doi:10.1093/comjnl/bxy056

A Model-based Keyword Search Approach for Detecting Top-k Effective Answers

Asieh Ghanbarpour and Hassan Naderi*

Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
**Corresponding author: naderi@iust.ac.ir*

Keyword search (KWS) has been known as an attractive query processor in retrieving information from various types of data which could be modeled as a graph. An answer in response to a keyword query is a set of cohesively connected structures which shows how the data containing query keywords are interconnected in the graph. Finding answers to a given query efficiently and ranking the retrieved answers in an effective way are still two challenging problems in KWS domain. In this paper, we first propose a novel scoring function to optimize the accuracy of ranking the answers. This function is defined based on a carefully designed model called SARM which is an integrated model of the content and structure of an answer. We then develop a two-level KWS approach to support the efficient retrieval of top-k answers to a given query. This approach is based on pruning the search space to concentrate the search on the promising regions. The efficiency of this approach is improved by estimating the boundary scores of the answers in the regions. Extensive experiments conducted on a standard evaluation framework with three real-world datasets confirm the efficiency and effectiveness of the proposed approach.

Keywords: keyword search; database; language models; top-k query processing

Received 23 May 2017; revised 16 March 2018; editorial decision 6 May 2018
Handling editor: Joemon Jose

1. INTRODUCTION

With an increase in the amount of data in real-world databases, extensive efforts have been made towards developing effective methods for retrieving data against information needs. In this regard, many structured query languages have been proposed to query various types of databases. Some of these languages include SQL (for relational databases), XQUERY and XML-QL (for XML databases), RQL, RDQL and SPARQL (for semantic search on RDF databases) and Gremlin and Cypher (for graph databases). Using these languages is effective if users have a comprehensive knowledge of database schemas and are familiar with complex formal logic representation. These requirements have formed serious obstacles to common users wishing to search databases [1]. In recent years, some efforts have been directed toward providing more user-friendly interfaces for non-professional users. Natural language question answering (NL-QA) and keyword search (KWS) are two significant approaches in this direction [2]. In NL-QA systems, a user enters his query as a natural language sentence such as 'Where is the capital of Germany?'. The same query in KWS systems is expressed as 'Capital of Germany'. Since a natural language sentence has a complete syntactic structure, it can provide more semantic information than keywords about what the user is searching for. In contrast, keyword queries are more concise and flexible [2]. On the other hand, an answer to a query in an NL-QA system is a short passage of a resource (or a snippet of a document) related to the user's need [3]. However, a KWS system is able to retrieve not only the answers related to a single resource but the answers which are inferred from the relations among several resources. For example, one of the answers to keyword query 'Germany, France and Poland' on Mondial database is the bordering relationships between these countries that is 'France and Poland are connected through Germany'. NL-QA systems are unable to retrieve such answers. The focus of the current paper is on the problem of KWS over general graphs. In these graphs, a node could represent any information unit such as a resource, an entity, or a document based on the modeled background

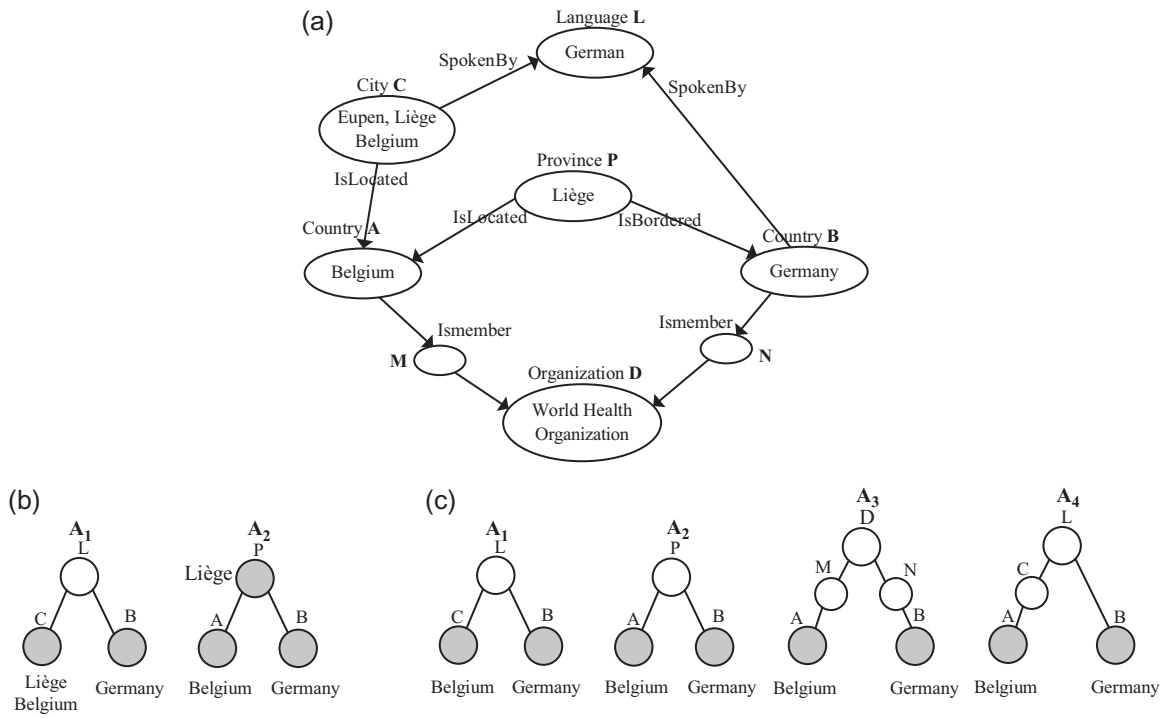

FIGURE 1. Some of the answers in response to given queries over a part of Mondial graph. (**a**) An example of Graph G. (**b**) Candidate answers to query. Q = {Liège, Belgium, Germany} over G. (**c**) Candidate answers to query Q = {Belgium, Germany} over G.

database. The edges of this graph show any homogeneous or heterogeneous relationship among the nodes. A keyword query is expressed as a set of keywords. A result to this query is a group of densely linked nodes in the graph, which covers all of the queried keywords [4]. As an example, consider a part of Mondial Graph[1] (with some modifications) shown in Fig. 1(a). Let query $Q = \{Li\grave{e}ge, Belgium, Germany\}$ be expressed over this graph. By ignoring the direction of edges, subgraphs A_1 and A_2 in Fig. 1(b) are two relevant answers to the query. Answer A_1 means 'German is the official language in Eupen city of Liège province in Belgium as it is in Germany' and answer A_2 means 'Belgium shares a border with Germany via Liège province'. Figure 1(c) also shows the relevant answers to query $Q = \{Belgium, Germany\}$.

It should be noted that KWS over semantic networks could be adapted to the semantic search [5–9]. It is reached by extracting a range of query semantics which the user has in mind when selecting the query's keywords. More precisely, a semantic query is equal to a query graph with constrained object nodes and property edges. Therefore, a keyword query could be transformed to a formal semantic query by constructing equivalent query graphs from keywords [1]. However, the domain of KWS is beyond the semantic networks and can be used on any type of graphs to detect the relationships among the nodes.

A major challenge for KWS over graph data is to develop an efficient algorithm to retrieve candidate answers in response to a given query. The efficiency of this algorithm mainly depends on the frequency of queried keywords in the graph and the catched ranking strategy [4]. For example, in [10], it is proved that if any answer is simply scored by the sum of edge weight, finding the top-ranked answer would be equivalent to the Steiner tree problem which is NP-hard. Due to the inherent difficulty of this problem, the use of heuristic strategies for solving it is justified.

The studies in the literature can be classified into four categories based on their heuristic search strategy in finding candidate answers: schema-based [11–14], exploration-based [15–18], Lawler-based [10, 19, 20] and virtual document-based (VD-based) methods [19, 21]. Schema-based methods employ the schema of the graph as a director map to guide the search toward goal regions. However, the schema of data is not always available or accessible. Exploration-based methods explore the search space by starting from the keyword nodes (the nodes containing queried keywords) and expanding them based on different expansion strategies until reaching common nodes. In the worst case, all of the graph may be searched using this approach. Lawler-based methods are based on dividing the search space into disjoint subspaces and distributing the search over them [10]. One of the main

[1] http://linkeddatacatalog.dws.informatik.uni-mannheim.de/dataset/mondial

contributions of these methods is presenting the answers in an exact ranked order during the search process [22]. But to this goal, the efficiency of the search has been sacrificed. In VD-based methods, the union of the text of a node and those of its neighborhood nodes are considered as a unit named as 'virtual document' [19]. The search on virtual documents is similar to that on the Web documents and it is done simply by detecting virtual documents which cover all or part of the queried keywords. Although structural closeness is considered in forming virtual documents (potential answers), the main contribution of VD-based methods relies on the textual aspects of answers which is insufficient in subgraph processing. On the other hand, since the size of the answers is restricted to the radius of virtual documents [19], the effectiveness of the corresponding search system is in the trade-off with the space used to store virtual documents. In this paper, we propose an exploration-based two-level KWS approach for efficient retrieval of a list of top-k answers to a given query. This approach is based on pruning the search space to concentrate the search on the promising regions in terms of answer inclusion. In this approach, a fast search is first executed on a summarized form of the graph to detect its promising regions. The successive search is executed on specified regions to retrieve the final answers. The proposed approach is further improved by estimating the score bounds of answers which could be found in a region. The estimated scores for answers provide the possibility of completing the set of candidate answers to be ranked before terminating the search.

The second challenge in KWS domain is to develop an effective and efficient method for ranking the retrieved candidate answers according to their relevance to a given query [4]. While, the relevance of an answer (subgraph) to a keyword query relies on the textual and structural properties of the answer. Most of the proposed ranking functions in the literature focused just on the structural properties of the answers. The number of nodes [11, 12], the distances between nodes [15, 16, 23], the distances between keyword nodes [24] and the distances between a root node and keyword nodes [10, 20, 25] are some of the factors used by these functions to rank answers. The absolute focusing of these functions on the structural properties of answers makes them unable to differentiate between the answers organized under the same structure (e.g. A_1, A_2 in Fig. 1 (b)). More recent KWS methods employed IR-style ranking factors to enhance the ranking quality. In most of these methods [19, 26, 27], the overall score of an answer was defined based on a linear combination of the textual (IR-based) and structural scores of the answer. However, interpreting the text of an answer separate from its structure decreases the distinction power of the induced function. For instance, consider answers A_1 and A_2 in Fig. 1(b). These answers cover the same set of keywords. Therefore, they receive the same textual score. In addition, by ignoring texts, the topological structures of the answers are similar. Therefore, the structural scores of the answers are also the same. In this case, any combination of textual and structural scores results in equal overall scores for the two answers. In this paper, we developed a novel ranking function based on an integrated model of content and structure of the answers to improve the accuracy of ranking. In the proposed model, any keyword is analyzed according to its structural position in the answer's graph. This idea was derived from positional language model (PLM) [28]. As a generation of probabilistic retrieval models, language models (LMs) have enjoyed much success in information retrieval systems due to their mathematical nature [29]. However, LMs (especially PLM) are basically designed for textual information and could not directly be applied to model subgraphs. In the current paper, a subgraph-specific model is proposed for KWS answers. These models are the basis of the proposed scoring function. The experimental evaluations on three real-world datasets show the effectiveness of the proposed scoring function in ranking the answers.

The main contributions of this paper are summarized as follows:

- We propose a novel two-level KWS approach which allows the effective pruning of the search space for efficient finding of a list of top-k relevant answers to a query.
- We develop an adapted version of graph partitioning and an approximate order search method to be used in the practical implementation of the proposed approach.
- We design three min-cost index structures to accelerate the retrieval of the data.
- To produce a more accurate order of answers to a given query, we develop a novel model-based ranking function. This function is defined based on an integrated modeling of the content and structure of the answers.
- We evaluate the efficiency and effectiveness of the proposed two-level KWS system by conducting expensive experiments on real-world sets of data to demonstrate the superiority of our system over the existing systems.

The rest of this paper is organized as follows. In Section 2, fundamental definitions along with the formal statement of the problem are provided. The model-based scoring function is discussed in Section 3. In Section 4, the two-level KWS system is discussed in detail. Experimental results are given in Section 5. The related works are discussed in Section 6, and the paper is concluded in Section 7.

2. PRELIMINARIES AND PROBLEM STATEMENT

Suppose a database which is modeled as a labeled graph $G = (N, E, T)$ with a set of nodes $N = \{n_1, \ldots, n_{|N|}\}$ and a set of edges E each connecting two nodes of N. For any node $n_d \in N$, there is a describing text. In this work, the same as [7, 15, 30], the text of each node is modeled as a bag of

words. The set of words in a node n_d is represented by $t(n_d)$ and the unique vocabulary of the graph is represented by T. The graph can be directed or undirected. In this work, similar as [10, 26, 31], we consider undirected graphs with weighted edges. These graphs are used to model different types of unstructured, semi-structured and structured data, such as web pages, XML documents and tabular data. It should be noted that the proposed approach is simply adaptable to work with directed graphs.

DEFINITION 1 (KWS problem). *Given graph G, an integer k as the desirable number of answers and a keyword query* $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ *in which any* q_i *shows a keyword, it is expected to find top-k relevant answers* $A_1, A_2, \ldots, A_k$ *to Q, whose are enumerated in a ranked order* [9].

The most basic and also the most common form of keyword queries consists of unordered lists of keywords connected by implicit conjunctions. A relevant answer to these queries is defined as follows.

DEFINITION 2 (Relevant answer to a query). *Given a labeled graph G and a query Q, a relevant answer* A_i *to query Q is a connected subgraph of G that satisfies the following conditions* [7]:

- (Coverage) for each keyword of the query, there is at least one node in the answer covering it.
- (Minimality) the answer does not contain any proper subgraph covering all of the queried keywords.

There may exist many relevant answers satisfying the above conditions for a large data graph. The observations show that the users usually examine only a few top answers (e.g. top-20). Therefore, to arrange the answers according to the user's needs, any answer is scored by a scoring function $Score(.)$ showing the relevance degree of the answer to the given query. The set of relevant answers is then ranked according to their relevance scores such that the answer with the highest score is ranked higher.

In what follows, we define some of the concepts which are used in the paper. The diameter of an answer is referred to the greatest shortest distance between any pair of the answer nodes. The eccentricity of a node n_d in an answer is defined as the greatest distance between n_d and any other nodes of N_A. The minimum eccentricity of all the nodes in N_A is named as the radius of the answer. The nodes of the graph containing a queried keyword are named as keyword nodes.

3. MODEL-BASED SCORING FUNCTION

The shortcoming of existing scoring functions motivates us to propose a new scoring function based on an integrated overview of content and structure of answers. For this purpose, we interpret a node of an answer using two models: The Internal Model (IM) which is estimated based on the internal content of node, and the Structure-Aware Relevance Model (SARM) which is estimated based on the IM parameters of the node, the IM parameters of its neighborhood nodes and the influence of the considered nodes on each other. Each node of an answer is scored based on its SARM. The score of an answer is calculated by an aggregation on the score on its nodes.

3.1. Internal model (IM)

If node n_d is considered as a document, its internal model, named as θ_d^{IM}, can be any well-known model such as multinomial, Bernoulli or Poisson [29]. We used unigram multinomial language model for the texts of nodes. However, any other language model could be adapted without any additional change in our method. In unigram multinomial language model, a text is considered as a sequence of words each generated independently. In this model, the sum of probabilities over all words of the text is assumed to be one. In the other words, the equation $\sum_{w_i \in n_d} P\left(w_i \mid \theta_d^{IM}\right) = 1$ is always maintained. With the assumption that n_d is a sample of θ_d^{IM}, the parameters of θ_d^{IM} are determined based on the Maximum Likelihood (ML) estimator as follows:

$$P(w_i \mid \theta_d^{IM}) = \frac{c(w_i, n_d)}{|n_d|} \tag{1}$$

where $c(w_i, n_d)$ shows the frequency of word w_i in n_d and $|n_d|$ (known as the node's size) represents the number of words in n_d.

3.2. Structure-Aware relevance model (SARM)

Structure-aware relevance model offers an environment modeling of a node to explain how information could be inferred from the node and its neighborhood.

DEFINITION 3 (r-neighborhood of node n_d in subgraph A). *The r-neighborhood of* n_d *comprises the set of nodes of A which lie at most at distance* δ *apart from node* n_d.

A structure-aware relevance model (SARM) is estimated for each node n_d of a candidate answer A based on its r-neighborhood in the answer. To preserve the locality of words, the nodes are contributed in the estimation of SARMs through their IM parameters. In the other words, SARM of node n_d is estimated over the union of IM parameters of node n_d and its r-neighborhood nodes. The influence of an IM parameter in the modeling of node n_d is proportional to the quality of the shortest path which connects the source node of the

parameter to node n_d. In this way, the parameters associated with the closer nodes have more contribution in the modeling of the center node than those associated with the farther nodes. Formally, Let $\widehat{n_d^r}$ shows the r-neighborhood of node n_d in answer A and N_d^r denotes the set of nodes in $\widehat{n_d^r}$ except n_d. Using the IM parameters of n_d and the IM parameters of its r-neighborhood nodes, the SARM of n_d is calculated as follows:

$$P(w_i \mid \theta_d^{SARM}) = \alpha \cdot P(w_i \mid \theta_d^{IM}) + (1 - \alpha) \cdot \sum_{n_h \in N_d^r} \left(\frac{Inf(n_d, n_h)}{\sum_{n_j \in N_d^r} Inf(n_d, n_j)} \times P(w_i \mid \theta_h^{IM}) \right) \quad (2)$$

where parameter α is a tuning parameter to control the impact ratio of internal information of a node relative to its neighborhood information. The function $Inf(n_d, n_h)$ shows the influence strength of node n_h on node n_d and vice versa (because the graph is undirected). The value of r (radius of the neighborhood) is set equal to the diameter of the answer. With this setting, the r-neighborhood of any node would cover all the other nodes of the answer. Therefore, each node will be modeled with the information of all the nodes contained within the answer. However, the value of r could be selected lower due to efficiency considerations.

Influence function: The influence function in Equation (2) is used to determine the impact of two nodes on each other. The aim of using this function is involving the structural properties of answers as well as the relative importance of intermediate nodes in modeling. We define the influence of two nodes on each other inversely proportional to the damping of the shortest path connecting the two nodes. Using the Gaussian kernel, the influence of node n_i and n_j on each other is calculated as follows:

$$Inf(n_i, n_j) = \exp\left[\frac{-(D(P_{n_i \rightsquigarrow n_j}))^2}{2\sigma^2} \right] \quad (3)$$

where parameter σ is a tuning parameter, similar to that used in Gaussian kernel.

The damping of a path is defined in positive relation with the weight of the path and in negative relationship with the importance of the intermediate nodes on the path. It means that a higher weight path between nodes tends to be dampened more heavily, resulting in the lower influence of endpoint nodes on each other. On the other hand, the more important intermediate nodes lead to moderate the path's damping, strengthening the influence between nodes. We define the damping of a path between two nodes n_i and n_j, noted by $D(P_{n_i \rightsquigarrow n_j})$, as follows:

$$D(P_{n_i \rightsquigarrow n_j}) = \sum_{e_k \in P_{n_i \rightsquigarrow n_j}} w(e_k) + (1 - \prod_{n_k \in P_{n_i \rightsquigarrow n_j}} I(n_k)) \quad (4)$$

The first expression in Equation (4) shows the sum on the weight of edges located on the path from n_i to n_j. The weights of edges can be determined by domain experts or defined as the inverse of the edge weights assigned by BANKS [15] and CI-Rank [30]. However, we simply set the weights of all edges equal to one. In this way, the weight of a path shows the distance between its end nodes. The expression $I(n_k)$ denotes the importance of node n_k. We set the importance of a graph node to be equal to its normalized PageRank value in the range of (0,1] [32]. PageRank values are calculated in an offline fashion independent of the query.

3.3. Smoothing SARM

In order to prevent assigning zero probabilities to unseen words, SARMs should be smoothed. It has been shown that smoothing improves the accuracy of the estimated relevance models [33]. JM-smoothing is a simple smoothing model, which interpolates the parameters of the estimated model with the parameters of collection language model through a fixed coefficient λ to control the amount of smoothing. A collection language model is referred to the estimated model over the contents of all nodes in the graph. If collection language model is denoted by θ_C, the smoothed version of SARM based on JM-smoothing is as follows:

$$P_{\text{Smoothed}}(w_i \mid \theta_d^{SARM}) = (1 - \lambda) P(w_i \mid \theta_d^{SARM}) + \lambda P(w_i \mid \theta_C) \quad (5)$$

3.4. Using SARMs in answer scoring

According to the base idea of the query-likelihood model, we assume a query is a random sample drawn from a structure-aware relevance model. It means that a user focuses on a node of an answer and express his/her query based on this node and its neighborhood, aiming at retrieving the answer. Suppose a candidate answer A in response to query Q, consisting nodes $n_1, \ldots, n_{|N_A|}$. In addition, suppose θ_d^{SARM} shows the estimated SARM for node n_d. The probability that a query Q is generated by θ_d^{SARM} is calculated as follows:

$$P(Q \mid \theta_d^{SARM}) = \prod_{q_i \in Q} P_{Smoothed}(q_i \mid \theta_d^{SARM})^{c(q_i, Q)} \quad (6)$$

where $c(q_i, Q)$ shows the frequency of keyword q_i in query Q. In calculating the probability $P\left(Q \mid \theta_d^{SARM}\right)$, it is assumed that each keyword q_i in the query Q is conditionally independent given θ_d^{SARM}. The estimated probability is used to score nodes of the answer as it is formulated in Equation (7). Using the logarithmic form of $P\left(Q \mid \theta_d^{SARM}\right)$ is to prevent a very small score:

$$\text{Score}(n_d, A, Q) = \log P(Q \mid \theta_d^{SARM}) = \sum_{q_i \in Q} c(q_i, Q) \log P_{\text{Smoothed}}(q_i \mid \theta_d^{SARM}) \quad (7)$$

Since the model of an answer's node depicts the textual and structural properties of the answer from the perspective of the node, combining the extracted scores from the SARMs of all the answer's nodes reflects an overall view of the answer. Based on this intuition, we define the score of an answer as a combination of the scores of its nodes. Using the sum-based strategy, the score of answer A retrieved in response to query Q is calculated as follows:

$$Score_s(A, Q) = \frac{1}{|N_A|} \sum_{n_d \in N_A} Score(n_d, A, Q) \quad (8)$$

where $|N_A|$ shows the number of nodes in answer A.

In the case where the parameter r (the radius of the neighborhood) is considered equal or greater than the radius of the answer, the information of all the answer nodes would be contributed in estimating the SARM of the center node. In this case, the score of the answer could be approximated by the score of its center node as follows (the center-based strategy):

$$Score_c(A, Q) = Score(cn_A, A, Q) \quad (9)$$

where cn_A denotes the center node of answer A. The recent strategy is more efficient because it confines the scoring calculation of a subgraph (answer) to that of a single node.

3.5. An example

In the following, the answers shown in Fig. 1(b) which are retrieved in response to query $Q = \{Belgium, Germany, Liege\}$ are scored using two mentioned strategies. We set $\alpha = 0.7$, $\sigma = 2$ and $r = 1$ and suppose that the importance of all nodes is equal to 0.8 (it is calculated using PageRank). The damping of path $P_{B \rightsquigarrow C}$ is calculated as follows:

$$D(P_{B \rightsquigarrow C}) = 1 + [(1 - (0.8)^3] = 1.488$$

Similarly, the damping of path $P_{L \rightsquigarrow C}$ is calculated equal to 0.136. Based on these values, the structure-aware relevance probability of word '*Germany*' in node C of answer A_1 is calculated as follows:

$$P(Germany|\theta_C^{SARM}) = 0.7 \times 0 + 0.3 \times \frac{e^{\left[\frac{-(1.488)^2}{8}\right]}}{e^{\left[\frac{-(1.488)^2}{8}\right]} + e^{\left[\frac{-(1.36)^2}{8}\right]}} = 0.1466$$

In the same manner, the structure-aware probabilities of words '*Liege*' and '*Belgium*' in node C of answer A_1 are calculated as 0.1534 and 0.7, respectively. The score of node C is calculated as

$$S(C, A_1, Q) = \log(0.1466) + \log(0.1534) + \log(0.7) = -1.8029$$

The scores of nodes B and L of answer A_1 are also calculated as -1.8029 and -1.8027, respectively. Therefore, the overall score of A_1 using the sum-based strategy is obtained as follows:

$$S(A_1, Q) = \frac{1}{3}[(-1.8027) + 2 \times (-1.8029)] = 1.8028$$

The score of answer A_2 is calculated as -1.8126 in the same manner. Based on the results of calculations, the answers A_1 is ranked higher than the answer A_2 using the sum-based strategy. This priority is also maintained using the center-based strategy.

4. PROPOSED KWS SYSTEM

In this section, we present an efficient two-level KWS approach (TKS) to retrieve a list of top-k answers to a given query ordered according to the model-based scoring function. In this approach, a summarized view of the graph (named as coarse-grained graph) is first searched to detect the promising regions of the graph. The retrieved results are presented as a set of super-answers which each contains all the queried keywords. A successive exact search (fine-grained search) is then performed on the regions specified by super-answers to retrieve the final answers. The idea of this approach is to prevent the exponential growth of execution time with increasing the level of exploration when using the exploration-based strategy (which is the most common strategy in the KWS over schema-free graphs). In the proposed approach, the growth of execution time is controlled by limiting the levels of exploration in the first phase of the search. Although this time is added by the time of search over a limited number of nodes in the second phase, it still remains significantly lower than the execution time of the base approach as it will be shown empirically in Section 5.

To supply a coarse-grained form of the graph, a partitioning is performed over the graph nodes. The detected partitions (blocks) are considered as the nodes of the coarse-grained graph and the connectivity between the blocks are considered as the edges. Two blocks are connected if there is at least one direct edge connecting their nodes. In what follows, we would first describe a partitioning method to construct the coarse-grained graph, then we would explain the way of search in each level of the proposed approach, and finally, we would improve the efficiency of the query processor by applying more pruning on the data.

4.1. Partitioning the Graph

Since the utility of graph partitioning in the proposed approach is pruning the search space, the way of partitioning would cause a significant impact on the search performance. Some of the expectations of a partitioning consistent with our search approach are (1) preserving the locality of nodes in the blocks, (2) restricting the radius of blocks to control the boundaries of the search and (3) placing a star node and its neighbors in multiple blocks to prevent an extensive tracking of outgoing paths after meeting a star node. We define a star node as a bridge node connecting a high number of its neighbor nodes. By putting the neighbors of a star node in different blocks, the search would be just followed in the targeted blocks after meeting the star node.

The mentioned objectives are not completely compatible with those of the most existing partitioning methods. Therefore, in this work, a modified version of BFS-based approach is developed to partition the graph in accordance with the requirements above. In the proposed approach, the problem with star nodes is handled by considering any of these nodes as a single-node block. The other nodes are partitioned as follows. Suppose the maximum radius of expected blocks is represented by ε. A node would be promoted to a seed node if it is not already assigned to a block as a member, and if it is located at least 2ε farther from the preselected seed nodes. The latter condition could be ignored if there is no unassigned node satisfying it. To identify a new block, a seed node is selected and a breadth-first search (BFS) is performed around it until ε levels. Any visited node during this search which has not been previously assigned to any block is added to the new block. The procedure of selecting a seed node and expanding is continued until no unassigned node remains.

LEMMA 1 (Satisfaction of the partitioning method). *Suppose that the set of blocks detected by the proposed partitioning method is represented by $\mathbb{B}$. Blocks in $\mathbb{B}$ are node-disjoint and their radiuses are not larger than ε. In addition, the union of all blocks in $\mathbb{B}$ covers the entire graph G.*

Proof. According to the proposed method, any node is only once assigned to a block. Therefore, the blocks are node-disjoint. On the other hand, the expansion of seed nodes to build blocks is continued until ε levels. Therefore, the radiuses of blocks could not exceed ε. Continuing partitioning until the assignment of all nodes guarantees that the set of blocks covers the graph.

4.2. Indexing Schemas

Index structures are used to accelerate search on the graph. We used three indexing schemas in our work.

Keyword indexing: Let $N(k_i)$ shows the set of nodes of N which contain k_i and $\mathbb{B}(k_i)$ denotes the set of blocks which cover k_i. In keyword indexing, any unique keyword k_i of T is mapped to two lists. The first list contains the nodes of $N(k_i)$. In this list, any node $n_d \in N(k_i)$ is stored along with the internal probability of k_i in n_d as $(n_d, P(k_i|\theta_d^{ILM}))$. The second list stores the blocks of $\mathbb{B}(k_i)$ which each is accommodated with the internal probability of k_i. Since multiple nodes in a block may contain k_i, the maximum internal probability of k_i in the block nodes is considered for the block. Formally, each block b_j is indexed for keyword k_i as $(b_j, max_{n_d \in b_j} P(k_i|\theta_d^{ILM}))$. The entries of both lists are sorted in a non-increasing order of their probability values. Keyword indexing schema is used to find the nodes and blocks of the graph containing the queried keywords to initiate the search process. The space complexity of this index is related to the frequency of keywords in the graph nodes.

Node indexing: In this schema, any block b_i is mapped to a list of nodes associated to b_i. As the blocks are node-disjoint, any node is indexed once. Therefore, the space complexity of the node index is $O(|N|)$, where $|N|$ is the number of graph nodes.

Summarized distance indexing: In naïve distance indexing, the shortest path between any two nodes of the graph, $\delta(n_i, n_j)$, is stored. This information leads to a significant reduction in run-time. However, the main problem of the naive indexing schema is its space complexity $O(|N|^2)$ which is too big even for graphs of moderate sizes. In the summarized version, for any node n_d of the graph, the shortest distances of n_d to the center node of its block $(\mathbb{B}(n_d))$ and the center nodes of adjacent blocks to $\mathbb{B}(n_d)$ is stored. This information is calculated once in a preprocessing step by performing a BFS starting at each block's center until at most $(2\varepsilon + 1)$ levels. Let the average branching factor of the coarse-grained graph is shown by bf. The space complexity of summarized distance indexing is from $O(bf \times |N|)$. Since the branching factor is logically a small constant value because of observing locality constraint, summarized distance indexing schema dramatically reduces the space requirement while it is significantly useful in approximating distances among the graph nodes.

4.3. KWS on the coarse-grained graph

An abstract view of the partitioned graph is called coarse-grained graph (CG) with the blocks as its nodes. The connectivity between the blocks specifies the edge set of CG. In the proposed approach, the first level of KWS is executed on the coarse-grained graph to detect the candidate super-answers. Any super-answer (a subgraph of blocks) specifies a region of the base graph containing at least one answer. We denote the blocks covering at least one queried keyword as the source blocks. The basis of KWS over coarse-grained graph is on spreading information from the source blocks to their neighborhood in a breadth-first manner until some blocks of the graph receive all the information needed to cover the query. The way of search over a coarse-grained graph is shown in Algorithm 1. In this algorithm, CG is a coarse-grained graph with $|\mathbb{B}|$ blocks. The parameter ξ shows the radius of expansion. It is set to $\frac{\delta}{\varepsilon}$ in the experiments where δ shows the maximum radius of the expected answers. It should be noted that the blocks of star nodes are not considered in determining the radius of expansion. In Line 1, an ordered list of source blocks is added into S_Q^1. Any other S_Q^t, $2 \leq t \leq \xi$, stores the Id of blocks which received information from the blocks of S_Q^{t-1}. In Lines 2–5 the single blocks covering all the queried keywords are detected and added to the list of super-answers. Lines 6–16 show an iterative expansion-based process to detect the other super-answers. In each iteration, any block $b_i \in S_Q^t$ spreads its information to its direct neighbors $\Gamma(b_i)$. The information is sent as a box named 'holding list'.

Algorithm 1 Keyword search over the coarse-grained graph.

Input: parameter ξ, coarse-grained graph CG
Output: a set of super-answers $\hat{\mathbb{A}}$
1. $S_Q^1 \leftarrow$ the set of source blocks of CG in descending order of their internal probabilities
2. **for** $b_i \in S_Q^1$ **do**
3. **if** $(Q \subseteq HL(b_i))$ **then**
4. $\hat{\mathbb{A}} \leftarrow$ build a subgraph having b_i as its single block
5. **delete** b_i from S_Q^1
6. **for** $1 \leq t < \xi$ **do**
7. **for** $b_i \in S_Q^t$ **do**
8. **for** $b_j \in \Gamma(b_i)$ **do**
9. **if** $(HL(b_i)$ *is not* ***NULL***$)$
10. **send** $HL(b_i)$ to b_j
11. $HL(b_j) \leftarrow HL(b_j) \cup HL(b_i)$
12. **if** $\left(Q \subseteq HL(b_j)\right)$ **then**
13. $\widehat{a_i} \leftarrow$ **Form** the super-answer by a backward search
14. **if** ($\widehat{a_i}$ is not replicate **and** it is minimal)
15. $\hat{\mathbb{A}} \leftarrow \widehat{a_i}$
16. **else** $S_Q^{t+1} \leftarrow b_j$
17. **return** $\hat{\mathbb{A}}$

The holding list of any block $b_i \in \mathbb{B}$, shown by $HL(b_i)$, comprises the queried keywords saved in b_i and those received from the neighbors of b_i $(\Gamma(b_i))$. Each keyword k_i in a holding list is stored along with the block Ids from which it has been received and its internal probability in these blocks. The holding list of any receiver node is updated by the new information (Line 11). An answer is detected when all the queried keywords are observed in the holding list of a block (Line 12). The subgraph of the detected answer is retrieved using the block Ids saved in the holding lists (Line 13). In Line 14, the found answer is checked for replication and minimality.

THEOREM. *If the information of source blocks is spread up to ξ levels, all the super-answers with the radius equal or lower than ξ will be detected by Algorithm 1.*

Proof. Since the super-answers are connected subgraphs, the information of source blocks of a super-answer would be received in its center node after at most ξ steps of spreading (the maximum radius of super-answers). Therefore, by spreading information from all of the source blocks up to ξ levels, the center nodes of all the expected super-answers would receive the information needed to cover the query. The related super-answers are detected in these nodes and formed by a backward search based on the stored information in the blocks.

4.4. KWS on the main graph

Any super-answer retrieved by the coarse-grained search specifies a region of the main graph which should be searched more precisely. However, if the fine-grained search is exactly confined to the blocks of candidate super-answers, some valuable answers may be ignored. These answers are subgraphs connecting the keyword nodes of source blocks by passing through unselected blocks. To clarify, consider the graph shown in Fig. 1(a). Let this graph is partitioned into three blocks $b_1 = \{B, N, D, M\}$, $b_2 = \{A, C, L\}$ and $b_3 = \{P\}$. Obviously, all of these blocks are directly connected to each other in the coarse-grained form of the graph. Consider $Q = \{Germany, Belgium\}$ is queried on the graph. Certainly, after a coarse-grained search, super-answer $\hat{a} = \{b_1, b_2\}$ would be retrieved for the given query. This super-answer is minimal and covers all the queried keywords. If the fine-grained search is confined to the blocks b_1 and b_2, answers A_3 and A_4 would be retrieved. However, there is another answer (A_2) which connect the same keyword nodes by passing through the unchecked block b_3. This answer is not detected even though it has a higher score than A_3 and A_4.

This problem is handled by considering a limited number of neighbor blocks of a super-answer in the fine-grained search. Formally, suppose $\mathbb{B}(\hat{a})$ shows the blocks of

super-answer $\hat{a}$. The set of blocks which are contributed in the fine-grained search of $\hat{a}$ is determined as follows:

$$BS(\hat{a}) = \mathbb{B}(\hat{a}) \cup \{b_k \mid \exists\ b_i\ ,\ b_j \in\ \mathbb{B}(\hat{a}) \text{ where } (b_k, b_i) \in E_{\mathbb{B}},\ (b_k, b_j) \in E_{\mathbb{B}}\} \quad (10)$$

where $E_{\mathbb{B}}$ shows the set of edges in the coarse-grained graph. It should be noted that even by this improvement, the comprehensive of our searching method is not guaranteed. It means that some duplicate answers may not be retrieved using the proposed method. Two answers are duplicate if they cover the same set of keyword nodes even though these answers may have different intermediate nodes connecting the keyword nodes [10].

The overall procedure of fine-grained search over the specified set of nodes is similar to Algorithm 1. However, there are some differences. The fine-grained search on a super-answer $\hat{a}$ is initiated from the keyword nodes associated to the blocks of $\mathbb{B}(\hat{a})$ as seed nodes. In addition, the expansion of each node is restricted to its neighbor nodes which are members of the $BS(\hat{a})$. Formally, each node n_d send its holding list to the node set $(\Gamma(n_d) \cap N(BS(\hat{a})))$, where $N(BS(\hat{a}))$ denotes the set of nodes which are associated to the blocks of $BS(\hat{a})$. The search process is continued until top-k answers are outputted, all the keyword nodes are extended until δ (the maximum radius of the expected answers) levels or all the super-answers are searched. It should be noted that any other KWS algorithm which able to confine the search to the specified regions of the graph could be used as the search engine in our framework.

4.5. Efficient interacting of the two levels of search

In the simplest form of interaction, the two levels of search could be executed sequentially on the graph as follows: first, all the super-answers are retrieved by searching over the coarse-grained graph, then the regions specified by the super-answers are searched in the fine-grained level to retrieve candidate answers. The set of all the candidate answers is finally ranked according to their scores to present top-k of them to the user. This way of search leads to a low response time of system because presenting answers would be delayed until completing the both levels of search. This approach is modified in this section to be executed with more degree of parallelism. The first level of search specifies the more promising regions (super-answers) of the graph in terms of answer inclusion. If an order could be defined among the super-answers such that the super-answers containing better answers are searched earlier, it would be possible to present top-k answers before the expansion of all the super-answers. For this purpose, an Upper bound Score (US) and a Lower bound Score (LS) are calculated for each super-answer. These scores show the maximum and minimum scores of answers which are included in the super-answer.

In the proposed approach, the list of candidate super-answers is first sorted in descending order of their USs. The super-answers are then expanded in an iterative manner. In each iteration, the top super-answer $\hat{a}_i$ is selected to expand. Suppose $\mathbb{A}$ shows the set of answers included in super-answer $\hat{a}_i$. The score of any answer of $\mathbb{A}$ is calculated using the proposed model-based scoring function discussed in Section 3. For any answer $a_j \in \mathbb{A}$, if its score is equal to the upper bound score of super-answer $\hat{a}_i$, it is directly sent to the output. Otherwise, this answer in along with its score is pushed to priority queue $\mathbb{Q}$ which its elements have been sorted in descending order of their scores. The priority queue $\mathbb{Q}$ is utilized to guarantee that the expected order of answers is maintained in the output. In this approach, before expanding the selected super-answer $\hat{a}_i$, the scores of current answers in $\mathbb{Q}$ are compared with the upper bound score of $\hat{a}_i\,(US(\hat{a}_i))$. The answers of $\mathbb{Q}$ whose score is equal or larger than $US(\hat{a}_i))$ have certainly the upper score than the unseen answers. Therefore, they are removed from $\mathbb{Q}$ and sent to the output. If the number of presented answers is not reached k, super-answer $\hat{a}_i$ is expanded and its fine-grained answers are pushed into $\mathbb{Q}$. Query processing can terminate safely when providing top-k answers or expanding all the candidate super-answers.

The proposed approach has also utilized an early pruning on the list of candidate super-answers. Accordingly, any super-answer $\hat{a}_i$ with $US(\hat{a}_i)$ lower than the LS of at least k super-answers is discarded and pruned before expanding. Formally, the pruned set is equal to

$$Pruned\_Set = \{\hat{a}_i \mid\ \forall\ \hat{a}_j \in k\_LS,\ US(\hat{a}_i) < LS(\hat{a}_j)\} \quad (11)$$

where k_LS shows the set of top-k super-answers in terms of LS. Using this pruning, none of final top-k answers would be lost, because even though any super-answer $\hat{a}_j \in\ k\_LS$ only produces one answer with the lowest score $\left(LS(\hat{a}_j)\right)$, the set of derived answers from the members of k_LS would be ranked higher than the answers extracted from any pruned super-answer. The overall view of processing a keyword query by the proposed approach is shown in Fig. 2.

4.5.1. *Calculate the upper bound and lower bound scores*

In this section, an upper bound score and a lower bound score are estimated for any answer contained in a super-answer using the summarized distance index discussed in Section 4.2.

THEOREM 1. *The lowest bound score of induced answers from super-answer $\hat{a}$, $LS(\hat{a})$, is equal to the score of $\hat{a}$ when the weight of each of its edge connecting blocks b_x and b_y is determined as follows*:

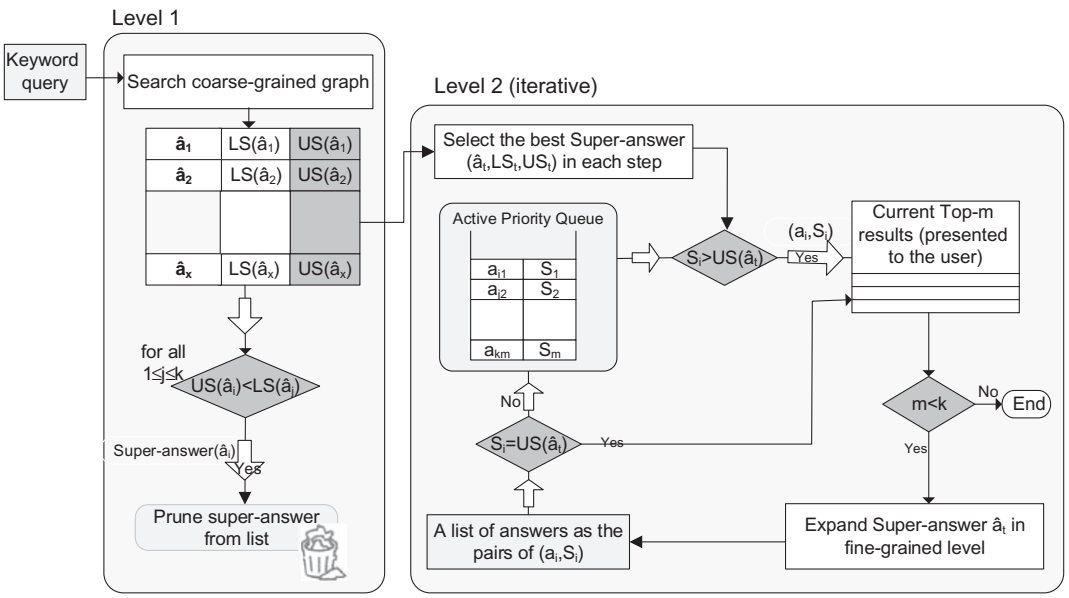

FIGURE 2. The overall procedure of proposed approach (the white arrows show processing of all the input entries).

$$\mathrm{LS}(\mathrm{b_x}, \mathrm{b_y})_{\mathrm{b_y}\in \mathrm{Neighbors}(\mathrm{b_x})} = \begin{cases} \min(f(b_x), f(b_y), f_t(b_x, b_y)) & \text{if } b_x \text{ and } b_j \text{ are both source blocks} \\ \min(f(b_x), f_t(b_x, b_y)) & \text{if just } b_x \text{ is a source block} \\ f_t(b_x, b_y) & \text{if none of blocks are source blocks} \end{cases} \quad (12)$$

where

$$f(b_x) = \max_{\substack{n_i\in\mathbb{N}_k(b_x)\\ n_j\in\mathbb{N}_k(b_y)}} (\delta(n_i, c(b_x)) + \delta(n_j, c(b_x)) \quad (13)$$

$$f_t(b_x, b_y) = \max_{\substack{n_i\in\mathbb{N}_k(b_x)\\ n_j\in\mathbb{N}_k(b_y)}} (\delta(n_i, c(b_x)) + \delta(n_j, c(b_y)) + \delta(c(b_x), c(b_y))), \quad (14)$$

$\mathbb{N}_k(b_x)$ shows the set of keyword nodes in block b_x of super-answer $\hat{a}$ and $c(b_x)$ denotes the center node of this block.

Proof. Suppose two neighbor blocks b_x and b_y in Fig. 3, each of them covering a keyword node n_i and n_j respectively. According to the information of summarized distance index, three ways could be imagined to relate keyword nodes of these blocks: connecting through the center node of block b_x (with the cost of $f(b_x)$), connecting through the center node of block b_y (with the cost of $f(b_y)$) and connecting through the center nodes of two blocks (with the cost of $f_t(b_x, b_y)$). The cost of the least expensive path among the center-based paths is considered as the weight of the edge connecting two blocks (Equation (12)). Since there may be more than one keyword node in a block, the maximum function is used in the cost calculations to take the farthest keyword nodes into consideration. Two other conditions in Equation (12) are the simpler forms of the first condition when one or both of the blocks are devoid of keywords.

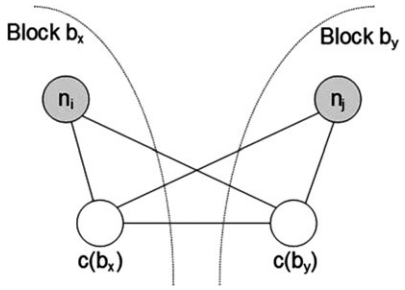

FIGURE 3. Connecting paths between two nodes $\boldsymbol{n_i}$ and $\boldsymbol{n_j}$ of adjacent blocks through center nodes.

THEOREM 2. *The upper bound score of induced answers from super-answer $\hat{a}$, $US(\hat{a})$, is equal to the score of $\hat{a}$ when the weights of all of its edges are set to one.*

Proof. The lowest score answer will be induced from a super-answer when each two neighbor blocks of the super-answer are connected through a lowest weight edge and the subgraph derived from these connections covers all the query

keywords. In this case, we have $\mathrm{US}(\mathrm{b_x}, \mathrm{b_y})_{\mathrm{b_y} \in \mathrm{Neighbors}(\mathrm{b_x})} = 1$ for all $b_x \in \mathbb{B}(\hat{a})$ and $b_y \in \mathbb{B}(\hat{a})$. It is noted that the nodes of unconnected blocks could not have the common edges.

4.6. An Improvement for more efficiency

The strength of pruning could be improved if the boundary scores of super-answers are determined more precisely. In calculating the US of a super-answer, it was assumed that two keyword nodes in neighbor blocks are directly connected. However, the connection of the nodes could be directly checked through the edge set. By knowing these connections, the weight of any edge between blocks b_x and b_y of super-answer $\hat{a}$ is calculated by Equation (15). The improved US of super-answer $\hat{a}$ is determined by updating the weight of its edges to those calculated by the following equatios:

$$\mathrm{US}(\mathrm{b_x}, \mathrm{b_y})_{\mathrm{b_y} \in \mathrm{Neighbors}(\mathrm{b_x})} = \begin{cases} 2 - \max\limits_{\substack{n_i \in \mathbb{N}_k(b_x), n_j \in \mathbb{N}_k(b_y) \\ t(n_i) \cap Q \neq t(n_j) \cap Q}} (e(n_i, n_j)) & \text{if } b_x \text{ and } b_y \text{ are both source blocks} \\ 1 & \text{otherwise} \end{cases} \quad (15)$$

The function $e(n_i, n_j)$ returns one if there exist a direct connection between nodes n_i and n_j, else it returns zero. Equation (15) shows that if there is no direct connection between the keyword nodes of two blocks, the length of their connecting path is at least 2. In the other cases, this value is set to 1.

5. EXPERIMENTS

5.1. Datasets and queries

The effectiveness of a retrieval system depends on how much the retrieved responses by the system addresses the information needs. Coffman and Weaver [34, 35] developed a benchmark evaluation framework consisting a set of queries and their relevant answers to test the effectiveness of KWS systems. This framework consists of three real-world datasets including Mondial, Wikipedia and IMDB (Internet Movie Database).[2] The information of each dataset is organized as some tables, which their tuples are connected to each other through some foreign keys. IMDB and Wikipedia datasets contain six tables with the tuples which relatively cover large blocks of text. While Mondial dataset contains 26 tables and the text of each of its tuples is restricted to some descriptive keywords. In addition, the density of relationships in Mondial dataset is higher than that in two other datasets. Fifty queries were designed for each dataset, each of them is accompanied by a list of accurate answers derived from querying on the related dataset with SQL statements. The queries were designed to explore the relationships among the entities of the dataset. For example, the query 'reeves wachowski' (Query #48 on IMDb) is designed to find relevant results identify collaborations (through the title relation) between Keanu Reeves and (either of) the Wachowski brothers. The average number of tuples in an answer per query is 4.49.

TABLE 1. Characteristics of the evaluation datasets.

Dataset	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert T \rvert$
Mondial	17 000	56 000	12 000
Wikipedia	206 000	785 000	750 000
IMDb	1 673 000	6 075 000	1 748 000

5.2. Systems implementation and evaluation

We modeled any dataset as a graph where the tuples act as its nodes. The relationships between tuples via foreign keys determine the edges between corresponding nodes. The nodes Id were used to build the keyword index on the corpus of words in the graph. Stemming and stop-word removal were applied to the text of nodes before indexing. Since the name of tables may also be queried by users, the name of the related table was also indexed for each node. Some information about the derived graphs is listed in Table 1.

The system is implemented in java using JgraphT library[3] as the same as the other systems which are used for comparisons. The indexes are constructed using Apache Lucene.[4] The experiments are run on a windows server with a 2.6 GHz Intel Core i7 processor and 12 GB of RAM. The proposed system is named as TKS-Sum when it uses the sum strategy to score answers and it is named as TKS-Center when the center strategy is used.

To measure the effectiveness of the systems, we use the same metrics considered in the evaluation framework [34] including Mean Average Precision (MAP) [36], Mean Reciprocal Rank (MRR) [37], Top-1 [34] and 11-points interpolated average precision [36]. Mean Average Precision (MAP) provides a single-figure measure of quality across recall levels. It has been shown that among the IR measures, the MAP has especially good discrimination and stability [36]. Average Precision (AP) for a query is calculated by averaging the precision values after retrieving each relevant answer (the precision of any relevant answers not retrieved is set to zero). Mean average precision is the average of APs across all queries. The 11-points interpolated average

[2]The dataset, queries, and relevance assessments are taken from http://www.cs.virginia.edu/~jmc7tp/projects/search/.

[3]http://www.jgrapht.org/

[4]http://lucene.apache.org/java/docs/index.html

precision is a precision-recall measure that show the precision of the system at 11 levels of recall varying from 0 to 1. The *interpolated precision* at a recall level r is defined as the highest precision which is found for any recall level $r' \geq r$. The mean reciprocal rank is a statistical measure for evaluating any system that produces a ranked retrieved list to every query. It cares about the highest-ranked relevant answer retrieved for each query. The Reciprocal Rank (RR) of a query is calculated by reversing the rank of the highest-ranked relevant answer retrieved by the system. MRR of the system is calculated by averaging the RR values over all queries. The Top-1 of a system is the number of queries for which the first answer is relevant. In the other words, this measure shows the sum of precisions of the system at the first retrieved answer for a set of $|Q|$ queries.

For any query which is a set of unordered keywords with implicit conjunctions, top-100 of the most specific results (those covering all of the keyword queries) are retrieved. The results are presented in comparison with the results of some state-of-the-art systems including BANKS [15], CD [24] and GraphLM [19] which are respectively selected from the families of expansion-based methods, text-based methods and language model-based methods. A summary of these methods is presented in Section 6.

5.3. Effectiveness

We first study the impact of parameters of the proposed ranking model, α and λ, on the effectiveness of the systems. Parameter α defined in Equation (2) is used to control the participation ratio of a node's internal parameters to its neighborhood parameters in estimating the related SARM. Figure 4 represents how the effectiveness of the systems are affected by the parameter α.

When $\alpha = 1$, it means that the SARM of any node is estimated just based on the internal content of the node. In contrast, when $\alpha = 0$, it means that the internal content of the node is ignored and the SARM is estimated just based on its neighborhood content. The diagrams in Fig. 4 show that none of these settings resulted in an effective system. In the other words, the systems are more effective when the modeling of nodes is performed based on their enriched content by the neighborhood information $(0 < \alpha < 1)$. The diagrams also show that the MAP of systems over Wikipedia dataset is more stable against the variation of α than those over the other datasets. It is because of the text-rich content of nodes and low structural variation of answers in Wikipedia dataset. Since a large number of answers in response to the queries of this dataset have the same structure, utilizing only the content of nodes could properly reflect the meaning of answers. The other point that is evident in the diagrams is more stability of TKS-Sum system rather than TKS-Center against the variation of α. In the TKS-Sum, independent of the α value, all of the answer's nodes participate in the answer scoring. While in the TKS-Center, the answer is scored based on its center node and the other nodes participate in the scoring as the neighbors of the center node. Therefore, the latter system is more sensitive to the participation percentage of neighborhood contents (α) in the modeling. Based on the MAPs of both the systems, we choose $\alpha = 0.4$ in the experiments since the systems have the best performance with it for all datasets.

The parameter λ was used in smoothing the models. Figure 5 illustrates how the effectiveness of the systems is affected by the parameter λ. According to the diagrams, the sensitivity of the systems to the variation of λ is not very significant over Mondial and Wikipedia datasets. However, the smoothing could improve the effectiveness of systems over IMDb dataset when λ is between 0.4 and 0.8. Therefore, we choose $\lambda = 0.8$ for the other experiments.

Figure 6(a) shows the comparisons of our systems with some state-of-the-art systems in terms of MAP. The results of the proposed systems are obtained with the tuned parameters $\alpha = 0.4$ and $\lambda = 0.8$.

According to the diagrams, for the queries over Mondial dataset, the effectiveness of CD, GraphLM, TKS-Sum and TKS-Center is higher than 0.8, all better than BANKS. For the queries over Wikipedia, the MAPs of BANKS, CD and GraphLM are much lower than the MAP of TKS-Sum (0.817). For the queries over IMDb dataset, unlike the other

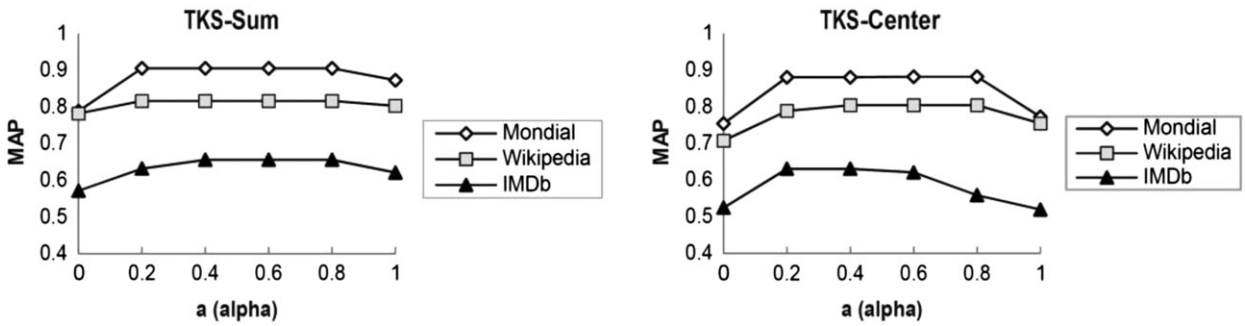

FIGURE 4. The effect of α on mean average precision for three datasets.

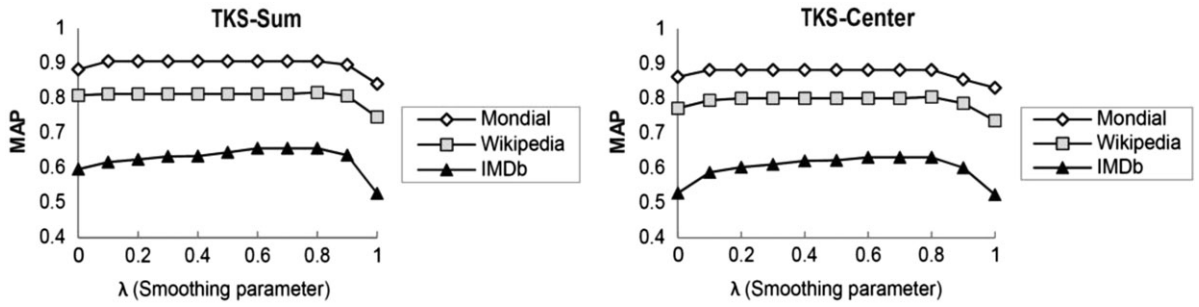

FIGURE 5. The effect of λ on mean average precision for three datasets.

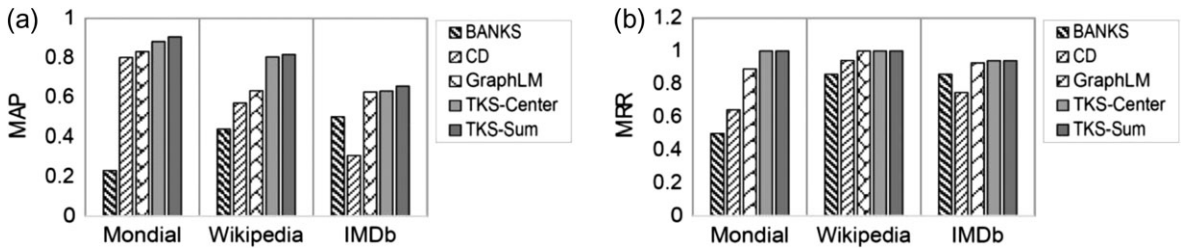

FIGURE 6. Comparison of the proposed systems with the state of the art systems in terms of effectiveness. (**a**) MAP values on all queries. (**b**) MRR values of queries with single-node.

two datasets, the effectiveness of CD is worse than that of the other systems. The reason is related to the high variety of entities with the same name in IMDb dataset. For example, there are 58 persons with the name of 'Abbott' in this dataset. In this case, the importance of nodes is an effective factor in the distinction of answers containing such nodes (entities). BANKS and TKS consider this factor through using the PageRank values of nodes, GraphLM uses the in-degree of nodes to contribute the importance of node, and CD ignores this factor in scoring answers. The superiority of the proposed systems in terms of effectiveness rather than the other systems shows the strength of the proposed scoring functions. GraphLM is the next best system in terms of MAP. It also uses a model-based scoring function. However, Modeling the text of answers separately from their structure and using a lower number of factors in scoring answers makes the effectiveness of GraphLM lower than those of the proposed systems.

Figure 6b shows the comparison of systems in terms of mean reciprocal rank over the queries of datasets which are answered by a single node. There are 20 queries in Mondial and IMDb datasets and 15 queries in Wikipedia dataset having this situation. Since the queries of this experiment are related to the answers with no structure (single-node answers), the effectiveness of the two proposed systems is identical. We get MRR = 1 on Mondial and Wikipedia datasets which means that the proposed methods could detect a relevant answer as the top-ranked one in all the examined queries of these datasets. On the IMDb dataset, TKS-Sum and TKS-Center achieve MRR = 0.9416 which is higher than the MRR of the other systems. The factor related to the importance of nodes plays a decisive role in this experiment. Through comparing the results of GraphLM and TKS systems, it could be concluded that using a more expert method such as PageRank in determining the importance of a node is more effective than using a local estimation of this factor.

The Top-1 and MRR of TKS systems and GraphLM system over all queries of datasets are compared in Fig. 7a and b. This experiment indicates that the overall MRR values of TKS-Sum are about 14%, 25.7% and 11.4% better than the MRR of GraphLM over Mondial, Wikipedia and IMDb datasets, respectively. It shows that the success of TKS-Sum in ranking the answers with more than one node is higher than that in ranking the single-node answers. This result shows the effective embedding of structural features in answer modeling of the proposed methods.

Figure 8 shows 11-point interpolated-precision curves of the examined KWS systems over a subset of 10 Trec-style queries (#36 and #45) of Wikipedia dataset. According to this figure, CD achieves the best precision in the first recall point. However, the effectiveness of this method drops quickly in the higher recall points. This behavior may be related to using the cover sets in scoring answers. A cover set shows the concentration of the related answer on the

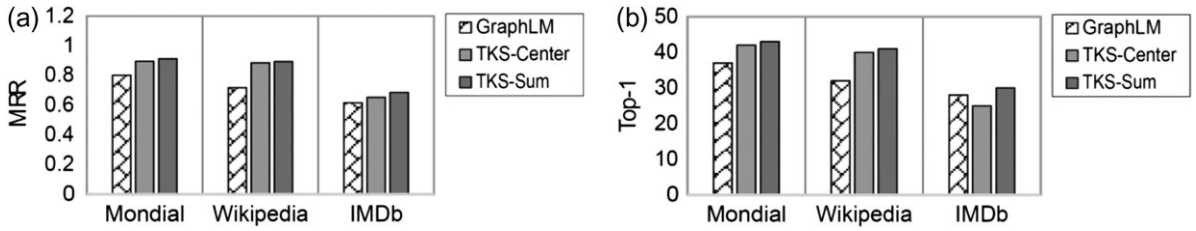

FIGURE 7. MRR and Top-1 values of the proposed systems in comparison to those of the GraphLM system. (**a**) MRR values (**b**) Top-1 values.

keywords of the query. However, since the use of structural features is limited in the answer scoring of CD, this approach could not be successful in the ranking of the other answers. The superiority of TKS-Sum in most of the recall points is evident in this experiment.

5.4. Efficiency

The efficiency comparison between TKS systems and the base KWS systems is shown in Fig. 9. In the base systems, the fine-grained search is directly executed on the main graph. This system is named as KS-Sum when it uses the sum-based strategy to score answers and it is named as KS-Center when it uses the center-based strategy. The mean execution times of examined systems to retrieve top-100 answers for 50 queries of each dataset are represented in Fig. 9. The radius of partitions for each dataset is set to the maximum radius of answers which are expected for the queries of the dataset. According to the figure, the TKS systems significantly outperform the base ones in terms of the execution time.

According to Fig. 9, the execution times of TKS-Sum and TKS-Center are very close to each other over all the datasets. It is because that the overall execution time of a KWS system is dominated by the search time of the system. However, the TKS systems are common in the search engine. Therefore, their execution times vary just as much as their ranking times.

Figure 9 also shows that the improvement of execution times on IMDb dataset is higher than those on Wikipedia and Mondial datasets. The reason is related to the density of the datasets. Although the mean branching factor in IMDb graph is low, it contains many star nodes which are in direct connections with more than fifty hundred nodes. The star nodes cause significant problems in the based expansion-based KWS method. This problem is handled in the proposed method by confining expansion from star nodes.

6. RELATED WORKS

Research on KWS has been conducted in four categories: KWS on XML datasets [38, 39], KWS on RDF datasets [17, 40, 41], KWS on relational datasets [12, 27, 42, 43] and

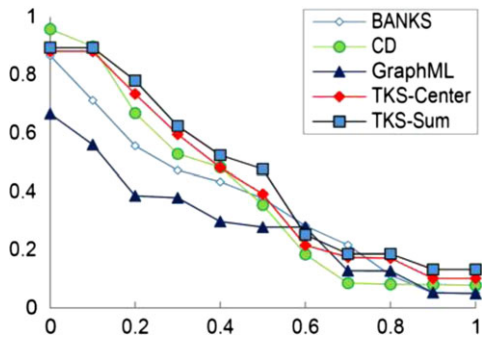

FIGURE 8. Comparison of different methods in terms of 11-point interpolated precision for 10 Trec-style queries (#36–#45) of Wikipedia.

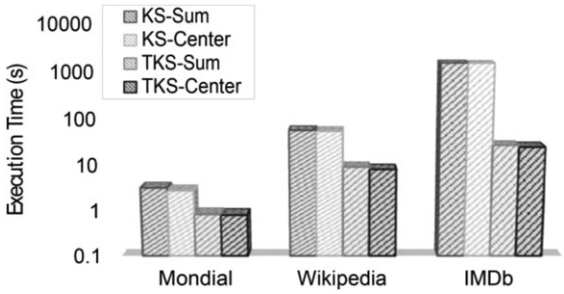

FIGURE 9. Mean execution times of TKS systems and the base systems.

KWS on schema-free graphs [8, 10, 12, 16, 18, 31, 44]. KWS over XML data is mainly used to retrieve meaningful snippets of documents which are relevant to the user's query [4]. An XML document is generally modeled as a labeled and directed tree. In this type of modeling, any element, attribute, and text value of the XML document is considered as a node of the tree. Over an XML tree, various LCA (Lowest Common Ancestor)-based semantics, such as MLCA, CVLCA, SLCA, ELCA and MaxMatch have been defined to form the answers [45]. Any of these semantics defines the position of a common ancestor identified by the semantics in descendant positions. Since all of the LCA-based methods are tree-based, they could not be efficiently applied to graph-structured data.

In the domain of KWS, RDF data are usually modeled as a node-labeled and edge-labeled directed graph [41] where the nodes represent resources or literals and the edges represent properties carrying heterogeneous semantics. In many existing works over RDF data, keyword queries were adapted to semantic search. In these works, a keyword query was translated into a set of relevant SPARQL queries. In SPARK [1], this translation is initiated by mapping the keywords to related resources. Any possible combination from different resources of each keyword is considered as a query set. Then, a query graph is constructed for each query set using Minimum Spanning Tree algorithm. Finally, each query graph is translated into a SPARQL query by using conversion rules. In [41], the RDF graph is searched at the type level using an index-based search method to find subtrees covering all of the keywords. The retrieved type-based subtrees are then translated into SPARQL queries. By executing any SPARQL query, some of the relevant answers to the query are retrieved. These answers are ranked based on the root-based semantic which scores each answer based on the sum of distances from any node to the answer's root node. The initial search in [46] is performed on a summary of the original graph containing structural (schema) elements only. The retrieved schema subgraphs are used to extract the corresponding SPARQL queries.

In the proposed systems for KWS over relational databases [14, 16, 47], a relational database is viewed as a graph with tuples as its nodes and foreign key references among tuples as its edges. Many approaches, however, use the schema of relational database to generate candidate answers and then translate them into SQL queries. These queries are executed on the database to get trees of tuples as final answers.

The studies in the literature could be classified into four categories based on their search strategy. The schema-based methods are specially designed for schema-based graphs (relational, XML and RDF) [11, 12]. Although the schema of data could be properly applied to facilitate the search process, it is not always available or accessible. Exploration-based methods are based on performing a backward search on the graph starting from keyword nodes to reach a common node. BANKS [15], Bidirectional search [23], BLINKS [16] and DPBF [31] are some of these methods. In Bidirectional algorithm, in addition to a backward search, a forward search is performed from the potential roots of the answer toward keyword nodes. This algorithm uses a heuristic activation strategy to order the nodes to expand. The answers in BANKS and Bidirectional methods are ranked using a notation of proximity coupled with a notation of prestige of nodes based on in-links. BLINKS [16] uses a graph partitioning besides an efficient bi-level indexing structure [4] to accelerate the exploration of the graph. One group of indexes is used to travel between the blocks and another is used to access the data within the blocks. In retrieving an answer, BLINKS begins from a keyword node and searches its block with the help of intra-block indexes. If searching for an answer is expanded to the neighboring blocks, it utilizes multiple cursors and sends each of them to a neighbor block to continue the search in them [19]. There is a major difference between TKS (the proposed method) and BLINKS. TKS uses partitioning to prune the search space and direct the search toward more promising regions, while BLINKS utilizes partitioning to reduce the indexing complexity and localize the search. DPBF [31] is a dynamic programming algorithm for solving the KWS problem. In this algorithm, the answer trees are completed using prior knowledge about the connectivity of the nodes. However, the retrieved answers by this method are distinct root trees. This leads to missing some answers when top-k or all of the answers need to be produced. Park *et al.* [18] proposed a threshold-based algorithm by the assumption of search on multi-dimensional data. In this method, the answers with more coverage of keywords are ranked higher even if they are not minimal.

VD-based methods have been proposed in order to enjoy the success of IR-based search strategies in retrieving and ranking the relevant answers. The main effort of these methods is mapping the graph search space into document search space. For this purpose, the concept of virtual document is defined to integrate the content of some neighborhood nodes. The search is then executed on the virtual documents rather than on the graph. GraphLM [19] is a VD-based method which mainly focuses on ranking the answers. In this method, any answer is mapped to a document by concatenating the texts of its nodes. These documents are modeled using a text-based language model to determine their text-based relevance score to the query. The overall score of an answer is a combination of this score and a structural score obtained based on the structural properties of the answer. One of the main disadvantages of this method is separate consideration of textual and structural weighting of the answers which may weaken the method in distinguishing between relatively similar answers (e.g. answers A_1 and A_2 in Fig. 1). In Cover Density (CD) [24], a group of connected tuples plays the same role as a virtual document. In this method, the answers are ranked first by coordination level (i.e. the number of terms in common with the query) and then by term co-occurrences. Although CD considers some important parameters in scoring the answers, it ignores the structural importance of nodes in the graph which is an effective factor in distinguishing between similar answers. In [43], a novel indexing on tuple units was proposed to identify the answers efficiently. The tuple units were obtained by joining the connected tables. The proximity of nodes and the TF/IDF of keywords in the nodes are two factors considered in this method in ranking the answers.

Lawler-based methods [10, 20, 22] follow Lawler's procedure to retrieve top-k answers to a keyword query. In Lawler's procedure, the search space is divided into disjoined subspaces. The best answer in each subspace is found and

considered as the best global answer. The subspace related to the best global answer is further divided into subspaces and the best answer among the new subspaces and the other subspaces in the previous level is considered as the next best global answer. However, finding a global answer in each step needs an exploration-based search from scratch. Therefore, these methods require more time for retrieving top-k answers than exploration-based methods. In [22], Lawler's procedure was used to enumerate the answers in a 2-approximate order by height. In [20], the r-cliques covering all the queried keywords were represented as answers. In [10], the answers which covered the same set of keyword nodes were considered duplicate. This work proposed some algorithms to produce a list of non-duplicate answers. In all the mentioned Lawler-based methods, the answers are scored based on the sum of the weights of the answer's edges.

7. CONCLUSION

In this paper, we developed a novel structure-aware relevance model to be used in scoring answers (subgraphs) retrieved in response to a given query. This model was estimated based on a wide range of the characteristics of an answer including the contents of its nodes, the distances between its nodes, the importance of its nodes, and the weights of its edges. Employing such comprehensive models in determining the relevance score of answers has led to an effective list of top-k answers. Furthermore, we presented a two-level KWS approach to support the efficient retrieval of top-k answers ordered by the model-based scores. In the proposed approach, a major pruning is performed on the initial graph to direct the search toward regions containing answers. The efficiency of this approach is further improved by estimating the boundary scores of answers in the detected regions. As was shown in the experiments, using these strategies has led to a significant saving in the runtimes of the systems especially over the datasets with high connections.

REFERENCES

[1] Zhou, Q., Wang, C., Xiong, M., Wang, H. and Yu, Y. (2007) SPARK: Adapting Keyword Query to Semantic Search. *Proc. 6th Int. The Semantic Web*, Busan, Korea, pp. 694–707.

[2] Han, S., Zou, L., Xu Yu, J. and Zhao, D. (2017) Keyword Search on RDF Graphs—A Query Graph Assembly Approach. ArXiv e-prints, 227–236.

[3] Jurafsky, D. and Martin, J.H. (2014) *Speech and Language Processing*. Pearson, London.

[4] Aggarwal, C.C. and Wang, H. (2010) *Mining and Managing Graph Data*. Springer, New York Dordrecht Heidelberg London.

[5] Cohen, S., Mamou, J., Kanza, Y. and Sagiv, Y. (2003) XSEarch: A Semantic Search Engine for XML. *Proc. 29th Int. Conf. Very Large Data Bases*, Berlin, Germany, 29, pp. 45–56.

[6] Tran, T., Cimiano, P., Rudolph, S. and Studer, R. (2007) Ontology-Based Interpretation of Keywords for Semantic Search. *Proc. 6th Int. Semantic Web*, Busan, Korea, pp. 523–536.

[7] Ning, X., Jin, H., Jia, W. and Yuan, P. (2009) Practical and effective IR-style keyword search over semantic web. *Inf. Process. Manage.*, **45**, 263–271.

[8] Virgilio, R.D., Cappellari, P. and Miscione, M. (2009) Cluster-Based Exploration for Effective Keyword Search over Semantic Datasets. In Laender, A.F., Castano, S., Dayal, U., Casati, F. and de Oliveira, J. (eds.) *Conceptual Modeling—ER200*, 9. Springer, Berlin, Heidelberg.

[9] Cappellari, P., Virgilio, R.D. and Roantree, M. (2012) Path-oriented keyword search over graph-modeled Web data. *World Wide Web*, **15**, 631–661.

[10] Kargar, M., An, A. and Yu, X. (2013) Efficient duplication free and minimal keyword search in graphs. *IEEE Trans. Knowl. Data Eng.*, **26**, 1657–1669.

[11] Hristidis, V. and Papakonstantinou, Y. (2002) Discover: Keyword Search in Relational Databases. *Proc. 28th Int. Conf. Very Large Data Bases*, Hong Kong, China, pp. 670–681.

[12] Agrawal, S., Chaudhuri, S. and Das, G. (2002) DBXplorer: A System for Keyword-Based Search over Relational Databases. *Proc. 18th Int. Conf. Data Engineering*, pp. 5–16.

[13] Hristidis, V., Gravano, L. and Papakonstantinou, Y. (2003) Efficient IR-style Keyword Search over Relational Databases. *Proc. 29th Int. Conf. Very Large Data Bases*, Berlin, Germany, 29, pp. 850–861.

[14] Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R. and Velegrakis, Y. (2016) Combining user and database perspective for solving keyword queries over relational databases. *Inf. Syst.*, **55**, 1–19.

[15] Hulgeri, A. and Nakhe, C. (2002) Keyword Searching and Browsing in Databases using BANKS. *Proc. 18th Int. Conf. Data Engineering*, pp. 431–440.

[16] He, H., Wang, H., Yang, J. and Yu, P.S. (2007) BLINKS: Ranked Keyword Searches on Graphs. *Proc. 2007 ACM SIGMOD Int. Conf. Management of Data*, Beijing, China, pp. 305–316.

[17] Wang, D., Zou, L. and Zhao, D. (2015) Top-k queries on RDF graphs. *Inf. Sci.*, **316**, 201–217.

[18] Park, C.-S. and Lim, S. (2015) Efficient processing of keyword queries over graph databases for finding effective answers. *Inf. Process. Manage.*, **51**, 42–57.

[19] Mass, Y. and Sagiv, Y. (2012) Language Models for Keyword Search over Data Graphs. *Proc. 5th ACM Int. Conf. Web Search and Data Mining*, Seattle, Washington, USA, pp. 363–372.

[20] Kargar, M. and An, A. (2011) Keyword search in graphs: finding r-cliques. *Proc. VLDB Endowment*, **4**, 681–692.

[21] Lopez-Veyna, J.I., Sosa-Sosa, V.J. and Lopez-Arevalo, I. (2014) A low redundancy strategy for keyword search in structured and semi-structured data. *Inf. Sci.*, **288**, 135–152.

[22] Golenberg, K., Kimelfeld, B. and Sagiv, Y. (2008) Keyword Proximity Search in Complex Data Graphs. *Proc. 2008 ACM SIGMOD Int. Conf. Management of Data*, Vancouver, Canada, pp. 927–940.

[23] Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R. and Karambelkar, H. (2005) Bidirectional Expansion for Keyword Search on Graph Databases. *Proc. 31st Int. Conf. Very Large Data Bases*, Trondheim, Norway, pp. 505–516.

[24] Coffman, J. and Weaver, A.C. (2010) Structured Data Retrieval Using Cover Density Ranking. *Proc. 2nd Int. Workshop on Keyword Search on Structured Data*, Indianapolis, IN, pp. 1–6.

[25] Qin, L., Yu, J.X., Chang, L. and Yufei, T. (2009) Querying Communities in Relational Databases. *IEEE 25th International Conference on Data Engineering*, 724–735.

[26] Li, G., Ooi, B.C., Feng, J., Wang, J. and Zhou, L. (2008) EASE: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-structured and Structured Data. *Proc. 2008 ACM SIGMOD Int. Conf. Management of Data*, Vancouver, Canada, pp. 903–914.

[27] Xu, Y., Guan, J., Li, F. and Zhou, S. (2013) Scalable continual top-k keyword search in relational databases. *Data Knowl. Eng.*, **86**, 206–223.

[28] Lv, Y. and Zhai, C. (2009) Positional Language Models for Information Retrieval. *Proc. 32nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Boston, MA, USA, pp. 299–306.

[29] Zhai, C. and Hirst, G. (2008) *Statistical Language Models for Information Retrieval*. Morgan & Claypool Publishers.

[30] Yu, X., Yu, Z., Liu, Y. and Shi, H. (2017) CI-Rank: collective importance ranking for keyword search in databases. *Inf. Sci.*, **384**, 1–20.

[31] Ding, B., Yu, J.X., Wang, S., Qin, L., Zhang, X. and Lin, X. (2007) Finding Top-k Min-Cost Connected Trees in Databases. *23rd Int. Conf. Data Engineering (IEEE)*, pp. 836–845.

[32] Page, L., Brin, S., Motwani, R. and Winograd, T. (1999) *The PageRank Citation Ranking: Bringing Order to the Web*. Stanford InfoLab.

[33] Zhai, C. and Lafferty, J. (2001) A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. *Proc. 24th Annu. Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, LA, USA, pp. 334–342.

[34] Coffman, J. and Weaver, A.C. (2010) A Framework for Evaluating Database Keyword Search Strategies. *Proc. 19th ACM Int. Conf. Information and Knowledge Management*, Toronto, ON, Canada, pp. 729–738.

[35] Coffman, J. and Weaver, A.C. (2014) An empirical performance evaluation of relational keyword search techniques. *IEEE Trans. Knowl. Data Eng.*, **26**, 30–42.

[36] Manning, C., Raghavan, P. and Schutze, H. (2008) *Introduction to Information Retrieval*. Cambridge University Press.

[37] Craswell, N. (2009) Mean Reciprocal Rank. In Liu, L. and Tamer Özsu, M. (eds.) *Encyclopedia of Database Systems*. Springer US, Boston, MA, pp. 1703–1703.

[38] Feng, J., Li, G., Wang, J. and Zhou, L. (2010) Finding and ranking compact connected trees for effective keyword proximity search in XML documents. *Inf. Syst.*, **35**, 186–203.

[39] Le, T.N., Bao, Z. and Ling, T.W. (2015) Exploiting semantics for XML keyword search. *Data Knowl. Eng.*, **99**, 105–125.

[40] Elbassuoni, S. and Blanco, R. (2011) Keyword Search over RDF Graphs. *Proc. 20th ACM Int. Conf. Information and Knowledge Management*, Glasgow, Scotland, UK, pp. 237–242.

[41] Le, W., Li, F., Kementsietsidis, A. and Duan, S. (2014) Scalable keyword search on large RDF data. *IEEE Trans. Knowl. Data Eng.*, **26**, 2774–2788.

[42] Liu, F., Yu, C., Meng, W. and Chowdhury, A. (2006) Effective Keyword Search in Relational Databases. *Proc. 2006 ACM SIGMOD Int. Conf. Management of Data*, Chicago, USA, pp. 563–574.

[43] Jianhua, F., Guoliang, L. and Jianyong, W. (2011) Finding top-k answers in keyword search over relational databases using tuple units. *IEEE Trans. Knowl. Data Eng.*, **23**, 1781–1794.

[44] Kim, S., Lee, W., Arora, N.R., Jo, T.-C. and Kang, S.-H. (2012) Retrieving keyworded subgraphs with graph ranking score. *Expert Syst. Appl.*, **39**, 4647–4656.

[45] Liu, Z. and Chen, Y. (2011) Processing keyword search on XML: a survey. *World Wide Web*, **14**, 671–707.

[46] Tran, T., Haofen, W., Rudolph, S. and Cimiano, P. (2009) Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF)Data. *IEEE Int. Conf. Data Engineering*, pp. 405–416.

[47] Park, J. and Lee, S.-g. (2011) Keyword search in relational databases. *Knowl. Inf. Syst.*, **26**, 175–193.