Hindawi
Scientific Programming
Volume 2017, Article ID 7613026, 12 pages
https://doi.org/10.1155/2017/7613026

Research Article

Keyword Query Expansion Paradigm Based on Recommendation and Interpretation in Relational Databases

Yingqi Wang, Nianbin Wang, and Lianke Zhou

College of Computer Science and Technology, Harbin Engineering University, Harbin, China

Correspondence should be addressed to Nianbin Wang; wangnianbin@hrbeu.edu.cn

Received 15 January 2017; Revised 17 April 2017; Accepted 3 May 2017; Published 29 May 2017

Academic Editor: Michele Risi

Due to the ambiguity and impreciseness of keyword query in relational databases, the research on keyword query expansion has attracted wide attention. Existing query expansion methods expose users' query intention to a certain extent, but most of them cannot balance the precision and recall. To address this problem, a novel two-step query expansion approach is proposed based on query recommendation and query interpretation. First, a probabilistic recommendation algorithm is put forward by constructing a term similarity matrix and Viterbi model. Second, by using the translation algorithm of triples and construction algorithm of query subgraphs, query keywords are translated to query subgraphs with structural and semantic information. Finally, experimental results on a real-world dataset demonstrate the effectiveness and rationality of the proposed method.

1. Introduction

In recent years, keyword query in relational databases has been widely applied due to its simplicity and ease of use [1, 2]. Although this method does not require users to be knowledgeable about the underlying structure of databases and structured query language (e.g., SQL), its semantic fuzziness and its expressive power are limited due to the lack of structure [3]. In addition, ordinary users are usually unable to specify exact keywords to describe their query intention, which makes it harder to return adequate results through keyword query [4, 5]. For these reasons, the precise and recall of keyword query methods cannot be effectively guaranteed [6]. Therefore, query expansion has even been an important research branch, which can completely and precisely interpret the query and then improve the recall and precision of query results [7–10].

Problem and Motivation. Query expansion is to provide more descriptions for the information requirements to improve the query performance. First, we formally define the problem of query expansion as follows.

Definition 1 (query expansion). Given an input keyword query $Q = \{k_1, k_2, \ldots, k_m\}$ over a relational database, query expansion is to find k related queries Q_i $(i = 1, \ldots, k)$ with k largest score(Q, Q_i), where score(Q, Q_i) evaluates the relevance between Q and Q_i.

Then, we will illustrate the research motivation of this paper by the following examples and analysis. Although numerous query expansion methods can be found in the scientific literature, unfortunately these studies suffer from two main limitations. First, existing approaches do not consider the relationship between keywords, which causes the low query precision. Second, most of the existing work neglects the similar words or related items of query keywords, leading to the low query recall. On the one hand, Meng et al. propose a semantic approximate keyword query method based on keyword and query coupling relationships [11]. This method partly solves the problems of semantic fuzziness and limitations of expression, but it analyzes the keyword and query coupling relationships through query history. When the query history of database is incomplete and even missing, the method will not be able to conduct semantic analysis normally. Additionally, the queries obtained by this method contain only content information related to the keywords rather than the structure information between keywords, thus affecting the query precision. For example, suppose a user issues the keyword query "Machine learning Arthur

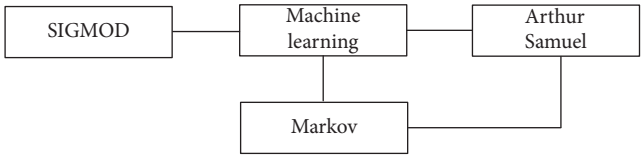

Figure 1: An expanded query.

Samuel" in DBLP to retrieve the paper "Machine learning" published by Arthur Samuel. The query expansion contains "Machine learning Arthur Samuel SIGMOD" using the above query expansion method. The expansion extends the query with the content but lacks structure information. Consequently, all tuples containing "Machine learning," "Arthur Samuel," or "SIGMOD" will be returned. Obviously, such results are not precise enough and many of them may not be useful. The method proposed in this paper extends the initial query with content and structure related to the keywords. An expanded query is as shown in Figure 1.

It can express the potential semantic and structural information of original keyword query: find out the paper "Machine learning" published in SIGMOD and authored by Arthur Samuel; simultaneously find out the paper "Markov" which cites paper "Machine learning" and authored by Arthur Samuel. In contrast to the query expansion in [11], the method proposed in this paper extends the original query to query subgraphs with underlying structure of databases and thus further improves the query precision. On the other hand, Ganti et al. [12] translates keyword queries into SQL based on the materialized mappings. Bergamaschi et al. [3] propose a Metadata method to translate keyword queries into SQL based on Munkres algorithm. Though these methods describe users' query intention to a degree, they do not take into account similar words and related items extension. Thus these methods have relative high query precision, but their recall needs to be further improved. For example, assume that a user hopes to study the methods of data analysis. Because of his knowledge limitation, the user accesses the database DBLP and submits a keyword query "Machine learning Arthur Samuel," and the expanded query obtained via the above methods is as follows: select R3.Title from Author R1, PaperAuthor R2, Paper R3 where R2.Pid=R3.Pid and R2.Aid=R1.Aid and match(R3.Title) against ("Machine learning" in Boolean mode) and match(R1.Name) against ("Arthur Samuel" in Boolean mode), while the user may also be intending to retrieve papers which have similar or relevant topics to "Machine learning" such as paper "data mining" published by Micheline Kamber. Since paper "data mining" cites paper "Machine learning" and they are much related to each other, the user is also interested in it. The results of the query expansion method proposed in this paper are as shown in Figure 2. Compared with the previous result, these results not only have structural relationship between keywords but also contain the related or similar queries with query keywords.

The analysis and examples of the query expansion methods in Section 1 illustrate that the key challenge here is to develop an approach which balances both query precision and recall. In this paper we focus on the problem of how to tackle the above two limitations and then improve the performance of keyword query expansion approach. We propose a novel query expansion method ReInterpretQE based on query recommendation and interpretation, which extends a keyword query to a list of query subgraphs. These subgraphs could better capture users' information need and the possible semantics of the keyword query and then guide users to explore related tuples in the relational database. First, we construct a term similarity matrix based on the tuple information contained in relational databases. Second, we make the query recommendation using the similarity matrix and dynamic programming to get a query list. This query list consists of top-k queries related to the initial query. Finally, we transform the keyword query into query subgraphs. Through the query recommendation and interpretation, the query expansion method improves the recall and precision of query results.

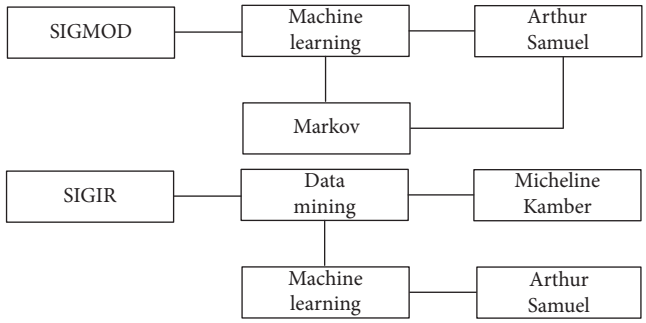

Figure 2: An expanded query of ReInterpretQE.

The main contributions of this paper are summarized as follows:

(1) We present a keyword query expansion paradigm ReInterpretQE in relational databases, which is based on query recommendation and interpretation.

(2) We design a probabilistic recommendation algorithm based on similarity matrix and dynamic programming.

(3) We propose a keyword query interpretation method. It uses statistical information and schema graph of database for translating a keyword query to query subgraphs.

(4) We conduct extensive experiments on the DBLP dataset, and experimental results demonstrate the effectiveness and feasibility of the proposed method.

The paper is organized as follows: Section 1 introduces the research motivation, main contribution, and structure of this paper; Section 2 reviews the related work; Section 3 describes the architecture of approach ReInterpretQE and then provides details of algorithms in query recommendation and query interpretation; Section 4 conducts the experiments on a real dataset and compares the experimental results; Section 5 concludes this paper and prospects the study in future.

2. Related Work

This section discusses the related research work. It mainly includes the following two parts: keyword query in relational databases and keyword query expansion.

2.1. Keyword Query in Relational Databases. Recently, keyword query methods in relational databases have been studied extensively [2, 13–15]. According to the different modeling methods of databases, the existing query methods can be divided into two main categories: schema graph-based methods and data graph-based methods. In [16–21], the database is modeled as a schema graph, where nodes represent tables and edges represent primary-foreign-key relationships. These methods enumerate all possible CNs (Candidate Networks) based on the schema graph. Although these methods store the abstract structural information and take little memory, the generation process of CNs needs to take a huge amount of time. Correspondingly, in [22–27] the database is modeled as a data graph, where nodes represent tuples and edges represent primary-foreign-key relationships between tuples. The methods identify the minimum connection trees that contain the keywords on the data graph. A major challenge of the methods is the maintenance of data graph. The whole data graph needs to be reconstructed when the data in database changes, which is often a time-consuming exercise. Therefore, it is important to study the dynamic update of databases and design incremental query methods based on the dynamic construction of data graph.

2.2. Keyword Query Expansion Paradigm. In the field of keyword query in relational databases, the majority of existing studies have focused on how to improve the efficiency of query algorithms, while effective query preprocessing has yet to be investigated. Since keyword queries lack the structural information and users tend to select inappropriate keywords, semantic fuzziness becomes an urgent problem to be solved. The method in [28] expands the original query by using query log mining techniques, but it is not applicable in the relational databases. Reference [7] selects the related query expression using user feedback. The results obtained by this query expression are more accordant with users' query intention. However, it needs the interaction of humans, so its efficiency is low. In [3], the keyword query is transformed to a SQL statement based on Munkres, which provides possible semantic descriptions. This method is useful for identifying users' query intention. Nevertheless, the approach does not consider the multiple connection between keywords (i.e., there are various explanations for a keyword query). In addition, the above approach does not take into account the similar words and related items which can well express the intention of users. An analysis model about coupling relationship is presented in [11]. It extracts semantic relations based on query history. However, this model also has great limitations. When the query logs are missing or user's preference often changes, the model cannot be applied effectively. Therefore, this paper proposes a query expansion method ReInterpretQE, which consists of two steps: query recommendation and query interpretation. First, we construct similarity matrix based on the structure and content information in databases and put forward a probabilistic recommendation algorithm using dynamic programming. Second, we come up with a keyword query interpretation method to transform the keywords to subgraphs based on the statistics and schema graph of database. Experimental results show that both the recall and precision of query results have been improved by using the proposed query expansion method.

3. Query Expansion Approach

3.1. Overview of ReInterpretQE. To solve the two problems of semantic fuzziness and limitations of expression, this paper comes up with a novel two-step query expansion method, ReInterpretQE. This method is based on query recommendation and query interpretation. The goal of query recommendation is to extend the initial query Q to a list of keyword queries related to it, so that the query results are more comprehensive and better to meet the demands of users. The query interpretation is to translate the list of keyword queries into query subgraphs, which can lock the query results more precisely. It is designed to improve the recall and precision of query results. Figure 3 shows the architecture diagram of ReInterpretQE, which is divided into two main phases: query recommendation and query interpretation.

Phase 1 (query recommendation). In the process of query recommendation, the intrasimilarity and intersimilarity between terms are calculated using the structure information, content information, and words cooccurrence, and a similarity matrix is constructed based on the above two similarities. Then the idea of dynamic programming is used to build Viterbi model; thus a keyword query $Q = \{k_1, k_2, \ldots, k_m\}$ can be extended to a keyword query list $Q^1 = (q_1^1, q_2^1, \ldots, q_m^1) \cdots Q^k = (q_1^k, q_2^k, \ldots, q_m^k)$. The query list produced by the query recommendation process is semantically related to the original query.

Phase 2 (query interpretation). In the process of query interpretation, an algorithm is put forward for translation from keywords to triples. Then query subgraphs are built for each query in query list using the schema graph of database. The implementation detail of the algorithm will be introduced in the subsequent sections.

First, Section 3.2 describes how to recommend query list with the same or close similarity of meaning according to structure information and content information in databases. This problem can be solved effectively by the construction of term similarity matrix and probabilistic recommendation algorithm. The method makes the query results contain more information that users want to obtain. Thus the query recall can be further improved. Section 3.3 puts forward a two-step query interpretation method. The keyword query can be translated into query subgraphs with potential structural information through the following steps: Step 1: translation from keywords to triples; Step 2: construction of query subgraphs. With the above process of

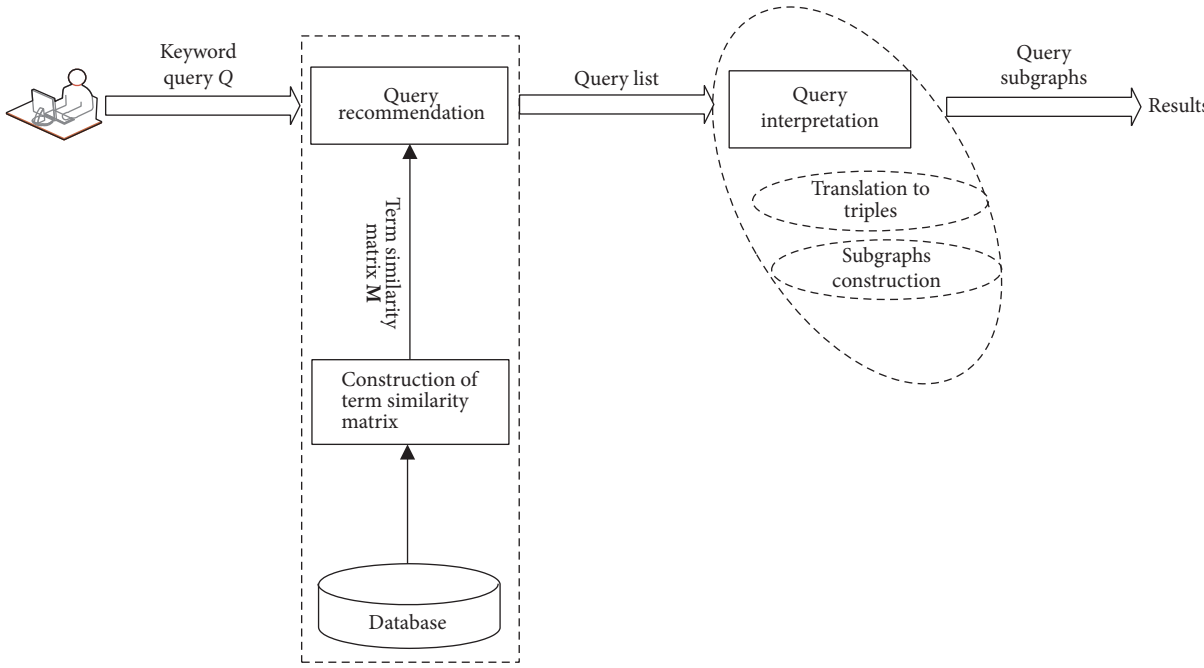

Figure 3: Architecture of ReInterpretQE.

query interpretation, the query precision is improved effectively.

3.2. Query Recommendation. It is difficult to specify proper keywords to express query intention for a common user. Therefore, a lot of information related to query cannot be returned as results. This section presents a query recommendation method to extend the original query and derives a series of queries which have a similar semantic to the original query. Thus the recall of the query results can be improved. Assume that a user submits a query $Q = \{k_1, k_2, \ldots, k_m\}$. First, Section 3.2.1 constructs the term similarity matrix to find the top-n keywords $k_i^1, k_i^2, \ldots, k_i^n$ related to any keyword k_i in original query Q. Second, Section 3.2.2 proposes a probabilistic recommendation algorithm. It uses dynamic programming to build Viterbi model for translating the original query keyword $Q = \{k_1, k_2, \ldots, k_m\}$ to a query list $Q^1 = (q_1^1, q_2^1, \ldots, q_m^1) \cdots Q^k = (q_1^k, q_2^k, \ldots, q_m^k)$.

3.2.1. Construction of Term Similarity Matrix. In the construction phase of term similarity matrix, from two aspects of intrasimilarity and intersimilarity, this paper calculates the similarity between keywords based on the structure information and content information.

(a) Intrasimilarity. Normally, in information retrieval if two keywords often appear in the same documents, thus keywords are regarded as semantically related. In relational databases, tuples are generally taken as virtual documents. Similarly, the higher the cooccurrence of two keywords in the same tuples, the higher the degree of similarity between two keywords. The intrasimilarity between two keywords is measured using Jaccard similarity coefficient, as shown in

$$\text{simi}_{\text{in}}\left(k_i, k_j\right) = \begin{cases} J\left(k_i, k_j\right) = \dfrac{\left|\text{TS}\left(k_i\right) \cap \text{TS}\left(k_j\right)\right|}{\left|\text{TS}\left(k_i\right) \cup \text{TS}\left(k_j\right)\right|} & i = j \\ 1 & i \neq j, \end{cases} \tag{1}$$

where TS(k_i) and TS(k_j) are tuple sets containing k_i and k_j, respectively.

We can obtain the intrasimilarity between any two keywords in databases by formula (1). Example 2 will further show the calculation process above.

Table 1: Relational database DBLP.

Pid	Title
P_1	Database, query, structured information
P_2	Query, statistical analysis, machine learning
P_3	Probability, machine learning, data mining
P_4	Structured information, database, data mining

Example 2. To facilitate explanation of the approach, we simplify the structure and content of database as shown in Table 1. There is a data table with four tuples P_1, P_2, P_3, and P_4 in DBLP database. We use da, qe, si, sa, ml, pr, and dm to represent database, query, structured information, statistical

analysis, machine learning, probability, and data mining. The intrasimilarity between keywords da and qe is as follows:

$$\begin{aligned}\text{simi}_{\text{in}}(\text{da}, \text{qe}) &= J\left(k_i, k_j\right) = \frac{\left|\{P_1, P_4\} \cap \{P_1, P_2\}\right|}{\left|\{P_1, P_4\} \cap \{P_1, P_2\}\right|} \\ &= 0.3.\end{aligned} \tag{2}$$

Similarly, we can calculate the intrasimilarity between any two keywords and get the following intrasimilarity matrix $\mathbf{M}_{\text{in}}$:

$$\mathbf{M}_{\text{in}} = \begin{bmatrix} & \text{da} & \text{qe} & \text{si} & \text{sa} & \text{ml} & \text{pr} & \text{dm} \\ \text{da} & 1 & 0.3 & 1 & 0 & 0 & 0 & 0.3 \\ \text{qe} & 0.3 & 1 & 0.3 & 0.5 & 0.3 & 0 & 0 \\ \text{si} & 1 & 0.3 & 1 & 0 & 0 & 0 & 0.3 \\ \text{sa} & 0 & 0.5 & 0 & 1 & 0.5 & 0 & 0 \\ \text{ml} & 0 & 0.3 & 0 & 0.5 & 1 & 0.5 & 0.3 \\ \text{pr} & 0 & 0 & 0 & 0 & 0.5 & 1 & 0.5 \\ \text{dm} & 0.3 & 0 & 0.3 & 0 & 0.3 & 0.5 & 1 \end{bmatrix}. \tag{3}$$

(b) Intersimilarity. The intrasimilarity reflects the direct semantic similarity according to the cooccurrence of keywords. Additionally, there is also indirect semantic similarity between keywords. For example, although da and sa do not appear in a tuple at the same time, they have the indirect similarity via keyword qe, because da and qe appear in the same tuple P_1, while sa and qe appear in the same tuple P_2. Keyword qe is called the semantic associative term; thus keywords da and sa have the indirect semantic similarity. Next, we will detail the computing process of intersimilarity.

Suppose keywords k_i and k_j belong to different tuples and the set of semantic associative terms is T. If k_m is any element in T, then the intersimilarity between k_i and k_j via associative term k_m is as shown in

$$\begin{aligned}&\text{simi}_{\text{out}}\left(k_i, k_j \mid k_m\right) \\ &\quad = \min\left\{\text{simi}_{\text{in}}\left(k_i, k_m\right), \text{simi}_{\text{in}}\left(k_j, k_m\right)\right\}.\end{aligned} \tag{4}$$

The intersimilarity between k_i and k_j is as shown in

$$\begin{aligned}&\text{simi}_{\text{out}}\left(k_i, k_j\right) \\ &\quad = \begin{cases} \dfrac{\sum_{\forall k_m \in T} \text{simi}_{\text{out}}\left(k_i, k_j \mid k_m\right)}{|T|} & i \neq j \\ 1 & i = j. \end{cases}\end{aligned} \tag{5}$$

Example 3. We take the DBLP database in Table 1, for example, as well. Example 2 leads us to know that the intrasimilarity between keywords si and ml is $\text{simi}_{\text{in}}(\text{si}, \text{ml}) = 0$. They have the intersimilarity via keywords qe and dm, where $\text{simi}_{\text{in}}(\text{si}, \text{qe}) > 0$, $\text{simi}_{\text{in}}(\text{ml}, \text{qe}) > 0$, $\text{simi}_{\text{in}}(\text{si}, \text{dm}) > 0$, and $\text{simi}_{\text{in}}(\text{ml}, \text{dm}) > 0$. Thus the intersimilarity between si and ml via qe is as follows:

$$\begin{aligned}&\text{simi}_{\text{out}}(\text{si}, \text{ml} \mid \text{qe}) \\ &\quad = \min\left\{\text{simi}_{\text{in}}(\text{si}, \text{qe}), \text{simi}_{\text{in}}(\text{ml}, \text{qe})\right\} = 0.3.\end{aligned} \tag{6}$$

The intersimilarity between si and ml via dm is as follows:

$$\begin{aligned}&\text{simi}_{\text{out}}(\text{si}, \text{ml} \mid \text{dm}) \\ &\quad = \min\left\{\text{simi}_{\text{in}}(\text{si}, \text{dm}), \text{simi}_{\text{in}}(\text{ml}, \text{dm})\right\} = 0.3.\end{aligned} \tag{7}$$

In conclusion, the intersimilarity between si and ml is as follows:

$$\begin{aligned}\text{simi}_{\text{out}}(\text{si}, \text{ml}) &= \frac{\sum_{\forall k_m \in \{\text{qe}, \text{dm}\}} \text{simi}_{\text{out}}(\text{si}, \text{ml} \mid k_m)}{\left|\{\text{qe}, \text{dm}\}\right|} \\ &= 0.3.\end{aligned} \tag{8}$$

The intersimilarity matrix of DBLP database in Table 1 is as follows:

$$\mathbf{M}_{\text{out}} = \begin{bmatrix} & \text{da} & \text{qe} & \text{si} & \text{sa} & \text{ml} & \text{pr} & \text{dm} \\ \text{da} & 1 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0 \\ \text{qe} & 0 & 1 & 0 & 0 & 0 & 0.3 & 0.3 \\ \text{si} & 0 & 0 & 1 & 0.3 & 0.3 & 0.3 & 0 \\ \text{sa} & 0.3 & 0 & 0.3 & 1 & 0 & 0.5 & 0.3 \\ \text{ml} & 0.3 & 0 & 0.3 & 0 & 1 & 0 & 0 \\ \text{pr} & 0.3 & 0.3 & 0.3 & 0.5 & 0 & 1 & 0 \\ \text{dm} & 0 & 0.3 & 0 & 0.3 & 0 & 0 & 1 \end{bmatrix}. \tag{9}$$

(c) Construction of Term Similarity Matrix. Formula (10) integrates the intrasimilarity and intersimilarity to calculate the similarity between any two keywords

$$\begin{aligned}\text{simi}\left(k_i, k_j\right) &= \alpha\, \text{simi}_{\text{in}}\left(k_i, k_j\right) \\ &\quad + (1 - \alpha)\, \text{simi}_{\text{out}}\left(k_i, k_j\right),\end{aligned} \tag{10}$$

where $\alpha \in [0, 1]$ is the balance factor to adjust the contribution of two similarities to the final results. From the result of parameter setting experiment in Section 4.2, the precision of term similarity calculation reaches the maximum when α equals 0.5. So we can get the following term similarity matrix of DBLP database in Table 1:

$$\mathbf{M} = \begin{bmatrix} & \text{da} & \text{qe} & \text{si} & \text{sa} & \text{ml} & \text{pr} & \text{dm} \\ \text{da} & 1 & 0.15 & 0.5 & 0.15 & 0.15 & 0.15 & 0.15 \\ \text{qe} & 0.15 & 1 & 0.15 & 0.25 & 0.15 & 015 & 0.15 \\ \text{si} & 0.5 & 0.15 & 1 & 0.15 & 0.15 & 0.15 & 0.15 \\ \text{sa} & 0.15 & 0.25 & 0.15 & 1 & 0.25 & 0.25 & 0 \\ \text{ml} & 0.15 & 0.15 & 0.15 & 0.25 & 1 & 0.25 & 0.15 \\ \text{pr} & 0.15 & 0.15 & 0.15 & 0.25 & 0.25 & 1 & 0.25 \\ \text{dm} & 0.15 & 0.15 & 0.15 & 0.15 & 0.15 & 0.25 & 1 \end{bmatrix}. \tag{11}$$

Input: Keyword query $Q = \{k_1, k_2, \ldots, k_m\}$, list of related keywords $k_i^1, k_i^2, \ldots, k_i^n$, $1 \le i \le m$
Out: $Q^1, Q^2, \ldots, Q^k$, where $Q^i = (q_1^i, q_2^i, \ldots, q_m^i)$
Method:
(1) Delete the redundant related keywords

$$\left\{k_1', k_2', \ldots, k_g'\right\} \leftarrow \{k_1^1, k_1^2, \ldots, k_1^n\} \cup \{k_2^1, k_2^2, \ldots, k_2^n\} \cup \cdots \cup \{k_m^1, k_m^2, \ldots, k_m^n\} \ (g \le m \cdot n)$$

(2) Build Viterbi model $\lambda = (A, B, \pi)$

$$A_{g,g} = \left\{a_{ij} \parallel 1 \le i, j \le g\right\}, \ a_{ij} = \text{simi}(k_i', k_j')$$
$$B_{g,m} = \left\{b_{ij} \parallel 1 \le i \le g, \ 1 \le j \le m\right\}, \ b_{ij} = \text{simi}(k_i', k_j)$$
$$\pi_i = P(k_i') = \text{simi}(k_i', k_1)$$

(3) Initialize

//Variable $\delta_t(i)$ is the maximum probability in all paths whose state is i at time t.
//Variable $\psi_t(i)$ the $i-1$th node of path with the maximum probability in state i at time t.
$$\delta_1(i) = \pi_i b_i(k_1), \ i = 1, 2, \ldots, g$$
$$\psi_1(i) = 0, \ i = 1, 2, \ldots, g$$

(4) Loop

For $t = 2, 3, \ldots, m$
$$\delta_t(i) = \max_{1 \le j \le g} \left[\delta_{t-1}(j) a_{ji}\right] b_i(k_t), \ i = 1, 2, \ldots, g$$
$$\psi_t(i) = \arg\max_{1 \le j \le g} \left[\delta_{t-1}(j) a_{ji}\right], \ i = 1, 2, \ldots, g$$

(5) End

Set $P\{P^1, P^2, \ldots, P^k\} = \text{top-}k_{1 \le i \le g} \delta_m(i)$
Set $q_m\{q_m^1, q_m^2, \ldots, q_m^k\} = \arg \text{top-}k_{1 \le i \le g}[\delta_m(i)]$

(6) Paths backtracking

For $t = m-1, m-2, \ldots, 1$
$q_t^1 = \psi_{t+1}(q_{t+1}^1) \cdots q_t^k = \psi_{t+1}(q_{t+1}^k)$

(7) Return top-k keyword queries

$Q^1 = (q_1^1, q_2^1, \ldots, q_m^1) \cdots Q^k = (q_1^k, q_2^k, \ldots, q_m^k)$

Algorithm 1: Probabilistic recommendation algorithm.

When a user submits a query $Q = \{k_1, k_2, \ldots, k_m\}$, we can get the keyword lists $k_i^1, k_i^2, \ldots, k_i^n$, $1 \le i \le m$, related to the original query according to the term similarity matrix **M**.

3.2.2. Probabilistic Recommendation Algorithm. This section comes up with a probabilistic recommendation algorithm. We build the Viterbi model using dynamic programming and generate the query list related to query input, as shown in Algorithm 1.

3.3. Query Interpretation. As a fuzzy query method, keyword query cannot reflect query intention accurately. This section translates the keyword query $Q = (q_1, q_2, \ldots, q_m)$ to a set of query subgraphs $QGS\{QG_1, QG_2, \ldots, QG_n\}$ by the translation algorithm of triples and construction algorithm of query subgraphs, where QG_i is the schema subgraph. Compared with the keyword query, the query subgraph not only contains the content information but also carries structural and semantic information. Thus it can more accurately reflect the users' query intention.

3.3.1. Translation from Keywords to Triples. In order to identify users' query intention, we should know exactly the role of a keyword in databases, that is, to know whether it is Metadata or content data. Thus each keyword should be extended to include the table name, attribute names, and attribute values of the table where the keyword is located.

Table 2: Table names statistics.

Keywords	Table names
paper	Paper
...	...

Table 3: Attribute names statistics.

Keywords	Attribute names
name	Paper.Name
...	...

Table 4: Attribute values statistics.

Keywords	Attribute values
database	Paper.Name.P_1
...	...

The paper proposes the triple structure $Tri(TN, AN, AV)$ to describe the above information. Before introducing the translation algorithm of triples, we first define the following three statistics tables: statistics table TN of table names, statistics table AN of attribute names, and statistics table AV of attribute values. They make statistical analysis on the table names, attribute names, and attribute values, respectively. These three statistics tables have similar structures, as shown in Tables 2, 3, and 4.

By analyzing the statistics tables, it is obvious that every keyword in query Q may correspond to several different semantic interpretations; namely, keyword k_i can be translated to a set of triples. Query Q is also translated to several sets of triples. In most relational databases, the numbers of table names and attribute names are far less than the number of attribute values, so most ambiguities occur during the translation process of attribute values. Therefore, when we conduct the translation, we first try to match the keywords to the table names and attribute names. Then we match the remaining keywords to the attribute values to get the final sets of triples. Algorithm 2 shows the details of this translation process.

Lines (1)–(9), (10)–(24), and (25)–(39) in Algorithm 2, respectively, match the keywords in query Q to statistics TN of table names, statistics AN of attribute names, and statistics AV of attribute values. Triples corresponding to each keyword in query Q can be obtained. As each keyword may correspond to more than one attribute name or attribute value, lines (40)–(46) and (47)–(52) handle these cases, respectively. It makes the different attribute names or attribute values corresponding to the same keyword translated to multiple sets of triples. All possible sets of triples are generated by the algorithm.

3.3.2. Query Subgraphs Construction. Before the construction of query subgraphs, we should combine the triples in $TriS[i] = (Tri_{i1}, Tri_{i2}, \ldots, Tri_{im})$. The paper makes the following merger rules.

We assume that $TriS[i]$ is the set of triples corresponding to query Q. $Tri_{ij} \in TriS[i]$ will be added to a new group, if Tri_{ij} satisfied the following three cases.

Case 1. The table name of Tri_{ij} is different from all the table names of $Tri_{ik} \in TriS[i]$, $k \in [1, j-1]$.

Case 2. The table name of Tri_{ij} is the same as the one of $Tri_{ik} \in TriS[i]$, $k \in [1, j-1]$, but the attribute name and attribute value of Tri_{ij} are null.

Case 3. The table name and attribute name of Tri_{ij} are the same as the ones of $Tri_{ik} \in TriS[i]$, $k \in [1, j-1]$, but the attribute of Tri_{ij} is single-valued.

These merger rules are intended to add the triples corresponding to different entities into different groups, so that the interpretation process of triples is further refined. After the translation from keywords to triples and the mergers of triples, this paper translates the merged triples $TriG[i](g_{i1}, g_{i2}, \ldots, g_{in})$ to query subgraphs, where $\forall g_{ij} \in TriG[i]$ contains all the triples belonging to the same entity. Algorithm 3 describes the construction process of query subgraphs in detail.

Lines (1)-(2) of Algorithm 3 initialize variables. Lines (3)–(7) create node v'_j for the triple group g_{ij} in triple queries $TriG[i](g_{i1}, g_{i2}, \ldots, g_{in})$ and add it to the query subgraph QG. Line (8) traverses the schema graph G to match the minimal subgraph SG corresponding to the above nodes. Lines (9)-(10) and (11)-(12) add the intermediate nodes and edges of SG to query subgraph QG, respectively. Lines (13)-(14) return the set QGS of query subgraphs.

Table 5: Description of attributes in datasets.

Datasets	Number of tuples	Number of relationships
DBLP	4.3 M	16.5 M

4. Results and Discussion

Our experiments are conducted on the real dataset DBLP [29]. The experiments mainly deal with the selection of balance factor α in the calculation process of term similarity and the performance evaluation of our query expansion method ReInterpretQE. Section 4.1 introduces the dataset, query sets, and experimental environment used in the experiment. Section 4.2 compares the precisions of the term similarity calculation with different parameters to choose the optimal value of α. Section 4.3 gives contrast experiments using Metadata [3] and K-coupling [11] as the baselines. The performance of these algorithms is evaluated, respectively. Performance metrics include precision, recall, and F-score. The experiment is used to verify the performance of the method ReInterpretQE.

4.1. Experimental Setup

4.1.1. Dataset. The paper uses the DBLP dataset released in March 2015 [29]. The address of downloading is http://dblp.uni-trier.de/. The main statistics of the dataset are as shown in Table 5. DBLP is a computer bibliography dataset widely used for query expansion in relational databases. The dataset records the information about papers published by scholars. Its original form is XML and we use the Java SAX API to parse the XML file. Then we can obtain five data tables, where tables Author, Paper, and Conference contain information about scholars, papers, and conferences, respectively. Tables Cite and Write are relationship tables. The former specifies the reference relationships and the latter contains the writing relationships between scholars and papers. Figure 4 shows a sample of five tables from the DBLP dataset. The database DBLP contains a number of tuples. They have semantic relevance in content and primary-foreign key relationships in structure. So the DBLP dataset is very appropriate for testing the performance of our query expansion method.

4.1.2. Query Sets. In the experiment, we invite researchers to choose keywords from DBLP dataset and then build the keyword query they want to perform. By this method, 6 sets of queries with length ranging from 1 to 6 are obtained to form the query sets. Each set contains 10 queries. According to the above method, the researchers constantly submit queries in an extensive scope and we collect 600 queries from researches as the query history.

4.1.3. Experimental Environment. Our experiments are performed on a computer running Windows 10 with Intel(R)

Input: $Q = (q_1, q_2, \ldots, q_m)$
Output: $TriS[1], TriS[2], \ldots, TriS[i]$; $TriS[i] = (Tri_{i1}, Tri_{i2}, \ldots, Tri_{im})$
Method:
(1) if (Q is not Null) then
(2) for i = 1 to $|Q|$ do
(3) scan Table TN;
(4) if (matched(q_i) == TRUE) then
(5) (q_i, $null$, $null$).add To(TNS);
(6) Q.remove(q_i);
(7) end if
(8) end for
(9) end if
(10) if (Q is not Null) then
(11) for j = 1 to $|Q|$ do
(12) scan Table AN;
(13) IsAN = FLASE;
(14) foreach AN_i in AN do:
(15) if (matched(q_j) == TRUE) then
(16) ($q_j.TN$, q_j, $null$).add To(ANS_j);
(17) IsAN = TRUE;
(18) end if
(19) end foreach;
(20) if (IsAN == TRUE) then
(21) Q.remove(q_j);
(22) end if
(23) end for
(24) end if
(25) if (Q is not Null) then
(26) for m = 1 to $|Q|$ do
(27) scan Table AV;
(28) IsAV = FLASE;
(29) foreach AV_i in AV do:
(30) if (matched(q_m) == TRUE) then
(31) ($q_m.TN$, $q_m.AN$, q_m).add To(AVS_j);
(32) IsAN = TRUE;
(33) end if
(34) end foreach;
(35) if (IsAN == TRUE) then
(36) Q.remove(q_m);
(37) end if
(38) end for
(39) end if
(40) $TriS[1] = (TNS_1, TNS_2, \ldots, TNS_{|TNS|}, 1, 1, \ldots, 1)$;
(41) for k = 1 to $|ANS|$ do
(42) Lth = Length($TriS$[ ]);
(43) for n = 0 to $|ANS_k| - 1$
(44) $TriS[n * \text{Lth} + 1], TriS[n * \text{Lth} + 2], \ldots, TriS[n * \text{Lth} + \text{Lth}] = (TriS[1], TriS[2], \ldots, TriS[\text{Lth}]) * (1, 1, \ldots, 1, ANS_{kn}, 1, \ldots, 1)^{\text{T}}$;
ANS_{kn} is the ($|TNS| + k$)th
(45) end for
(46) end for
(47) for k = 1 to $|AVS|$ do
(48) Lth = Length($TriS$[ ]);
(49) for n = 0 to $|AVS_k| - 1$
(50) $TriS[n * \text{Lth} + 1], TriS[n * \text{Lth} + 2], \ldots, TriS[n * \text{Lth} + \text{Lth}] = (TriS[n * \text{Lth} + 1], TriS[n * \text{Lth} + 2], \ldots, TriS[n * \text{Lth} + \text{Lth}]) * (1, 1, \ldots, 1, AVS_{kn}, 1, \ldots, 1)^{\text{T}}$;
AVS_{kn} is the ($|TNS| + |ANS| + k$)th
(51) end for
(52) end for
(53) Result: $TriS[1], TriS[2], \ldots, TriS[i]$;

Algorithm 2: Translation algorithm of triples.

Cid	Name
$c1$	ACM
$c2$	SIGIR
$c3$	VLDB
$c4$	ICDE
$c5$	SIGMOD

Pid	Formal models for expert finding in enterprise corpora	Year	Cid
$p1$	A hidden Markov model information retrieval system	1999	$c2$
$p2$	Tappan Zee (north) bridge: mining memory accesses for introspection	2013	$c1$
$p3$	Keyword proximity search on XML graphs	2003	$c4$
$p4$	Formal models for expert finding in enterprise corpora	2006	$c1$
$p5$	Schema-free XQuery	2004	$c3$

Aid	Pid
$a1$	$p1$
$a2$	$p1$
$a2$	$p2$
$a3$	$p2$
$a4$	$p3$

Pid	CitedPid
$p1$	$p4$
$p3$	$p5$

Aid	Name
$a1$	Miller
$a2$	Tim Leek
$a3$	Josh Hodosh
$a4$	A. Balmin
$a5$	Sunil

Figure 4: Example of relational database DBLP.

Input: Merged triples $TriG[i](g_{i1}, g_{i2}, \ldots, g_{in})$, Schema graph G
Output: Set of Query Subgraphs QGS
Method:
(1) $QGS \leftarrow \phi$; $BN' \leftarrow \phi$; $BN \leftarrow \phi$
(2) Let QG be a query subgraph;
(3) for $j = 1$ to $|TriG[i]|$ do
(4) Create a node v'_j for group of g_{ij};
(5) Let v'_j corresponds to v_j in G;
(6) $BN' = BN' \cup \{v'_j\}$; $BN = BN \cup \{v_j\}$
(7) Insert v'_j into QG;
(8) $SG = findSubgraph(BN, G)$;
(9) foreach intermediate node iv in SG do
(10) Create a node iv' and insert it into QG;
(11) foreach edge $e(iv_x, iv_y)$ in SG do
(12) Create an edge $e(iv'_x, iv'_y)$ in QG;
(13) Add QG into QGS;
(14) Return QGS;

Algorithm 3: Query subgraph construction algorithm.

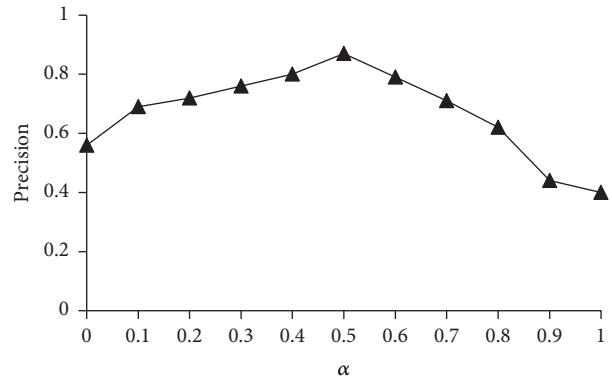

Figure 5: Precision of results for different values of α.

Core(TM) i5-4570 CPU @ 3.20 GHz, 4 GB of RAM, and 1 TB Disk. All the algorithms are implemented in Java.

4.2. Parameter Setting. The experiment in this section is to evaluate the impact of α on the precision of calculation results and provides us guidelines in choosing a good value of α. First, we randomly select 8 keywords from the query sets in Section 4.1. For each keyword k_i, we can get the corresponding top-6 related keywords by formula (10). Further, we can get 11 different sets of results by adjusting the parameter α from 0 to 1. Second, we integrate the related keywords in the above results to get the set K_i (set size $\leq$ 66). Finally, we calculate the cooccurrence rate of keywords in set K_i and k_i. Mark the keywords ranked top-10 as real set related to k_i. For keyword k_i and based on real set, the precisions under different α are as follows: the ratio between the number of keywords related to k_i through formula (10) and the total number of the received keywords (13). The precisions corresponding to 8 keywords are summed and averaged to get the final precision. Figure 5 illustrates how the precision is adjusted by the parameter α.

As shown from Figure 5, the precision of the terms similarity calculation is the maximum, 0.87, when α equals 0.5. So we choose the parameter $\alpha = 0.5$ for the following experiments.

4.3. Performance Study. In this subsection, we report the performance of the query expansion method ReInterpretQE in comparison with the state-of-the-art approaches, Metadata [3] and K-coupling [11]. The performance is measured by three evaluation metrics: precision, recall, and F-score. A corresponding SQL statement is generated for each query in the query set. The results obtained through SQL statement are perceived as real query results and added in the test set Test. Given the result set Result and real result set Test, the precision, recall, and F-score can be calculated as follows:

$$\text{precision} = \frac{\text{Result} \cap \text{Test}}{\text{Result}}, \tag{12}$$

$$\text{recall} = \frac{\text{Result} \cap \text{Test}}{\text{Test}}, \tag{13}$$

$$F\text{-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{14}$$

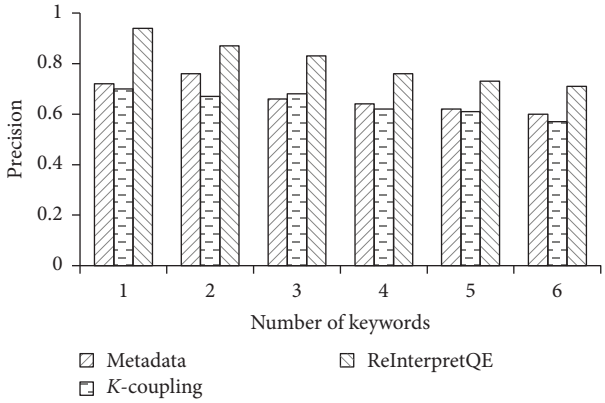

Figure 6: Precision comparison on DBLP database.

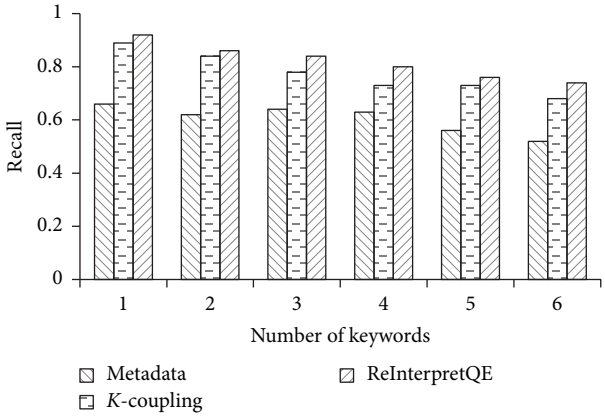

Figure 7: Recall comparison on DBLP database.

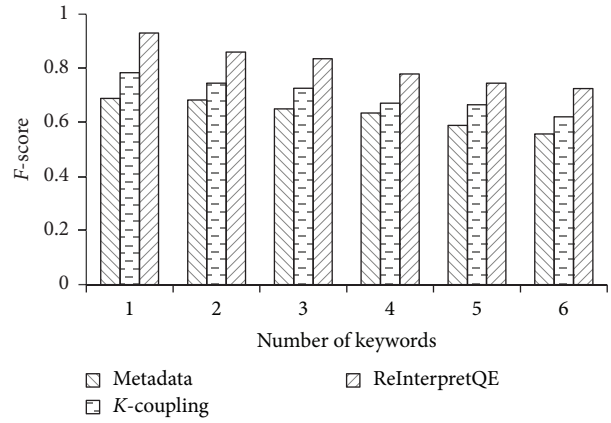

Figure 8: F-score comparison on DBLP database.

Figures 6, 7, and 8 show the comparisons in terms of precision, recall, and F-score. Each data point on the x-axis of Figures 6, 7, and 8 corresponds to the number of keywords. The y-axis presents the corresponding precision, recall, and F-score.

As we can see in Figure 6, the query expansion method ReInterpretQE proposed in this paper achieves much higher precision than the methods such as Metadata and K-coupling significantly. For example, the average precision of ReInterpretQE reaches 0.81, 20.9% and 26.6% higher than Metadata and K-coupling, respectively. Specifically, when the number of keywords is 4, the precision of ReInterpretQE is 0.76, while the precision of Metadata and K-coupling is 0.64 and 0.62. The ReInterpretQE method increases the precision by 18.8% and 22.6%, compared with Metadata and K-coupling, respectively. Overall, the precision of Metadata method is little higher than K-coupling. And the precision of ReInterpretQE is improved obviously compared with the other two. This comparison shows the significance of our proposed query interpretation, which can help to describe the semantics of keyword queries and thus significantly improve the query precision. More specifically, the reason for the poor performance of K-coupling, as compared to Metadata and ReInterpretQE, is that K-coupling method focuses on identifying a set of keyword queries related to the given keyword query. The expanded queries obtained by the K-coupling method are still keyword queries without the structure information between keywords. Yet the inherent ambiguity of keyword queries may directly affect the query precision. Methods Metadata and ReInterpretQE transform the initial keyword query into SQL and query subgraphs, respectively. The expansions of both methods contain structural information, which is helpful to improve the query precision. The reason for effective and improved expansion of ReInterpretQE over Metadata is that Metadata does not consider the multiple connection between keywords and various explanations for a keyword query, while ReInterpretQE translates the keyword query to a set of query subgraphs with structural and semantic information through the translation algorithm of triples and construction algorithm of subgraphs. The subgraphs can locate the query results more precisely.

To further evaluate the performance of our method, we vary different numbers of input keywords and compare the corresponding recalls. Figure 7 plots the average recall of the different query expansion methods. The evaluation results show that ReInterpretQE generally produces expansion of higher recall compared to the other two, suggesting that query recommendation is crucial to obtain good performance. More precisely, when the number of keywords is 4, the recall of Metadata and K-coupling is 0.63 and 0.73, respectively, while the one of ReInterpretQE is 0.80. As expected, the recall of ReInterpretQE is approximately 9.6% higher than that of the K-coupling method, and it is even higher when compared with the other method, Metadata. We observe that the methods K-coupling and ReInterpretQE beat Metadata significantly. This is so because the Metadata method does not take similar words and related items into account, while the methods K-coupling and ReInterpretQE deal with the problem accordingly, which can help to progressively and efficiently make query expansion and thus lead to higher query recall. ReInterpretQE always generates better results than K-coupling. For example, ReInterpretQE achieves 0.82 average recall, which leads to about 5.1% over K-coupling. The reason for this phenomenon is that K-coupling only uses keyword coupling relationship matrix to analyze the original query, while ReInterpretQE conducts the similar

words and related items expansion for every keyword and expands the query to a query list using probabilistic recommendation algorithm. ReInterpretQE builds Viterbi model using dynamic programming and generates the query list related to query input. After this operation, the query results can include more complete and comprehensive information. And the recall of results has been further improved.

Figure 8 further illustrates that the query expansion method ReInterpretQE outperforms the baseline methods through the comparison analysis of these methods on F-score. The overall trend is clear. At all thresholds we evaluated, the results produced by ReInterpretQE are significantly better than the other two methods. For example, when the number of keywords is 4, the F-score of Metadata and K-coupling is 0.63 and 0.67, respectively, while the F-score of ReInterpretQE is 0.78. The ReInterpretQE method increases the F-score by 23.8% and 16.4%, compared with Metadata and K-coupling. We investigated the reasons that ReInterpretQE has higher performances than the baseline methods. On the one hand, ReInterpretQE calculates the term similarity considering both intrasimilarity and intersimilarity. Thus the similarity calculation between terms is more reasonable. Then Algorithm 1, probabilistic recommendation algorithm, is proposed based on the term similarity. It constructs the Viterbi model using dynamic programming, which can improve the query recall. On the other hand, ReInterpretQE designs Algorithm 2, translation algorithm of triples, to perform the translation from keywords to triples. Then Algorithm 3, query subgraph construction algorithm, is used to transform the triples to query subgraphs. In the construction of the query subgraphs, ReInterpretQE not only considers the expansion in the structure and content of keyword query, but also considers the various explanations for a keyword query. So the query precision is further improved.

Summary. Based on this observation, we realize that ReInterpretQE indeed boosts the performance of query expansion and has a clear positive effect on quality of query results. Thus ReInterpretQE can be considered as a quite effective and practicable algorithm for query expansion.

5. Conclusions

Aiming at addressing the problems of semantic fuzziness and expression limits, the paper proposes a novel two-step query expansion method, ReInterpretQE. The method translates the keyword query to query subgraphs with potential structural and semantic information. Compared with the traditional methods, the method completes the query expansion and analysis only depending on the structure and content information of databases, without the requirement of query logs. In addition, the method uses query recommendation and query interpretation to balance the precision and recall of query results. Finally, experimental results on DBLP dataset verify the effectiveness of the proposed method. There are many open questions in the research of query expansion in relational databases. In the future work, we will make further research and discussion on it. For instance, we will take into account the influence of feedback on the performance of query expansion.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Yingqi Wang and Nianbin Wang conceived and designed the experiments; Lianke Zhou performed the experiments; Lianke Zhou analyzed the data; Yingqi Wang wrote the paper.

Acknowledgments

This work is sponsored by the National Natural Science Foundation of China under Grant nos. 61272185 and 61502037, the Fundamental Research Funds for the Central Universities (no. HEUCF160602), the Natural Science Foundation of Heilongjiang Province of China under Grant nos. F201340 and F201238, and the Basic Research Project (nos. JCKY2016206B001 and JCKY2015206C002).

References

[1] S. Bergamaschi, F. Guerra, and G. Simonini, "Keyword search over relational databases: issues, approaches and open challenges," in *Proceedings of the 2013 PROMISE Winter School: Bridging Between Information Retrieval and Databases*, pp. 54–73, Bressanone, Italy.

[2] J. Park and S.-G. Lee, "Keyword search in relational databases," *Knowledge and Information Systems*, vol. 26, no. 2, pp. 175–193, 2011.

[3] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis, "Keyword search over relational databases: A metadata approach," in *Proceedings of 2011 ACM SIGMOD and 30th PODS 2011 Conference*, pp. 565–576, Athens, Greece, June 2011.

[4] Z. Zeng, Z. Bao, M. L. Lee, and T. W. Ling, "A semantic approach to keyword search over relational databases," in *Proceedings of the 32nd International Conference on Conceptual Modeling*, pp. 241–254, Hong Kong, 2013.

[5] M. Kargar, A. An, N. Cercone, P. Godfrey, J. Szlichta, and X. Yu, "Meaningful keyword search in relational databases with large and complex schema," in *Proceedings of 2015 31st IEEE International Conference on Data Engineering (ICDE '15)*, pp. 411–422, Seoul, Republic of Korea, April 2015.

[6] L. Zhang, T. Tran, and A. Rettinger, "Probabilistic query rewriting for efficient and effective keyword search on graph data," *Proceedings of the VLDB Endowment*, vol. 6, pp. 1642–1653, 2013.

[7] Z. Zeng, Z. Bao, T. Wang, L. Mong, and L. Lee, "ISearch: an interpretation based framework for keyword search in relational databases," in *Proceedings of 3rd International Workshop on Keyword Search on Structured Data (KEYS '12)*, pp. 3–10, Scottsdale, Ariz, USA, May 2012.

[8] D. Yang, D.-R. Shen, G. Yu, Y. Kou, and T.-Z. Nie, "Query intent disambiguation of keyword-based semantic entity search in dataspaces," *Journal of Computer Science and Technology*, vol. 28, no. 2, pp. 382–393, 2013.

[9] J. Zhai, D. Shen, Y. Kou, and T. Nie, "Richly semantical keyword searching over relational databases," in *Proceedings of 9th Web*

Information Systems and Applications Conference (WISA '12), pp. 211–216, Haikou, China, November 2012.

[10] J. Yao, B. Cui, L. Hua, and Y. Huang, "Keyword query reformulation on structured data," in *Proceedings of IEEE 28th International Conference on Data Engineering (ICDE '12)*, pp. 953–964, Arlington, Va, USA, April 2012.

[11] X. Meng, L. Cao, and J. Shao, "Semantic approximate keyword query based on keyword and query coupling relationship analysis," in *Proceedings of 23rd ACM International Conference on Information and Knowledge Management (CIKM '14)*, pp. 529–538, Shanghai, China, November 2014.

[12] V. Ganti, Y. He, and D. Xin, "Keyword++: a framework to improve keyword search over entity databases," *Proceedings of the VLDB Endowment*, vol. 3, pp. 711–722, 2010.

[13] J. X. Yu, L. Qin, and L. Chang, "Keyword search in relational databases: a survey," *IEEE Data Engineering Bulletin*, vol. 33, pp. 67–78, 2010.

[14] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword search on structured and semi-structured data," in *Proceedings of International Conference on Management of Data and 28th Symposium on Principles of Database Systems (SIGMOD-PODS '09)*, pp. 1005–1009, Providence, RI, USA, July 2009.

[15] Y. Chen, W. Wang, and Z. Liu, "Keyword-based search and exploration on databases," in *Proceedings of 2011 IEEE 27th International Conference on Data Engineering (ICDE '11)*, pp. 1380–1383, Hannover, Germany, April 2011.

[16] S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: a system for keyword-based search over relational databases," in *Proceedings of 18th International Conference on Data Engineering*, pp. 5–16, San Jose, Calif, USA, March 2002.

[17] V. Hristidis and Y. Papakonstantinou, "Discover: keyword search in relational databases," in *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 670–681, Hong Kong, 2002.

[18] L. Qin, J. X. Yu, and L. Chang, "Keyword search in databases: the power of RDBMS," in *Proceedings of International Conference on Management of Data and 28th Symposium on Principles of Database Systems (SIGMOD-PODS'09)*, pp. 681–693, Providence, RI, USA, July 2009.

[19] G. Li, J. Feng, and L. Zhou, "Retune: retrieving and materializing tuple units for effective keyword search over relational databases," in *Proceedings of the 27th International Conference on Conceptual Modeling*, pp. 469–483, Barcelona, Spain, 2008.

[20] Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: top-k keyword query in relational databases," in *Proceedings of SIGMOD 2007: ACM SIGMOD International Conference on Management of Data*, pp. 115–126, Beijing, China, June 2007.

[21] F. Liu, C. Yu, W. Meng, and A. Chowdhury, "Effective keyword search in relational databases," in *Proceedings of 2006 ACM SIGMOD International Conference on Management of Data*, pp. 563–574, Chicago, Ill, USA, June 2006.

[22] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in *Proceedings of the 18th International Conference on Data Engineering*, pp. 431–440, San Jose, Calif, USA, March 2002.

[23] H. He, H. Wang, J. Yang, and P. S. Yu, "BLINKS: ranked keyword searches on graphs," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*, pp. 305–316, ACM, Beijing, China, June 2007.

[24] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding top-k min-cost connected trees in databases," in *Proceedings of 23rd International Conference on Data Engineering (ICDE '07)*, pp. 836–845, Istanbul, Turkey, April 2007.

[25] B. Kimelfeld and Y. Sagiv, "Efficiently enumerating results of keyword search over data graphs," *Information Systems*, vol. 33, no. 4-5, pp. 335–359, 2008.

[26] V. Hristidis, H. Hwang, and Y. Papakonstantinou, "Authority-based keyword search in databases," *ACM Transactions on Database Systems*, vol. 33, no. 1, article no. 1, pp. 24–63, 2008.

[27] X. Yu and H. Shi, "CI-rank: ranking keyword search results based on collective importance," in *Proceedings of IEEE 28th International Conference on Data Engineering (ICDE '12)*, pp. 78–89, Arlington, Va, USA, April 2012.

[28] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proceedings of the 15th International Conference on World Wide Web*, pp. 387–396, Edinburgh, UK, May 2006.

[29] DBLP, http://dblp.uni-trier.de/.