2012 IEEE 28th International Conference on Data Engineering

Keyword Query Reformulation on Structured Data

Junjie Yao Bin Cui Liansheng Hua Yuxin Huang

Department of Computer Science and Technology
Key Laboratory of High Confidence Software Technologies (Ministry of Education)
Peking University, P.R.China
{junjie.yao,bin.cui}@pku.edu.cn
{hualiansheng,dazzle1203}@gmail.com

***Abstract*— Textual web pages dominate web search engines nowadays. However, there is also a striking increase of structured data on the web. Efficient keyword query processing on structured data has attracted enough attention, but effective query understanding has yet to be investigated. In this paper, we focus on the problem of keyword query reformulation in the structured data scenario. These reformulated queries provide alternative descriptions of original input. They could better capture users' information need and guide users to explore related items in the target structured data. We propose an automatic keyword query reformulation approach by exploiting structural semantics in the underlying structured data sources. The reformulation solution is decomposed into two stages, i.e., offline term relation extraction and online query generation. We first utilize a heterogenous graph to model the words and items in structured data, and design an enhanced Random Walk approach to extract relevant terms from the graph context. In the online query reformulation stage, we introduce an efficient probabilistic generation module to suggest substitutable reformulated queries. Extensive experiments are conducted on a real-life data set, and our approach yields promising results.**

I. Introduction

Recent years have witnessed an increasing amount of structured data on the web, which becomes a prevailing trend [1]. As reported in a study of large-scale online user search behavior [2], the structural object is an increasing information need on general web search today, and accounts for more than half of user queries. Examples of structured data include e-commercial sites, knowledge databases and entity corpus in social media. Additionally, lots of structured data is extracted from text pages, and stored in some kinds of structured databases [3], [4]. Compared to the flat text, the structured data is able to include various associated attributes, and connections/relations between data items could also be easily represented, thus the structured data can be used to better organize online content.

The query result on structured data generally contains information stored in separate tables. Relevant pieces of data that are in different tables but inter-connected and collectively relevant to the input query are expected to be returned. In tradition, declare languages like SQL or predefined query forms are used for query on top of these databases. Due to the limitation and complexity of aforementioned query methods, the uncontrolled keyword query is more preferable to ordinary users in web search. Recently, there are lots of work on efficient and effective keyword search processing on structured data [5], [6], [7], [8]. Query suggestion is an promising alternative to improve the effectiveness of keyword search, however it has yet to be investigated in structured data scenario.

In reality, it is difficult for users to specify exact keywords to describe their requests in web search. This is mainly caused by natural language ambiguity or semantic "mis-match" gap between query words and underlying data. Therefore, understanding query intent behind simple query is a challenging task for effective search operation [9]. A query could be ineffective due to many reasons, among which miss-specification is a common one. Let's consider the following use cases, in the setting of bibliographic database DBLP [1]:

- **Similar words:** where two different words have the same meaning, but with different expressions. When a user searches papers related to "XML", he may not know authors also use "semi-structured" and "tree" to describe this similar concept. "probability" v.s. "uncertainty" is another quasi-synonym pair.
- **Related items:** where similar topics lie in other parts of the searched corpus. For example, a user searches R. Agrawal's work on "association rule", he may not know there also exist some related topics, e.g. "sequential pattern" and "frequent itemset". Other prominent researchers, such as Jiawei Han and Raghu Ramakrishnan are also experts in this area.

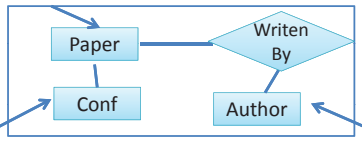
Better understanding the user intent and providing substitutive query are important requirements to improve the effectiveness of search engine, and query suggestion has been widely applied in text retrieval or web search. After a user submits a query, the typo modification or more appropriate queries are suggested. By collecting the query log or learning the user behavior, most of existing methods exploit large query logs to obtain similar queries to support query rewriting or generation, e.g., [10], [11]. For areas short of query logs, some previous work have also tried statistical extraction methods over text corpora, e.g., [12], [13].

However, these approaches are initially proposed for flat text documents, and cannot effectively model the attribute facets and connections within the structured data. First, we lack of structured query logs in structured data scenario. Second, there does not exist suitable document definition over

[1] http://dblp.uni-trier.de/

1084-4627/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDE.2012.76

ID	Paper Name	Year
P1	Threshold query optimization for uncertain data	2010
P2	Probabilistic string similarity joins	2010
P3	Efficient rank based KNN query processing over uncertain data	2010



ID	Conf Name
C1	SIGMOD
C2	PODS
C3	ICDE

ID	Author Name
A1	Sunil Prabhakar
A2	Ke Yi
A3	Xuemin Lin

Fig. 1. Tables and References in Structured DBLP Data

structured data. The ad hoc definition of results depending on input query is a double-edge sword of structured data. It is difficult to enumerate all possible types of query results, and the predefined virtual document approach [14] is not feasible. Even if we can manually define virtual documents by selectively extracting several tuples, this kind of virtual document is rather arbitrary and limited to extend.

In this paper, we investigate the problem of keyword query reformulation over structured data. Given an initial query, instead of query expansion or relevance feedback, we expect to suggest new but substitutive queries by exploiting the structural semantics of underlying data sources. Compared to some recent returned result analysis methods [15], [16], our approach provides more global indications and explores to a broader scope. Intuitions behind our approach lie at the ubiquitous cross-linked connections of structured data. It contains rich semantic information indicating the word closeness and context scenario. It is intuitive to understand that the surrounding text or the relevant tuples/attributes of the similar terms are also similar.

Figure 1 presents an example of table structure in DBLP. Different tables have fixed information and they are connected via key references. In this setting, "probabilistic" and "uncertain" in the motivating example usually appear in same conferences or are used by same researchers, while they may not appear together simultaneously, say in a title. Authors connected by similar conferences or using similar terms in their paper titles, could have similar research interests, though they have not ever collaborated.

In this work, we model the structured data with a heterogenous graph and propose an automatic approach for keyword query reformulation which consists of offline term relationship extraction and online query reformulation. In the offline stage, we resort to the rich structural context information and capture the term relation. Specifically, we build a term relation graph, where "term" refers to a textual word or phrase in the database. For each term, we employ an improved random walk model over the graph to obtain a list of most similar terms; we also compute the closeness of terms to measure the semantic distance in the graph. At the running time, given a query composed of several keywords, we propose a probabilistic query generation model to find top-k related queries, consisting with similar keywords of input query. Compared to those prior keyword query processing and result ranking methods, our work is complementary and focuses on query understanding by taking advantage of structural information in the underlying data sources.

The contributions of this paper could be summarized as follows:

1) We propose a graph structure to model the semantic relationships among terms in the structured data, which consist of not only tuple content but also interaction relations.
2) We present an improved Random Walk model to obtain the term similarity over the graph scope.
3) We design a novel probabilistic query generation model that determines good substitutive queries for an input query. Under the generative framework, we further discuss efficient top-k reformulated queries selection.
4) We conduct extensive experiments on real-life data to verify the effectiveness and efficiency of our methods.

The remainder of this paper is organized as follows. We first review related work, and then introduce the problem definition and solution overview in Section III. Next two sections provide details of our approach. Section IV describes the term relation extraction process. The reformulated query generation is presented in Section V. We show the experimental study in Section VI, and finally conclude this paper.

II. Related Work

In this section, we make a brief review of the literature in several categories:

Keyword Search over Structured Data: As shown in [3], there exist tens of millions of useful and query-able tables on the web. Keyword search over structured data has attracted lots of attentions in recent years [5], [6], [7], [17], [18], [19]. The result definition and query processing are studied extensively in the literature [20], [21], [22]. [20] introduced a search algorithm on the graph to generate search results. [21] utilized a PageRank strategy to assess the relevance and authority of nodes in the keyword search. [6] presented the top-k results ranking in relational databases.

Though there are also some tentative work towards keyword query understanding [23], [15], [24], [16], these previous work either chose manually created transform rules, or focus on local result analysis. The global scope and rich semantic structure information were usually ignored in previous work.

Node Similarity on Graph: Graph is an attractive model to describe the complex interactions and connections in many applications. Usually, existing keyword search methods model the structured data as a graph, where tuples in each table are connected by foreign key references. Together these tuples form a search result. There exist some work on applying graph

algorithms to find related tuples [21], summarize zuschema of relational database [25] and cluster the graph [26].

How to measure the similarity on the graph is a key issue for graph-based approaches, and different solutions have been proposed to address this problem, such as link analysis [11] and random walk model [27].

Query Suggestion/Recommendation: Search operation is typically an interactive progress, in which users iteratively submit their unknown questions and get informed. Query suggestion is commonly used by current web search engines to help users reformulate their queries with less effort. Traditional query expansion or relevance feedback generally extend the initial query, and usually causes the query intent drift. Shown in web query logs, a user usually modifies a query to another closely related one, and around 50% user sessions have reformulation actions [10].

Inspired by these, automatic query reformulation (a.k.a. rewriting, substitution) has become an attractive approach for query suggestion [10]. In query reformulation, a totally new query is suggested, which is similar or related to the initial one. Most of existing automatic query reformulation methods utilize large scale query log mining techniques, e.g., [11], [10]. Statistical query generation is also discussed in [13], [12]. However, these work are not applicable in structured data, resulting from absence of keyword query log and dynamic linkage nature of structured data.

III. Overview

A. Problem Formulation

We first introduce the notations and concepts used in the paper. Specifically, the structured data in this paper refers to relational data, which consists of tables and references between tables. An example of structured DBLP data has been show in Figure 1.

Usually, a database could be viewed as a labeled graph, where tuples in different tables are treated as nodes, and foreign key references correspond to connections between nodes. Note that a graph constructed in this way has a regular structure because the schema restricts the node connections. The proposed solutions are also applicable to other kind of schema or even schemaless structured data, e.g., XML, RDF and graph data.

Definition 1 (Tuple Graph): A tuple graph is a graph $G = \{V, E\}$, where V is the set of nodes, and each node corresponds to one tuple; E is the set of edges, and each edge represents a foreign key reference.

In Figure 1, we could replace the inner schema graph with tuple nodes and transform it to a tuple graph.

Definition 2 (Keyword Query): An input keyword query Q consists of a set of distinct keywords, i.e., Q= $[q_1, q_2, \ldots, q_n]$, where each keyword q_i $(i = 1, ..., n)$ is a word or a topical phrase, depending on the tokenization/segmentation.
Without loss of generality, we simply use *term* to represent the word or phrase in the following discussions.

Definition 3 (Query Result): Keyword query results are the substructures that connect the matching nodes (in the tuple graph). In a graph data model, a query result is defined as a subtree of the data graph where no node or edge can be removed without losing connectivity or keyword matches [5], [20].

Be aware that, our work does not focus on: 1) how to present a new result definition to keyword query; 2) how to process keyword query Q efficiently. Instead, our work addresses the problem of how to reformulate the original query and present substitutive queries to the users, which is complementary to these prior work.

Definition 4 (Reformulated Query): Given an input query Q= $[q_1, q_2, \ldots, q_n]$, find k reformulated queries Q'_i $(i = 1, ..., k)$ with k largest $score(Q, Q'_i)$, where $score(Q, Q'_i)$ evaluates the substitution validity between Q and Q'_i.

The generated new query should be similar to original input and also accord with the structural semantics in the underlying database. As a combination of individual terms, the straightforward similarity between queries could be measured by aggregation of similarity between the corresponding terms. However, it is not surprising that even each term of the new query is similar to the corresponding ones in the original query, together the new query do not cover any meaningful result, because these terms do not occur together or semantically relevant amongst them. $score(Q, Q'_i)$ should include both similarity and utility metrics, and the quality details of reformulated queries will be introduced in Section V.

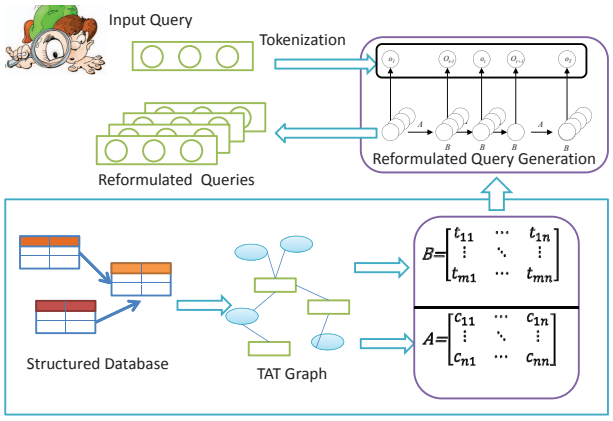


Fig. 2. Flowchart of Reformulated Query Generation

B. Proposed Approach

The flowchart of our approach is shown in Figure 2. In the off-line stage, we model the structured data in an enriched graph and then deploy term relationship extraction, i.e., similarity and closeness shown in the figure. The online stage accepts submitted query and then fetches the corresponding terms' information. With an efficient probabilistic model to reduce the search space, we conduct query reformulation and present top-k reformulated queries to end users.

The off-line stage uses an improved random walk model to measure the term similarity and details will be presented in

Section IV. Compared to previous graph similarity measurements [27], [11], the new model personalizes the preference manner to guide the similarity computation on the graph. This kind of similarity is more suitable for similar term extraction in our complex heterogenous graph, and its advantage will be demonstrated in the experimental study.

The online stage utilizes the pre-computed results, i.e., term similarity and closeness, and generates reformulated queries. To facilitate efficient processing, we employ a probabilistic model to reduce the search space. It is shown that this simple but effective approximation does not degenerate the performance of query reformulation and additionally, the generative nature of this probabilistic model also shows advantage in multiple metric fusion, especially in our reformulation objective functions.

IV. Term Relation Extraction

In this section, we first propose a heterogenous graph structure to model the structured data. Based on the graph model, we further discuss how to extract term relationships, i.e., similarity and closeness, by exploiting the structural semantics from the underlying data sources.

A. TAT Graph Representation

In order to model term relationships in structured data, we design a ***t**erm **a**ugmented **t**uple graph* (TAT graph), which is defined as follows.

Definition 5 (Term Augmented Tuple Graph): $\mathcal{G} = (V \cup V_t, E \cup E_t)$ is a *term augmented tuple graph*, where 1) each node v in V is a *tuple node* that represents one tuple; 2) each node v_t in V_t is a *term node* that represents one term; 3) each edge e in E connects two tuple nodes, representing a foreign key reference; 4) each edge e_t in E_t connects one tuple node v and one term node v_t, if v_t appears in the corresponding tuple v.

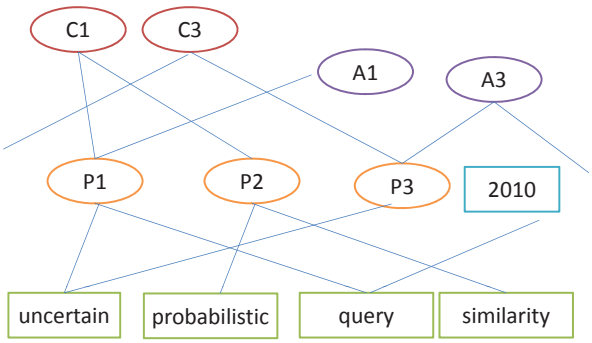


Fig. 3. Term Augmented Tuple Graph

In Figure 3, ellipse nodes are original tuples and rectangle nodes are term nodes extracted from paper title field of the DBLP example shown in Figure 1. Authors are connected to their published papers. The papers are associated with the conferences/journals and extracted terms. Connections between heterogeneous nodes represent the co-occurrence or key reference between them. To define term nodes in our graph, we would like to extract multiple terms from a field consisting of a long term sequence, e.g., paper title. However, for some tuples of which all terms stand together for certain semantic meaning, e.g., author name or institute name, segmentation should not be applied and these nodes are also searchable as simple term nodes. Term nodes with same text extracted from different fields, are considered as different. We label them with field identifiers. That is to say, words from conference names are distinguished from words from paper titles with different field labels.

Different from traditional tuple graph, TAT graph semantically incorporates tuple and term relationships in a uniform way. We believe this is an intriguing character of graph structure, which can model rich direct and indirect connections amongst structured data. In Figure 3, authors who do not collaborate actively but share similar research interests, could also be connected by similar conferences/keywords. It is possible that keywords used by similar authors belong to related research areas, though they do not have to co-occur in same paper titles.

In TAT graph, the unified modeling of terms and tuples enables us to better represent and extract their similarity or closeness relations. Later in this section, we will also discuss the novel weight method over TAT graph.

B. Term Similarity in Linked Context

We first discuss how can term similarity be extracted from the aforementioned TAT graph structure. Compared to flat representation, graph structure is more capable of modeling relations between terms. In contrast, using "frequent co-occurring" to model term similarity [15] ignores rich structural semantics in the relational databases.

Take DBLP data as example, to measure similarity between two authors, a natural way is to represent each author with associated publications or conferences. However, it is not a trivial task to choose appropriate representation subsets. Too short representation cannot provide sufficient information, while too long representation is usually very sparse. Additionally, this kind of the representation could not capture indirect connections between the authors.

The graph structure we used here alleviates these problems. It could carry more global correlations between two nodes and also their multi-facet relations, i.e., different connection paths between them. For example, in Figure 1, the connection between two authors could be measured not only by shared conferences or co-authored papers, but also the shared keywords occurred in their paper titles or even indirect similar keywords on the graph.

Random walk is considered as an effective method to measure node proximity over graph [25], [27]. In this subsection, we propose a novel term similarity measurement, derived from the TAT graph. Different from traditional simple random walk methods, we propose to design a novel contextual preference vector and weighting flow to bias the algorithm behavior.

1) Basic Random Walk Model: Random walk is an iteration process to update node scores across the graph. After several

iterations, the random walk process will usually converge to a stable state. After that, relative importance of each node is measured by its stable score.

Considering a general graph $G = (V, E)$ with node set V and edge set E, A is its adjacency matrix. $\vec{r}$ is a *preference vector* inducing a probability distribution over V, Random Walk vector $\vec{p}$ over all vertexes V is defined as the following equation:

$$\vec{p} = \lambda \cdot A \cdot \vec{p} + (1 - \lambda) \cdot \vec{r} \quad (1)$$

In the iteration process, a random walker starts from a node and chooses path randomly among the available outgoing edges, according to the edge weights. With a restart probability, he goes back to a node based on the weighted distribution in $\vec{r}$. After several iterations, node scores will converge. This is guaranteed in most cases [28].

If preference vector $\vec{r}$ is uniform over V, $\vec{p}$ is referred as a global random walk vector. For non-uniform $\vec{r}$, the solution $\vec{p}$ will be treated as a *personalized random walk* vector [29]. By appropriately selecting $\vec{r}$, the rank vector $\vec{p}$ can be made to bias certain nodes. In most personalized cases, when the ith coordinate of $\vec{r}$ is 1 and all others are 0, the random walk will be referred as the *individual random walk* [27].

To extract a specific node t_0's similar ones, we can assign a preference vector $\vec{r}$ biased to t_0 and then run the random walk. The convergence score is a measurement of relative similarity, i.e.,

$$sim(t_0, t) = \vec{p}[t]|\vec{r}_0. \quad (2)$$

$\vec{r}_0$ is the preference vector with t_0's corresponding weight as non-zero and others zero. The above equation means that, in an individual random walk model biased to t_0, if $\vec{p}[t_1] > \vec{p}[t_2]$, node t_1 is more similar to t_0 than t_2.

Different from the homogeneous graph, here we only extract similar nodes belonging to same classes of the initial node. For example, given an author node, we expect to find similar author nodes, while ignoring similar keyword or conference nodes found in the random walk process.

2) Contextual Biased Preference: To extract similar nodes of node t_0 in random walk, a straightforward similarity computation solution is to bias t_0, by assigning t_0's corresponding element in $\vec{r}$ with 1 and 0 for others. In an initial trial, we find that this approach is locally sensitive. For example, given an author in Figure 3, how to find similar authors interested in same conferences/research areas? An individual random walk starting from a specific author could only find some close collaborators. Since similar terms generally appear in similar context, here we further model a contextual preference vector as one improvement of random walk. Starting random walk process from this author's primary conference and research areas, we may encounter other valuable findings. Specifically, we utilize the surrounding tuples/terms as the context setting for a given term/tuple node. This context setting is used for node similarity extraction in our proposed approach.

Definition 6 (Context Node): For a term node, the context is tuple nodes it belongs to. Tuple node's context is the affiliated term nodes and tuple nodes it connects to. Nodes in these context are called context nodes.

There probably exist several different fields in context nodes. Here, the field means type of node, e.g., tuple or attributes in a relational table. We fetch some top related nodes from each field into the contextual vector. Here, to better model the context information, we define two metrics, i.e., field weight and node weight in each field.

We first measure the field weight. Given a starting node t_0, with associated context nodes from m fields. We resort to the cardinality statistics. We weight each field as the fraction of its cardinality $|F_i|$ and t_0's frequency $freq(t_0)$, $1 \leq i \leq m$. Second, we measure a context node v_c's weight with respect to starting node t_0 as $freq(v_c, t_0) \cdot idf(v_c)$. $freq(v_c, t_0)$ measures the co-occurrence statistics with t_0, and the $idf(v_c)$ is the inverse of global occurrence statistics of node v_c. Given the above two weight measurements, we can compute v_c's corresponding weight $w(v_c)$ which is the combination of affiliated field weight and node weight, i.e, $1/|F_i| \cdot freq(v_c, t_0) \cdot idf(v_c)$. The reason we choose this kind of combination is that, the representability of field and node affect the behavior of random walk. It could guide the performance of node similarity extraction.

Algorithm 1 Contextual Random Walk based Term Similarity Extraction

Input: starting node t_0 for similarity extraction,
TAT graph G with linkage information A,
damping parameter in random walk λ.

```
F ← {F_1,F_2,...,F_i,...} // context fields of t_0.
C ← {c_11,c_12,...,c_ij,...} // context nodes of t_0.
for c ∈ C do
    c_ij = 1/|F_i| · freq(c_ij, t_0) · idf(c_ij)
end for
r ← [0, ..., c_ij, ...] //preference vector.
repeat
    p_t = λ · A · p_{t-1} + (1 − λ) · r
until |p_t − p_{t-1}| < ε or predefined iteration times
return score vector p.
```

We illustrate our random walk based term similarity extraction process in Algorithm 1. First we select the context preference with regard to the starting node t_0. We give a preference vector $\vec{r}$ for random walk process, where 0 is set for all other non-context nodes in the graph. Then we conduct the random walk until it converges or reaches predefined iteration times. After iteration, a list of top similar nodes of t_0 is returned, i.e., the most similar terms extracted by our model.

Figure 4 depicts the difference between basic random walk and the contextual preference biased enhancement. In the basic model, we could only extract terms highly co-occurred with "uncertain", i.e., "similarity" and "query" in the figure. The results are similar to the frequent co-occurrence based similarity measurement, which could not provide useful latent information. With the help of contextual preference, we first identify some preference nodes, i.e., two paper tuple nodes in

the figure, and the new random walk model could go through the contextual part of the graph to find other relevant terms, e.g., "probabilistic" in the figure.

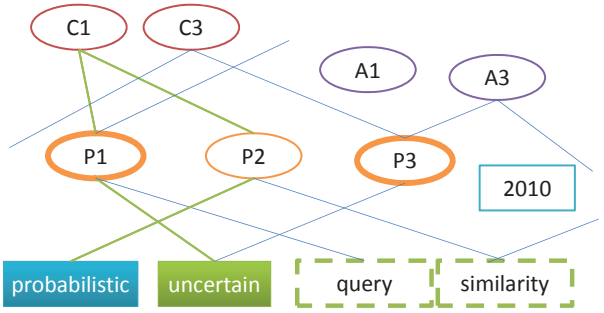


Fig. 4. Random Walk on TAT Graph. Starting node is "uncertain". Basic random walk could only find dot lined "query " and "similarity". In our enhancement, we first grasp the contextual node as P1, P3, then from them, we extract the "probabilistic" as a more similar node.

C. Term Closeness Extraction

As introduced in Section III, term closeness is another important relation measurement, capturing the distance between term nodes. This is an indication of two terms' keyword search result coverage, which is an important metric for valid reformulated queries. Shorter paths between two terms indicate a closer relation and a good corpus coverage. At first glance, it looks similar to the previous similarity metric based on random walk. However, the random walk method combines the global graph structure to derive node similarity and it mixes different paths between two nodes. This is inappropriate for node closeness.

A straightforward solution of term closeness is to fetch the exact search result and then count the returned items. However, as we have to construct the keyword search results on the fly, this is especially prohibitive, considering that every possible combinations of each candidate term pairs need to be evaluated. For example, if we have an input query with three keywords which is relatively short, every query term has 10 candidate similar terms, then we need to compare the closeness of 10*10*10 term triples. A feasible approach is to summarize the target corpus by term pair coverage, and estimate the result size of each query. Compared to previous result estimation or database summary work [30], here we have to validate a much larger number of substitutable queries, and also maintain the frequency and length information of paths, which were somewhat ignored before.

In this part, we use an efficient path search method to compute the shortest paths between two term nodes, and choose the summarization over all possible paths as the closeness metric. The formulation is defined as:

$$clos(v_i, v_j) = \sum_{\tau: v_i \to v_j} \frac{1}{len(\tau)} \quad (3)$$

In the first stage, we search each node's shortest path to others. Starting from a node v, we identify the direct connected ones, which have distance 1. From these, we then expand to distance 2. Distance $i+1$ nodes can be easily derived from distance i ones in each iteration. We maintain top ones and prune less frequent to guarantee the extraction performance. At the second stage, we collect the nodes' connectivity information and compute the associated nodes' closeness based on the above equation.

target term	ranked close terms	ranked close conferences
"probabilistic"	"generation", "document", "distribution", ... "boundary" ...	VLDB, SIGMOD, AAAI, ... ICDM ...

TABLE I
EXTRACTED CLOSE TERMS

Table I presents an example of close terms extracted for term "probabilistic", which is reasonably intuitive. From the table, we can see that it has close connection with "generation" and "distribution", as they appear together more often, while its connections with "strict" or "boundary" are loose. Regarding to the close conferences, we find it has more close relation with database and artificial intelligence conferences than counterpart ones in data mining areas. Interestingly, this finding is quite consistent with the results returned by Google search engine with a simple keyword search test. We get more than 150 thousand results using keywords "probabilistic" + "VLDB"/"SIGMOD"/"AAAI", while only around 50 thousand results with keywords "probabilistic" + "ICDM".

V. REFORMULATED QUERY GENERATION

In this section, we will introduce the online processing of query reformulation over structured data. We first define the quality of reformulated query, and then propose an efficient probabilistic model for query reformulation.

A. Quality of Reformulated Queries

Given an input query $Q = [q_1, q_2, ..., q_m]$ and each query term's similar term extension list $q_i \Rightarrow \{t_{i1}, t_{i2}, ..., t_{in}\}$, merely integrating these similar term lists is not enough. Sometimes, though $sim(q_i, t_{il})$ and $sim(q_j, t_{jk})$ are both very high, t_{il} and t_{jk} may have no connection in the structured data, i.e., they are meaningless when combined. Note that, the keyword search over relational database generally returns a set of tuples, each of which is either obtained from a single relation or by multiple joined relations. We could not arbitrarily combine top ranked terms together.

Therefore, we identify two requirements for query reformulation on structured data: 1) **Cohesion**: The generated query should be a "valid" one and covers several tuples in the targeted structured corpus. Random selection from different term extension lists cannot guarantee this requirement. 2)

Similarity: The generated one should be similar to the original input in terms of some semantic concerns. These two measurements are separately represented by term closeness and term similarity.

The reformulation ability of new query Q' with respect to Q could be the aggregation of similar term pairs in the two queries. The extraction processes of these relations are well discussed in the prior section. We measure the reformulated probability of Q' given input query Q as:

$$p(Q'|Q) = p(q'_1)p(q_1|q'_1)\prod_{i=2}^{m} p(q'_i|q'_{i-1})p(q_i|q'_i)$$
$$= p(q'_1)sim(q_1,q'_1)\prod_{i=2}^{m} clos(q'_i,q'_{i-1})sim(q_i,q'_i) \quad (4)$$

In Equation 4, the multiplication is sensitive to zero. For example, given an input query Q with m terms, most terms in a reformulated query Q' are similar and close to those of Q. A merely mismatch of two terms in Q' may lead Q' to fail, i.e., $clos(q'_i,q'_{i-1})$=0, then Q''s score becomes 0. Here we use a global indication to smooth the scoring function of reformulated queries.

$$sim_{smo}(q'_i,q_i) = \lambda \times sim(q'_i,q_i) + (1-\lambda)\times\sum_{k=1}^{m} sim(q'_k,q_k) \quad (5)$$

$$clos_{smo}(q'_{i-1},q'_i) = \lambda\times clos(q'_{i-1},q'_i)+(1-\lambda)\times\sum_{k=2}^{m} clos(q'_{k-1},q'_k \ ) \quad (6)$$

This smooth process keeps the aggregated scores unchanged in order to maintain the probabilistic meaning of the parameters. And the smooth effect is controlled by parameter λ. The smooth procedure can help alleviate the problem when candidate query contains trivial words or irrelevant word pairs.

B. Probabilistic Query Generation Model

The search space for a candidate reformulated query is $O(n^m)$, given an input query with m terms and each term with a similar extension list in size n. Obviously, it is impossible to enumerate the huge search space for online query reformulations. Though we introduce a feasible quality measurement, the efficient query generation still requires an effective search space reduction.

We resort to a novel algorithm to find the top-k generated queries. It reduces the search space and also provides a probabilistic perspective to view this query reformulation problem. This guarantees efficient reformulated query generation in an interactive response time. Here we utilize a Hidden Markov Model (HMM) [31] to profile query reformulation.

In general, HMM [31] is composed of a set of hidden states S = $\{s_1,s_2,\ldots,s_N\}$, and observation symbols O = $\{o_1,o_2,\ldots,o_M\}$. It is a generative model with multiple steps. At each step t, the model is in a specific hidden state $q_t \in S$ and we observe an observation symbol $e_t \in O$. Having output a symbol, a transition is made to the next state.

The following three components describe an HMM behavior:

- the state transition probability, $A_{N,N}$ = $\{a_{ij} \parallel 1 \le i,j \le N \}$, where a_{ij} = $P(q_{t+1} = s_j \parallel q_t = s_i)$.
- the observation probability distribution in each state, $B_{N,M}$ =$\{b_{ij} \parallel 1 \le i \le N,\ 1 \le j \le M \}$, where b_{ij} = $P(e_t = o_j|q_t = s_i)$ is the probability of observed o_j in state s_i.
- the initial state distribution, π_N = $\{\pi_i \parallel i \in$ [1,N]$\}$, where π_i = $P(q_1 = s_i)$ is the initial probability of state S_i.

In query reformulation scenario, initial input query Q = [q_1, q_2 , …, q_m] is considered as observed symbols, and the hidden state set consists of all corresponding similar terms, i.e., q_1's similar term list : t_{11}, t_{12}, …, t_{1n}, q_2's similar term list: t_{21}, t_{22}, …, t_{2n}, …and q_m's counterpart: t_{m1}, t_{m2}, …, t_{mn}. A reformulated query of length m is composed of m similar terms, with the term in position i is selected from one of q_i's similar term list. Here we describe one candidate reformulated query as q'_1, q'_2, …, q'_m. This reformulated query corresponds to a hidden state sequence in HMM.

By extending this model to allow the original term existing in the new reformulated query or deletion of initial terms, we could also easily add the void and original states in each q_i's corresponding similar term list. That is to say, we could also suggest new queries containing terms existed in input query.

The emission probability of each query term t_{ij} in $L(q_i)$ (q_i's similar term list) to q_i is measured by similarity between t_{ij} and q_i discussed in Sec IV-B. The transformation probability between hidden states q'_i and q'_{i+1} depends on their closeness, i.e., $clos(q'_i,q'_{i+1})$. In this paper we use a first-order HMM, where the transition from $t-1$ to t is memoryless, and independent of state in $t-2$.

Given an input query, the distribution of initial hidden state is derived from the similarity metrics between states and the first observation:

$$\pi(t_{1j}) = \frac{1}{Z_t} freq(t_{1j}), 1 \le j \le n. \quad (7)$$

Z_t is a normalization factor over t, i.e. $\sum_{l=1}^{n} freq(t_{1l})$.

The probability of making a transition to q'_i when being in state q'_{i-1} depends on how often the terms q'_i and q'_{i-1} appear together, i.e., the closeness between these two terms.

$$A_{q'_{i-1},q'_i} = clos(q'_{i-1},q'_i), \quad (8)$$

The emission probability of a term t_{ij} to observed query term q_i is measured by

$$B_{t_{ij},q_i} = \frac{1}{Z_B} sim(t_{ij},q_i) \quad (9)$$

Z_B is a normalization factor over t', i.e.,$\sum_{j=1}^{n} sim(q_i,t_{ij})$.

In the HMM framework, good reformulated queries can now be determined as those hidden state sequences with a high probability of being traversed while emitting the original query Q = [q_1, …, q_m].

Given an input query, this model turn similar terms into reformulated queries. It could be regarded as a translation

process. Users have an information need, which could be described by different queries in the hidden state space of HMM model. The initial input query is an explicit representation of this information need. While we would like to find some good hidden queries relevant to initial query and cohesive in quality.

Under this modeling, we use the following metric to evaluate the quality of generated queries, and the top quality reformulated queries are returned.

$$p(Q'|Q) = \pi(q'_1)B(q'_1, q_1)\prod_{i=2}^{m} A(q'_{i-1}, q'_i)B(q'_i, q_i)$$
$$= \pi(q'_1)\prod_{i=1}^{m} B(q'_i, q_i)\prod_{i=1}^{m} A(q'_{i-1}, q'_i) \quad (10)$$

In the above equation, the first part is an aggregation of similarity over terms in hidden state sequence and second part with closeness. The HMM model naturally models query generation process, incorporating both term closeness and similarity together.

Algorithm 2 Extended Viterbi for Top-k Queries

Input: query $Q = [q_1, q_2, ..., q_m]$
term candidates QC=(q'_{11},q'_{12},...,q'_{1n},...,q'_{ij},...,q'_{mn}).
closeness and similarity relation matrix A and B.
$L[c, i, *]$: path list with no more than k paths in step c ending with hidden state i
$S[c, i, *]$: score vector of corresponding paths in $L[c, i]$

1: **for** $c = 1, \ldots, m$, each step in HMM **do**
2: **for** $i = 1, \ldots, n$ **do**
3: **if** $c = 1$ **then**
4: $S[c, i, 1] \leftarrow p(q'_{1i})B(q_1, q'_{1i})$, $L[c, i, 1].append(i)$
5: **else**
6: $S[c, i, *] \leftarrow$ top-k scored: $S[c-1, j, l]A(q'_{c-1,j}, q'_{ci})B(q_c, q'_{ci})$
7: $L[c, i, *] \leftarrow$ top-k scored: $L[c-1, j, l]$.append(i)
8: **end if**
9: **end for**
10: continue next step in HMM.
11: **end for**
12: **return** at step m, top-k sequences from $L[m, *, *]$.

C. Top-k Reformulated Query Generation

Given the initial query as an observed output sequence, and extracted similarity/closeness relation, we need to find top-k reformulated queries with highest scores. In an HMM model with its known parameters and the observed output sequence, finding the hidden state sequence that is most likely to generate the observed output is a common problem. This can be solved by the Viterbi algorithm [31], which is an efficient dynamic programming algorithm.

As Viterbi is limited to top-1 problem, here we present two extensions to find top-k reformulated queries. One is a simple variant of Viterbi algorithm, and another is enhanced with an A* best first pruning search strategy. We outline the algorithms in the following. The empirical comparison will be reported in the experiment section.

Algorithm 2 extends the storing of temporal conditions to top-k best sequences. The subsequence ended is enough to ensure the top-k scored queries found. The time cost of this algorithm is $O(mn^2k \log k)$, that is $k \log k$ times slower than the Viterbi algorithm. The time complexity of Viterbi algorithm is $O(mn^2)$, and its space complexity is $O(mn)$. A significant portion of the HMM state space can be pruned during the extraction. In the forward stage of Viterbi algorithm, states with zero or low closeness with the previous state could be discarded. Equation 10 implies that the step by step computation could eliminate some irrelevant states.

Algorithm 3 A* Strategy for Top-k Queries under Viterbi

Input: query $Q = [q_1, q_2, ..., q_m]$
term candidates QC=(q'_{11},q'_{12},...,q'_{1n},...,q'_{ij},...,q'_{mn}).
closeness and similarity relation matrix A and B.
incomplete path set $IP = [\,]$, complete path set $CP = [\,]$

1: run Viterbi algorithm along Q and QC for top-1 query, get $S[c, i, 1]$.
2: $h[c, i] \Leftarrow$ state score $S[c, i, 1]$
3: **for** $i = 1, \ldots, n$ **do**
4: $L_i \leftarrow i$
5: $L_i.h\_score \leftarrow h[m, i]$
6: $L_i.g\_score \leftarrow 1$
7: IP.insert(L_i)
8: **end for**
9: **while** $IP \neq \emptyset$ **do**
10: $L\_temp \leftarrow$ max(IP), remove L_temp from IP
11: **for** $i = 1, \ldots, n$ **do**
12: $c \leftarrow m - L\_temp.length$
13: $j \leftarrow L\_temp.last$
14: $L\_augment \leftarrow L\_temp$.append(i)
15: $L\_augment.h\_score \leftarrow h[step, i]$
16: $L\_augment.g\_score \leftarrow L\_temp.g\_score \cdot A(q'_{c-1,i}, q'_{cj}) \cdot B(q'_c, q'_{cj})$
17: **if** $L\_augment.length = m$ **then**
18: reverse $L\_augment$
19: CP.insert($L\_augment$)
20: **else**
21: IP.insert($L\_augment$)
22: **end if**
23: **if** $|CP|$ = k **then**
24: break;
25: **end if**
26: **end for**
27: **end while**
28: **return** CP

We proceed to present an improved query generation algorithm, sketched in Algorithm 3. In its first stage, Viterbi algorithm runs. Utilizing the information memorized by Viterbi, next in the line, an A* search strategy is performed to

determine top-k state sequences. The improvements of this new algorithm lie at: 1) it first utilizes efficient Viterbi to collect state connection information. 2) A* search strategy in second stage prunes irrelevant states.

In this algorithm, list CP contains complete paths from head to the tail hidden state, while IP holds incomplete paths to be augmented. Both CP and IP only need to hold at most k paths. g_score preserves an incomplete path's score, while h_score has the optimal score it can still get in the following steps based on its condition. An incomplete path with its potential optimal score beyond top-k can be safely filtered out. Given h_score previously calculated in Viterbi stage, it is guaranteed that there exists one complete augmentation from the current part which reaches score $g \cdot h$.

VI. Experiment

We conduct an empirical study on the proposed query reformulation approach. The efficiency and effectiveness aspects of the proposed approach are reported.

The experiment is conducted on the bibliographic dataset, DBLP. We extract tuples and terms from the DBLP data, and create the graph representation of it. We use a recent dataset release with about 700 thousand authors, 1.3 million papers and 4.5 thousand conferences. We then bulk this dataset into a MySQL database and create inverted index using Lucene [2]. Codes are written in Java and C#. The programs are performed on a Windows server with 4 core processors and 12G memory.

A. Effect on Term Similarity

One major contribution of our work is that we model the term similarity in a link context which can effectively profiles the characters of structured data. Specifically, we propose an enhanced random walk based similarity extraction method, which can facilitate "semantically" related term extraction. In this experiment, we first evaluate the effect on term similarity of different approaches. The difference between basic random walk and our proposed method has been discussed in Figure 4. Here we further compare the proposed random walk based method with the co-occurrence similarity approach [15], which extracts the similar terms with frequent co-occurring counts.

Table II presents a similar topic extraction case, in which target term "xml" is a popularly used word in paper titles. In traditional frequent co-occurrence method, only some correlated words can be selected. For example,"document", "integration" and "indexing" are subareas in XML research area. From them, we are able to understand specific topics given the input term. However, these co-occur terms could not indicate the other alterative meanings of input term. With the help of contextual Random Walk, we find some interesting similar or counterpart topics, such as "twig", "native" and "keyword", "html". Though there is still some noise in this kind of similar term extraction, we could find that the proposed approach is better at providing alternative and intuitive topics related but slightly diverse from original "xml".

target term	extract method	top ranked similar terms
"xml"	frequent co-occurrence	"document", "relation", "structure", "index", "processing", "integration", "schema"
	contextual Random Walk	"twig", "native", "dtd", "holistic", "keyword", "fragment", "html"

TABLE II
SIMILAR TERMS EXTRACTED THROUGH DIFFERENT METHODS

We could conclude that, the frequent co-occurrence method is useful to specify users' information need, but its suggestion is limited. The extracted method we proposed here could extend users' understanding and cognition of the underlying data sources.

In another case, we test author name as target term to find similar authors using different approaches. The existing co-occurrence methods can only find the collaborators of a certain author as the similar terms of it. For example, they can find the collaborators of "Jiawei Han", such as "Philip S. Yu" and "Jian Pei". However, our approach can also find other well known researchers in Data Mining area, e.g., "Christos Faloutsos" and "Rakesh Agrawal", though they are not the collaborator of "Jiawei Han". We find that, these indirect researchers are connected through conference and shared research topics.

The above two case studies show the effectiveness of the proposed similarity measurement. In this work, we designed a heterogeneous TAT graph structure to model the structured data, which semantically integrates tuple and term relationship in a uniform way. Furthermore, the proposed random walk enhancement based on the TAT graph can take the contextual information into effect, i.e., exploiting global correlations between two nodes and also their multi-facet relations. Therefore, the proposed similarity measurement can better capture the semantic relation between terms in the structure data. Since no ground truth of term similarity is available, we leave the quantitative analysis to the next experimental evaluation, i.e., how the above two similarity measurements affect the performance of query reformulation.

B. Effect on Query Reformulation

We next evaluate the performance of query reformulation on structured data. In this work, we design a probabilistic generation model to reformulate input query and suggest new substitutive queries, which can incorporate both similarity and closeness metrics in an HMM model. We notate the proposed query reformulation method as *TAT-based Reformulation* in

[2]http://lucene.apache.org

the following experiment. We choose two baseline methods to compare with this proposed approach.

- **Rank-based reformulation**: Given each term's similar term list, we enumerate the possible combinations of corresponding terms, and return the queries with top similarity scores with original query. We use this baseline to demonstrate the contribution of probabilistic query reformulation module in our approach.
- **Co-occurrence reformulation**: We utilize the proposed query reformulation framework, but replace the random walk based similarity with the co-occurrence based similarity [15]. This variant can be used to demonstrate the contribution of new similarity metric in our solution.

The test query set has 10 different queries which are chosen with various formats consisting of topical words, author or conference name, such as "knn uncertain" and "Christian S. Jensen spatio-temporal".

A ranked list of reformulated queries from each method is returned for every input query. Due to the absence of golden standard set, we use *Precision@N* to measure the performance of different algorithms, which is the percentage of top-N answers returned that are relevant. Here, the relevance refers to the similarity and semantic closeness of reformulated ones with respect to the input query. In the experimental evaluation, we fetch the top-N reformulated queries with the highest similarity, and ask three evaluators to judge the relevance between original query and reformulated queries.

1) Quality of Reformulated Queries: We first compare the effectiveness of different methods, and the results are shown in Figure 5. We present the average precision value of testing queries at rank position 1, 3, 5, 7 and 10. The figure shows that our approach has advantages over two baseline methods. Co-occurrence similarity based reformulation method could not compete with others. As it has same query reformulation model with our method, we could conclude that the similar terms extracted with frequent co-occurrence method may not be suitable for query reformulation.

As we discussed previously, the co-occurrence method generally finds the terms which co-occur frequently in a certain field of structured data, thus top scored similar terms are locally related. With the help of contextual biased random walk, we could utilize context information for similarity evaluation over underlying targeting corpora, and hence the reformulated queries generated from the corresponding similar term lists of query keywords have higher probability to produce valid query. Note that, the keyword query results are generally retrieved from joined tuples in multiple relations. In Section VI-A, we have presented the case study on the effect of similarity difference between co-occurrence based method and our proposed method. The query reformulation results further verify the effectiveness of the TAT graph based similarity measurement.

Compared with the greedy style Rank-based reformulation baseline, our proposed approach also performs better. The rank-based method simply combines the similar terms from the corresponding lists and ignores the semantical correlation over underlying structured data. Our probabilistic method takes structural information into consideration to evaluate the reformulated query quality. By combining similarity and closeness in a unified way, our approach can yield more satisfactory performance.

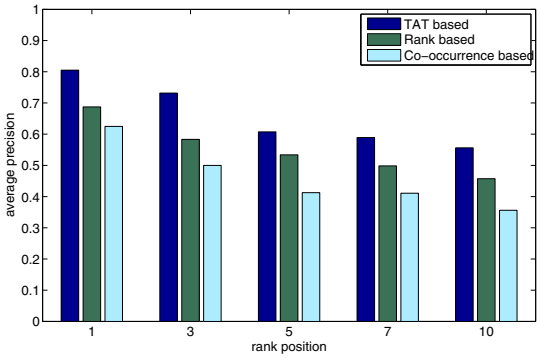


Fig. 5. Query Generation Performance of Different Methods

We next provide a showing case of reformulated queries. In Figure 6, we show a snapshot of the query reformulation demo interface. In the main column, traditional keyword search results are returned. In the right panel, some ranked reformulated queries are presented. The *initial* query "spatio temporal Christian S. Jensen" gets some *reformulated* suggestions: "moving object Ouri Wolfson" and "nearest neighbors Dimitris Papadias", which can not be generated by traditional query suggestion techniques, e.g., [15], [24].

Admittedly, there also exist some vague suggestions as we exploit larger scope of structured data for query reformulation. However, comparing the reformulated query with the returned paper list, we could find that these query suggestions are novel and diverse, beyond the returned papers and initial input query. Such query suggestions can help users better understand and explore the data source.

Fig. 6. Reformulated Keyword Queries

2) Time Efficiency: The query reformulation process requires fast response time. In Section V, we have discussed two algorithms in the probabilistic framework to support reformu-

lated query generation. Here we investigate their efficiency difference and deployment issues. In this experiment, we randomly sample 400 queries, varying query length from 1 to 8. These queries are chosen from the following fields: author name, paper title and conference name.

Figure 7 shows the run time of two algorithms. Algorithm 2 only uses k to prune the hidden state space and clearly fails to compete with the improved Algorithm 3. Algorithm 3 first uses a Viterbi to get reformulated query candidates and backward search to select top scored ones.

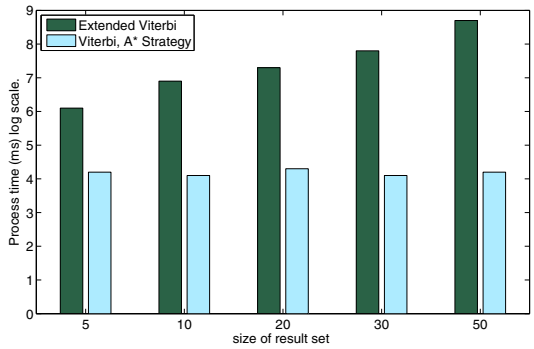


Fig. 7. Time Cost of Query Generation Algorithms

In the improved Algorithm 3, there are two stages, i.e., Viterbi based initialization and an A* backward search. Figure 8 illustrates the average response time in these two stages. The costs of both steps grow with the length of query, and among which the Viterbi initialization is a more costly procedure. When the length of query reaches 8 which can be considered as long query in real-life applications, the whole algorithm still yields satisfactory performance, which takes less than 0.2 second.

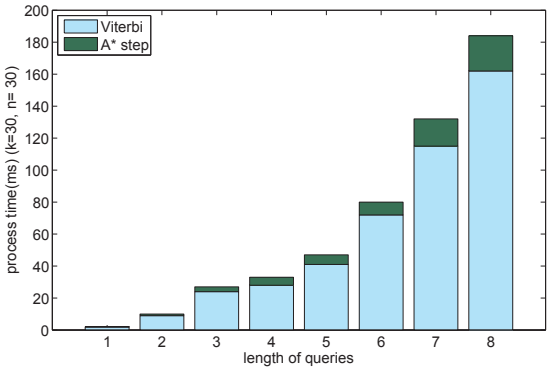


Fig. 8. Time Cost of new Top-k Query Generation Algorithm

3) Parameter Sensitivity: To show the robustness of query generation model, we further test two different parameters: the return size of top reformulated queries and the space of related hidden states in HMM model. These two parameters determine the search space for our generation model. The following two experiments use the improved A* solution as outlined in Algorithm 3.

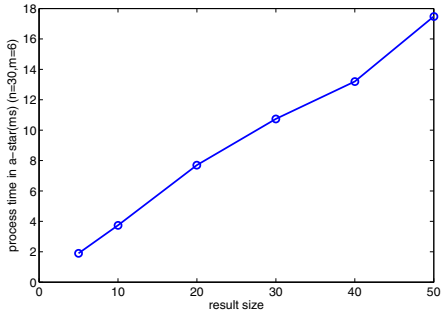


Fig. 9. Time Cost with Different Returned Queries

Figure 9 displays the average time cost, with the growth of number of queries recommended (k), when query length is set to 6, which is a relative long query as here we only extract topical words. Since the Viterbi step only generates the highest scored query, which is not affected by k larger than 1. We could find that, the time cost in A* search strategy stage grows linearly with k in Figure 9, which demonstrates the scalability in terms of the result size.

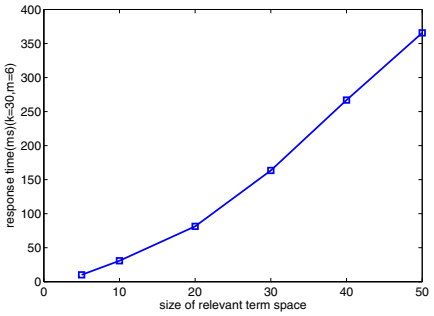


Fig. 10. Time Cost with Varied Sizes of Candidate States

Figure 10 presents the average response time by varying size of the hidden states in our proposed approach. In other words, in the online interactive query reformulation, how many similar terms for each input term can we fetch to ensure a fast response? The result shows that our approach is very efficient. Especially when the size of similar term list is less than 20. It is sufficient for most real interactive applications, such as Ajax or dialogue based query interface.

C. Effect on Reformulated Query Results

In the last experiment, we go a step further to test the keyword search results using the reformulated queries. The results can demonstrate how the reformulation can affect the search performance. The query set used for evaluation is the keywords extracted from the title of 19 SIGMOD Best Papers since 1994, and there are totally 19 queries in our test set. For each query, we get top-10 reformulated queries using different methods, conduct search using these reformulated keywords,

and finally examine the search results. We design two objective metrics to evaluate the performance:

- **Result size**: which is the average number of search results for the top-10 reformulated keywords of 19 queries. The larger size means the approach tends to generate higher quality reformulated queries, vise versa.
- **Query distance**: which is defined as the average term distance of every corresponding term pairs between two queries, where term distance is the shortest path distance of two terms in the TAT graph. In this experiment, we calculate the average query distances between top-10 reformulated queries and the original query. The distance reflects the diversity of the reformulated queries.

TABLE III
EFFECT ON QUERY REFORMULATION

	TAT based	Rank based	Co-occurrence based
Result size	20.89	9.21	14.16
Query distance	1.11	0.67	0.82

The results in Table III clearly show that our reformulation approach can produce reformulated queries with not only better quality but also higher diversity. This experiment further verifies the effectiveness of our TAT-based approach which can better exploit the semantics of underlying structured data.

VII. CONCLUSION

Information queries for text, multimedia and location are pervasive in current web search engines, and the availability of structured data continues to grow at an increasing speed. In this paper, we have addressed an interesting problem: how to automatically reformulate user's initial keyword query into similar or related ones, for better query understanding or recommendation.

Towards this goal, we proposed to model the structured data with a heterogenous graph and designed effective approaches for keyword query reformulation. Specifically, we presented a random walk based similar term extraction method, and utilized a probabilistic generation model for query reformulation. The experimental results show that our methods make a promising attempt towards this direction.

Compared to traditional query suggestion on text documents, keyword query reformulation on structured data has many open problems, which will be investigated in our future work. For example, a quantitative measurement of the effectiveness on term relation extraction and query generation is an ongoing work. With the collection of considerable query logs, the user interaction and feedback analysis on this new kind of query reformulation is another interesting extension. We could also exploit the reformulated queries to support ad hoc faceted retrieval over structured data, which is more intuitive and user friendly.

ACKNOWLEDGMENT

This research was supported by the grants of Natural Science Foundation of China (No. 61073019 and 60933004), and HGJ Grant No. 2011ZX01042-001-001.

REFERENCES

[1] R. MacManus, *Top 5 Web Trends of 2009: Structured Data*, http://www.readwriteweb.com/.
[2] R. Kumar and A. Tomkins, "A characterization of online search behavior," *IEEE Data Eng. Bull.*, vol. 32, no. 2, pp. 3–11, 2009.
[3] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: exploring the power of tables on the web," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 538–549, 2008.
[4] R. Gupta and S. Sarawagi, "Answering table augmentation queries from unstructured lists on the web," *PVLDB*, vol. 2, no. 1, pp. 289–300, 2009.
[5] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword search on structured and semi-structured data," in *Proc. of SIGMOD*, 2009, pp. 1005–1010.
[6] Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: top-k keyword query in relational databases," in *Proc. of SIGMOD*, 2007, pp. 115–126.
[7] S. Paparizos, A. Ntoulas, J. Shafer, and R. Agrawal, "Answering web queries using structured data sources," in *Proc. of SIGMOD*, 2009, pp. 1127–1129.
[8] N. Sarkas, S. Paparizos, and P. Tsaparas, "Structured annotations of web queries," in *Proc. of SIGMOD*, 2010, pp. 771–782.
[9] M. A. Hearst, *Search User Interface*. Cambridge Univ. Press, 2009, http://searchuserinterfaces.com/.
[10] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proc. of WWW*, 2006, pp. 387–396.
[11] I. Antonellis, H. G. Molina, and C. C. Chang, "Simrank++: query rewriting through link analysis of the click graph," *Proc. of VLDB*, pp. 408–421, 2008.
[12] D. Kelly, K. Gyllstrom, and E. W. Bailey, "A comparison of query and term suggestion features for interactive searching," in *Proc. of SIGIR*, 2009, pp. 371–378.
[13] E. Agichtein and L. Gravano, "Querying text databases for efficient information extraction," in *ICDE*, 2003, pp. 113–124.
[14] J.-H. Kang, M.-G. Kang, and C.-S. Kim, "Supporting databases in virtual documents," in *Proc. of ICADL*, 2001.
[15] Y. Tao and J. X. Yu, "Finding frequent co-occurring terms in relational keyword search," in *Proc. of EDBT*, 2009, pp. 839–850.
[16] N. Sarkas, N. Bansal, G. Das, and N. Koudas, "Measure-driven Keyword-Query expansion," *PVLDB*, vol. 2, no. 1, pp. 121–132, 2009.
[17] Z. Liu, P. Sun, and Y. Chen, "Structured search result differentiation," *PVLDB*, vol. 2, no. 1, pp. 313–324, 2009.
[18] G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and searching web tables using entities, types and relationships," *PVLDB*, vol. 3, no. 1, pp. 1338–1347, 2010.
[19] J. Pound, I. F. Ilyas, and G. E. Weddell, "Expressive and flexible access to web-extracted data: a keyword-based structured query language," in *Proc. of SIGMOD*, 2010, pp. 423–434.
[20] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. Parag, and S. Sudarshan, "Banks: browsing and keyword searching in relational databases," in *Proc. of VLDB*, 2002, pp. 1083–1086.
[21] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Objectrank: authority-based keyword search in databases," in *Proc. of VLDB*, 2004, pp. 564–575.
[22] F. Liu, C. Yu, W. Meng, and A. Chowdhury, "Effective keyword search in relational databases," in *Proc. of SIGMOD*, 2006, pp. 563–574.
[23] K. Q. Pu and X. Yu, "Keyword query cleaning," *PVLDB*, vol. 1, no. 1, pp. 909–920, 2008.
[24] R. Varadarajan, V. Hristidis, and L. Raschid, "Explaining and reformulating authority flow queries," in *Proc. of ICDE*, 2008, pp. 883–892.
[25] X. Yang, C. M. Procopiuc, and D. Srivastava, "Summarizing relational databases," *PVLDB*, vol. 2, no. 1, pp. 634–645, 2009.
[26] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *PVLDB*, vol. 2, no. 1, pp. 718–729, 2009.
[27] H. Tong, C. Faloutsos, and J. Pan, "Fast random walk with restart and its applications," in *Proc. of ICDM*, 2006, pp. 613–622.
[28] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge Univ. Press, 2007.
[29] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proc. of WWW*, 2002, pp. 517–526.
[30] Q. H. Vu, B. C. Ooi, D. Papadias, and A. K. H. Tung, "A graph method for keyword-based selection of the top-K databases," in *Proc. of SIGMOD*, 2008, pp. 915–926.
[31] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proc. of the IEEE*, 1989, pp. 257–286.