and Q_{i2}, a vector representation of $Q_{i1} = \bigwedge_j K[j](j = 1, ..., n)$ is a binary vector $\overrightarrow{Q}_{i1}$ of size n. The i-th element of the vector corresponds to keyword $K[i]$. If $K[i]$ appears among the keywords of Q_{i1} then $\overrightarrow{Q}_{i1}[i] = 1$, otherwise it is 0. Note that, since different pairs of queries usually contain different number of keywords, the cardinality of K is finite and varies depending on the compared queries.

For example, the queries Q_{35} and Q_{43} in Table 1 totally have five distinct keywords. We assume the order on them is $association\ rules$, $decision\ tree$, $prediction$, $classification$, and KDD. Then, the query Q_{35} and Q_{43} can be represented by the vector [1 1 1 0 0] and [0 1 0 1 1], respectively.

Step 2. Construct the semantic matrix. Given a pair of queries Q_{i1} and Q_{i2}, we also assume K be the set of all distinct keywords in Q_{i1} and Q_{i2} and n the number of keywords in K. The coupling relationships of all keywords in K can then be transformed into a semantic Matrix S_K, which is a $n * n$ matrix and each element $S_K(i, j)$ in it corresponds to the coupling relationship between keywords k_i and k_j.

Step 3. Compute the semantic similarity between queries. The traditional VSM-based cosine similarity measuring method assumes the keywords are independent in queries and ignores the coupling relationships between them. To address the omission of semantic relationships between keywords in VSM, based on the semantic matrix S_K constructed in Step 2, each keyword query vector is transformed into a new feature vector $\overrightarrow{Q'} = \overrightarrow{Q'} S_K$, which enriches the query vector representation with the coupling relationships between keywords. Then, using this transformation the corresponding kernel [4] of two query vector $\overrightarrow{Q}_{i1}$ and $\overrightarrow{Q}_{i2}$ can be written as,

$$k'(Q_{i1}, Q_{i2}) = \overrightarrow{Q}_{i1}(S_K^T * S_K)\overrightarrow{Q}_{i2}^T \tag{8}$$

Based on the query vector representations ($\overrightarrow{Q}_{i1}, \overrightarrow{Q}_{i2}$) and kernel $k'(Q_{i1}, Q_{i2})$, we can define the kernel-based cosine similarity between two queries as follows,

$$cos_{ker}(\overrightarrow{Q}_{i1}, \overrightarrow{Q}_{i2}) = \frac{k'(Q_{i1}, Q_{i2})}{\sqrt{k'(Q_{i1}, Q_{i1})}\sqrt{k'(Q_{i2}, Q_{i2})}} \tag{9}$$

Using Equation (9), the semantic similarity between each pair of queries in query history can be obtained. The matrixes of similarities between different pairs of queries obtained by using traditional VSM-based cosine similarity (short for V-COS) and kernel-based cosine similarity (short for K-COS) algorithms are showed in Table 5 and 6, respectively.

From Table 5 and 6, we can find that the similarities of a specified query to other queries calculated by V-COS algorithm are usually same. For example, the similarities between Q_{24} to Q_{13} and Q_{35} are all 0.33 and the same to Q_{35} to Q_{24} and Q_{43}, and thus lead the queries are difficult to differ from each other. While, the similarities of them calculated by K-COS algorithm are different and more close to reality.

Table 5: Query similarity matrix based on V-COS

	Q_{13}	Q_{24}	Q_{35}	Q_{43}
Q_{13}	1.00	0.33	0.00	0.67
Q_{24}	0.33	1.00	0.33	0.00
Q_{35}	0.00	0.33	1.00	0.33
Q_{43}	0.67	0.00	0.33	1.00

Table 6: Query similarity matrix based on K-COS

	Q_{13}	Q_{24}	Q_{35}	Q_{43}
Q_{13}	1.00	0.61	0.44	0.86
Q_{24}	0.61	1.00	0.72	0.44
Q_{35}	0.44	0.72	1.00	0.61
Q_{43}	0.86	0.44	0.61	1.00

6. TOP-K KEYWORD QUERY SELECTION

6.1 Top-K Keyword Query Selection Problem

Let Q be a given keyword query over database D and W be the query history. Based on the semantic similarities between different queries, the goal is to address the top-k query selection problem defined as,

$$\Gamma_k = argmax_{\Gamma'} \sum_{i=1}^{k(k<<n)} \delta_{sim}(Q, Q_i) \tag{10}$$

where, Γ_k is a list of k keyword queries and n is the number of all queries in query history. The objective of the problem is to find a set of number k queries in query history that semantically related closely as possible to the given query.

6.2 Approach

A straightforward algorithm to find the top-k related queries is used to compare the similarities of the given query to all queries and then rank them according to the similarities. The time complexity is $O(n^2 logn)$, where n is the number of all queries in query history. However, this complexity is unacceptable for a large scale query history. Thus, we have to find an approximate method to expeditiously find the top-k related queries. This paper proposes a three-step solution to resolve it. The first step is to find a few representative queries in query history and the second step is to order the remaining queries corresponds to each representative query. The third step is to select the top-k related queries based on these orders. The first and second steps are processed during offline time and the third step is processed during online time.

Step 1. Find representative keyword queries. Based on the semantic similarities between different pairs of queries, we provide an algorithm (Algorithm 3), which is inspired by the furthest-first traversal algorithm [2], to find the representative queries in query history. Let m be the number of queries in query history W. Also let l be the number of representatives and W_l the set of representatives. The algorithm starts by picking an arbitrary query in W as a representative query, denoted by $\bar{Q}_i$, and then adds it to the set $W_l = \{\bar{Q}_i\}$, while $\bar{Q}_i$ is removed from W. Then, it picks query $\bar{Q}_j$, which is furthest from $\bar{Q}_i$ that means the $\delta_{sim}(\bar{Q}_j, \bar{Q}_i)$ is the smallest among the $\{\delta_{sim}(Q_j, \bar{Q}_i)|j \in (1, ..., m)\ and\ j \neq i\}$ in W.

Algorithm 3: Representative queries finding algorithm

Input: queries in $W = \{Q_1, ..., Q_m\}$, number l.
Output: the set of l representatives $W_l = \{Q_1, , Q_l\}$.

$W_l \leftarrow \phi$.
pick arbitrary $Q_1 \in W$.
$W_l \leftarrow W_l \bigcup \{Q_1\}$.
$W \leftarrow W - \{Q_1\}$
for $i = 2$ *to* l **do**
  $Q_i = argmin_{Q' \in W} \delta_{sim}(Q', \bar{Q}_{i-1})$
  $W_l \leftarrow W_l \bigcup \{\bar{Q}_i\}$
  $W \leftarrow W - \{\bar{Q}_i\}$
Return $W_l = \{\bar{Q}_1, .., \bar{Q}_l\}$

Step 2. Create orders for representative queries. For each representative query $\bar{Q}_i$ create an order τ_i of all remaining queries (except $\bar{Q}_i$) in query history in descending order, according to their similarities to $\bar{Q}_i$. The output of this procedure is a set of l orders. According to the output orders, each query Q_j has a score that is associated with the position of Q_j in each order τ_i. The score of Q_j in τ_i that corresponds to $\bar{Q}_i$ is:

$$s(Q_j|\bar{Q}_i) = n - \tau_i(Q_j) + 1 \tag{11}$$

where, $\tau_i(Q_j)$ represents the position of Q_j in τ_i.

Using Algorithm 3, a few representative queries can be selected. When a given query is coming, it needs to only compute the similarities between the given query and representative queries, which is used as a weighting parameter for top-k query selection.

Step 3. Select top-k related queries. For a given query Q, using the output of Step 2, this step computes the set $Q_k(W) \subseteq W$ with $|Q_k(W)| = k$, such that $\forall Q_j \in Q_k(W)$ and $Q'_j \in \{W - Q_k(W)\}$ it holds that $score(Q_j, Q) > score(Q'_j, Q)$, with $score(Q_j, Q) = \sum_{i=1}^{l} \delta_{sim}(Q, \bar{Q}_i) s(Q_j|\bar{Q}_i)$.

We next describe a method, which employs the Threshold Algorithm (TA) [13], to provide the top-k related queries for a given query. The TA uses sorted and random modes to access the queries in the orders. The Sorted mode obtains the score of a query in an order by scanning the order of the queries from the top to down sequentially. The Random mode finds the score of a query in an order in one access. The top-k query selection algorithm is shown in Algorithm 4, where the score of Q_j found in each order τ_i to Q is computed by:

$$s(Q_j, Q) = \delta_{sim}(Q, \bar{Q}_i) s(Q_j|\bar{Q}_i) \tag{12}$$

where, $s(Q_j, Q)$ is weighted by the semantic similarity between the given query Q to the representative query $\bar{Q}_i$.

The score of Q_j in every other order is found via random access, and all these scores are summed, resulting in the final score of Q_j for the given query Q:

$$score(Q_j, Q) = \sum_{i=1}^{l} \delta_{sim}(Q, \bar{Q}_i) s(Q_j|\bar{Q}_i) \tag{13}$$

In Algorithm 4, λ is a threshold for the current *repeat* loop, for any query Q'_j that has not yet been seen in the *repeat* loop access, its score is less than λ. The time complexity of Algorithm 4 is $O(kl^2)$, where k is the number of queries need to be retrieved and l ($l << n$) is the number of representative queries. When l is small, Algorithm 4 achieves a significant decrease in complexity (In Section 7, we demonstrate that even for a small number of l, our top-k related query selection algorithm can achieve a relatively high accuracy).

Algorithm 4: The top-k query selection algorithm

Input: Orders set $W_l = \{\tau_1, ..., \tau_l\}$, given query Q, number k.
Output: Top-k related queries in query history.

Let $B = \{\}$ be a buffer that can hold k keyword queries.
Let L be an l size array that is used to store the score of the last visited query of each order by the end of the current round-robin cycle.
repeat
  for $i = 1$ *to* l **do**
    Retrieve next query Q_j from τ_i using sorted access.
    Compute $s(Q_j, Q) = \delta_{sim}(Q, \bar{Q}_i) s(Q_j|\bar{Q}_i)$ as Q'_j s score.
    Update $L[i]$ with score of Q_j in τ_i.
    Get score of Q_j from other orders $\{\tau_k | \tau_k \in W_l$ and $k \neq i\}$ via random access.
    $score(Q_j, Q) \leftarrow$ summing up of all the scores of Q_j retrieved from all the orders.
    Insert $\langle Q_j, score(Q_j, Q)\rangle$ into B in descending order.
    $\lambda \leftarrow \lambda + L[i]$.
until $B[k-1].score \geq \lambda$
Return B.

7. EXPERIMENTS

7.1 Experimental settings

The experiments are conducted on a computer running Windows 2008 with Intel P4 3.2-GHz CPU, and 8 GB of RAM. We implemented all algorithms in C# and SQL. We use the following two real datasets to evaluate the performance of our methods.

1. DBLP dataset. The download DBLP XML file is decomposed into 4 relational tables, that are *Authors*, *Papers*, *Write* and *Publications*, respectively. We built a keyword query system based on DBLP dataset and provided a web interface for users to submit keyword queries that they would execute. In this way, we collected 1,600 queries in an extensive scope and 500 queries are finally retained after pruned as the query history. Each remained query contained $3 \sim 5$ keywords and there are 1,674 distinct keywords in total. The keywords in query history are related to author, paper title, and conference name, and they are also inter-related within the same queries and/or across the different queries. So the query history over DBLP is very appropriate for testing the performance of our keyword coupling relationship and query semantic similarity measuring methods.

2. IMDB dataset. The Internet Movie Database (IMDB) contains movies, directors, actors and other movie-related information. For this database, we adopt the following strategy to simulate the query history. We first created a data view, in which each record is formed by joining all connected tuples according to the primary-foreign-key references. Then, we random selected 1,000 records from the data view

and extracted keywords respectively from the movie name, actor name, genre, role, and director name of each record. Next, the keywords extracted from each record were random selected to integrate as a keyword query. Finally, we totally formed 1,000 queries as the query history.

7.2 Accuracy of Keyword Coupling Relationships

This experiment aims to show how to determine the parameter α in Equation(7) to get the best accuracy of coupling relationships between keywords. To do this, we randomly selected 10 keywords from DBLP and IMDB query histories, respectively. For each keyword k_i, we first obtained the top-5 relevant keywords by using our keyword coupling relationship measuring method with respect to each value of parameter α in Equation (7) from 0 to 1 at the increments 0.1. Then, we mixed these keywords and thus a set K_i of 55 keywords was generated. Next, we calculated the frequency of occurrence for each distinct keyword in K_i and finally marked the top 5 most frequently occurring keywords as the relevant keywords since the more frequently the keyword occurring in K_i indicates the more the keyword relevant to the given keyword.

Based on the relevant keywords marked for each selected keyword k_i, we then use the *Recall* and *Precision* metrics to evaluate the accuracy of coupling relationships with respect to different values of α. Recall is the ratio of the number of relevant keywords retrieved by the algorithm to the total number of the keywords that were marked as relevant. Precision is the ratio of the number of relevant keywords retrieved by the algorithm to the total number of keywords that were retrieved. In our case, both the relevant and retrieved keywords number 5, making the Recall and Precision equal. Figure 3 shows the Recall&Precision of answers obtained by using our keyword coupling relationship measuring method with respect to different values of α on DBLP and IMDB datasets, respectively. Note that, the Recall&Precision for each value of α is averaged over 10 selected keywords.

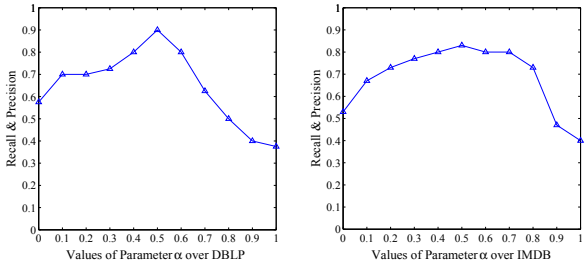

Figure 3: Accuracy of answers for different values of α on DBLP and IMDB datasets

It can be seen that the curve of Recall&Precession reaches the peak at $\alpha = 0.5$ for both DBLP and IMDB datasets, which demonstrates that our method achieves the best performance on these two datasets when α is set to 0.5, and the corresponding accuracy are 0.90 and 0.83, respectively. Note that, since the keywords from different datasets may have different coupling relationships, it is necessary to optimize the setting of α to achieve the highest accuracy. It also can be seen that, the peak accuracy on IMDB is lower than that on DBLP. This is because the query history of DBLP is the real queries user submitted, in which the coupling relationships between keywords of them are very strong, so that we can capture the coupling relationships more efficiently.

7.3 Accuracy of Keyword Query Semantic Similarities

This experiment aims to evaluate how well our query semantic similarity measuring algorithm captures the user query intentions. To verify the accuracy of the semantic similarities between different queries, we adopt the strategy as follows. We invited 10 people, which are researchers and PhD students, to choose the queries from the DBLP and IMDB query histories. For each selected query Q_i, we generated a set K_i of 30 queries from query history that likely to contain a mix of relevant and irrelevant queries in relation to the given query. Each set K_i is formed by mixing the top 10 results of each algorithm of kernel-based similarity (K-COS), traditional VSM-based cosine similarity (V-COS), and RANDOM (it selects the queries in a random order and provides a baseline to show the efficiency of the other two algorithms). Lastly, we presented the queries with their corresponding $K_i's$ to each user in our study. Each user had to mark the top 10 queries in K_i that they considered semantically related to Q_i. We then measured how closely the 10 queries marked as relevant by the user matched the 10 queries returned by each algorithm. The users were asked to describe whether they considered a query Q' related to a given query Q based on:

(i) the keywords in Q' are related to that of Q. For example, the keyword "sequential mining" is related to "association rules", hence the queries contains the keywords are considered to be related to each other.

(ii) the results of Q' are relevant to that of Q, although no keyword in Q' is same or related to Q. For example, the results of query "e-commerce, decision making" are partly overlapped with those of query "data analysis, decision tree", but the keywords in queries are not explicit related.

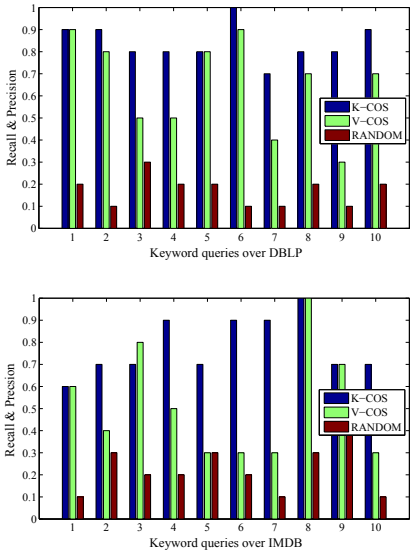

Figure 4: Accuracy for K-COS, V-COS, and RANDOM

The Recall&Precision metrics is also used to evaluate this overlap. Figure 4 shows the Recall&Precision of answers for K-COS, V-COS, and RANDOM. We can see that the Recall&Precision of K-COS is much higher than V-COS over the two datasets. The averaged Recall&Precision of K-COS is 0.84 for DBLP and 0.78 for IMDB while the V-COS is 0.65 for DBLP and 0.52 for IMDB. This is because V-COS learns the query similarity based on traditional VSM, which only considers the local overlapped information between the queries. In contrast, the K-COS considers both the local overlapped information between the queries and the coupling relationships of the keywords within/across queries. Additionally, the reasonability of keyword coupling relationship, which considers both the intra- and inter-coupling between keywords, is demonstrated by Experiment 7.2. Hence, the answers for the given query can meet the user's intentions more closely.

7.4 Accuracy of Top-k Query Selection

This experiment aims to test the accuracy of the top-k queries obtained using only the orders that corresponds to representative queries when compared with the top-k keyword queries obtained by computing the similarities of the given query to all queries in query history. To quantify this accuracy, we use $R(All, k)$ to denote the top-k queries returned by computing the similarities of the given query to all queries in query history, and $R(Rep, k)$ to denote the top-k queries returned using only the orders corresponding to representative queries. The overlap of two top-k answer sets is measured using the Jaccard coefficient:

$$J(R(Rep,k), R(All,k)) = \frac{|R(Rep,k) \bigcap R(All,k)|}{|R(Rep,k) \bigcup R(All,k)|} \quad (14)$$

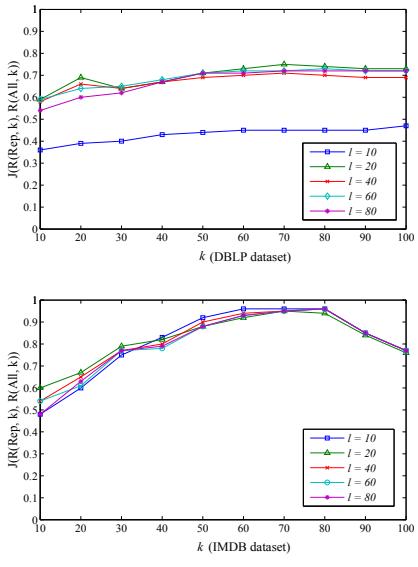

Figure 5: Accuracy of different l when value k varied

The coefficient falls into the interval of [0, 1] and the higher its value the more similar the two sets of queries are. In this experiment, we use three parameters: n, l, and k, to character the dataset. Here, n is the number of queries in each of the orders, l the number of representative queries, and k the number of queries needs to be selected. Figure 5 shows the value of the coefficients (averaged over 10 test queries) for different values of k, when l={10, 20, 40, 60, 80}. The values of n are fixed to 500 for DBLP and 1000 for IMDB (because there are 500 and 1000 queries in their query histories, respectively), and k is varied in {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}.

From Figure 5 we can see the coefficients corresponding to different numbers of l are nearly identical (except the case of l=10 for DBLP) and the accuracy is relatively high, which means when only a small number of representative queries (such as $l = 20$) are used to find the related queries, the information lost by looking at the orders for representatives instead of computing similarities of the given query to all queries in query history is not substantial. Additionally, the accuracy of l=10 is much lower than that of other numbers of l on DBLP dataset, which indicates that the number of l should be selected appropriately for different datasets.

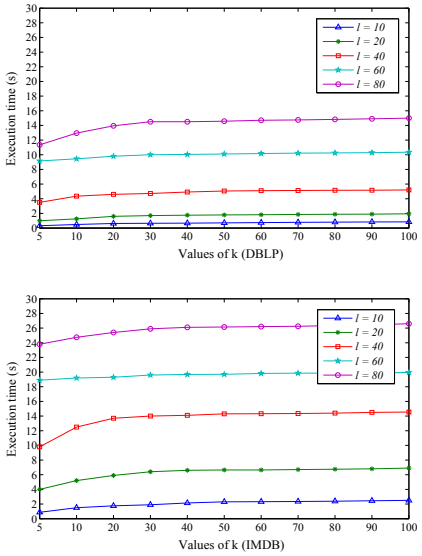

Figure 6: Performance of top-k selection algorithm

7.5 Experimental Results

This experiment aims to verify the performance of the top-k selection algorithm. In this experiment, we generate 5,000 and 10,000 keyword queries as query history for DBLP and IMDB datasets, respectively. Based on these datasets, we fix the number of l to 10, 20, 40, 60, and 80, respectively and then test the execution time of top-k selection algorithm for different k values (here, we set the number of k to 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100, repectively). Figure 6 shows the execution time on DBLP and IMDB datasets for different k values when l={10, 20, 40, 60, 80}.

From Figure 6, we can see that the algorithm runs fast, especially when $l \leq 20$ and $k \leq 10$. As demonstrated in Experiment 7.4, the information lost is not substantial by only using $l = 20$ representative queries to find related queries. Meanwhile, providing top-10 related queries are enough for most users in real applications. Therefore, our top-k selection algorithm can be very well suitable for processing the large scale of query history. Additionally, the performance

of the algorithm decreases with the increasing of value l and k. The reason is that the top-k query selection algorithm needs to deal with more queries in orders as the number l and k increased. We also computed the time consumption for computing the similarities of a given query to all queries in query history. It takes approximately 48 seconds for DBLP and 321 seconds for IMDB to obtain the similarities of a given query to all remaining queries in query history. Our top-k selection algorithm clearly outperforms existing methods and demonstrates more efficient performance.

8. CONCLUSIONS

This paper presented a novel approach to address the coupling relationships hidden between keywords and between keywords and queries to enhance semantic approximate keyword queries over Web databases. The techniques proposed in this paper can also be adopted in domains other than web databases. For example, scientific database users, especially the ones who are not familiar with the area, would benefit from a system that improves their original queries. Furthermore, this approach can be used both at the application level and be incorporated into most of existing keyword search frameworks to support the semantic approximate keyword search. The experiments on real dataset identified that the keyword coupling relationship and query semantic similarity measuring methods can capture the semantic relationships of keywords and queries more reasonable. The top-k related queries can be returned quickly and relatively high accuracy is achieved, even though only a small number of representative queries are retained.

It would be interesting to investigate (i) how to minimize the updating cost when the query history is varied, and (ii) the effect of introducing some diversity in the suggested queries.

9. ACKNOWLEDGMENTS

This work is supported by the National Science Foundation for Young Scientists of China (No.61003162) and the Young Scholars Growth Plan of Liaoning (No. LJQ2013038).

10. REFERENCES

[1] B. Aditya, G. Bhalotia, and S. Chakrabarti. Banks: browsing and keyword searching in relational databases. In *VLDB*, pages 1083–1086, 2002.

[2] R. Agrawal, R. Rantzau, and E. Terzi. Context-sensitive ranking. In *SIGMOD*, pages 383–394, 2006.

[3] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system forkeyword-based search over relational databases. In *VLDB*, pages 5–16, 2002.

[4] L. AlSumait and C. Domeniconi. Text clustering with local semantic kernels. In *Survey of Text Mining II*, pages 87–105, 2008.

[5] Z. F. Bao, J. H. Lu, and T. W. Ling. Xreal: an interactive xml keyword searching. In *CIKM*, pages 1933–1934, 2010.

[6] S. Bergamaschi, E. Domnori, and F. Guerra. Keyword search over relational databases: a metadata approach. In *SIGMOD*, pages 565–576, 2011.

[7] D. Bollegala, Y.Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *WWW*, pages 757–786, 2007.

[8] L. B. Cao, Y. M. Ou, and P. S. Yu. Coupled behavior analysis with applications. *ACM Trans. Knowl. Data Eng.*, 24(8):1378–1392, 2012.

[9] Z. Y. Chen and T. Li. Addressing diverse user preferences in sql-query-result navigation. In *SIGMOD*, pages 641–652, 2007.

[10] X. Cheng, D. Q. Miao, C. Wang, and L. B. Cao. Coupled term-term relation analysis for document clustering. In *IJCNN*, pages 1–8, 2013.

[11] G. Das, D. Gunopulos, and N. Koudas. Answering top-k queries using views. In *VLDB*, pages 451–462, 2006.

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

[13] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, pages 102–113, 2001.

[14] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient ir-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.

[15] V. Hristidis and Y. Papakonstantinou. Discover: keyword search in relational databases. In *VLDB*, pages 670–681, 2002.

[16] A. Huang, D. Milne, and E. Frank. Clustering documents using a wikipedia-based concept representation. In *Advances in Knowledge Discovery and Data Mining*, pages 628–636, 2009.

[17] L. B. Kong, R. Gilleron, and A. Lemay. Retrieving meaningful relaxed tightest fragments for xml keyword search. In *EDBT*, pages 815–826, 2009.

[18] Y. Luo, X. M. Lin, and W. Wang. Spark: top-k keyword query in relational databases. In *SIGMOD*, pages 305–316, 2007.

[19] N. Sarkas, N. Bansal, and G. Das. Measure-driven keyword query expansion. *PVLDB*, 2(1):121–132, 2009.

[20] S. Tata and G. M. Lohman. Sqak: doing more with keywords. In *VLDB*, pages 889–902, 2008.

[21] C. Wang, L. B. Cao, and M. C. Wang. Coupled nominal similarity in unsupervised learning. In *CIKM*, pages 973–978, 2011.

[22] C. Wang, Z. She, and L. B. Cao. Coupled clustering ensemble: incorporating coupling relationships both between base clusterings and objects. In *ICDE*, pages 374–385, 2013.

[23] X. Wang and G. Sukthankar. Multi-label relational neighbor classification using social context features. In *KDD*, pages 464–472, 2013.

[24] J. J. Yao, B. Cui, and L. S. Hua. Keyword query reformulation on structured data. In *ICDE*, pages 953–964, 2012.

[25] A. Yu, P. K. Agarwal, and J. Yang. Top-k prefrences in high dimensions. In *ICDE*, pages 748–759, 2014.

[26] R. Zhou, C. F. Liu, and J. X. Li. Fast elca computation for keyword queries on xml data. In *EDBT*, pages 549–560, 2010.