A systematic literature review and classification of approaches for keyword search over graph-shaped data

Leila Feddoul [a,*], Frank Löffler [b,a] and Sirko Schindler [c]

[a] *Heinz Nixdorf Chair for Distributed Information Systems, Friedrich Schiller University Jena, Jena, Germany*
E-mail: leila.feddoul@uni-jena.de

[b] *Competence Center for Digital Research, Michael Stifel Center, Jena, Germany*
E-mail: frank.loeffler@uni-jena.de

[c] *Institute of Data Science, German Aerospace Center DLR, Jena, Germany*
E-mail: sirko.schindler@dlr.de

Abstract. Knowledge graphs provide machine-interpretable data that allow automatic data understanding and deduction of new facts. However, machines are not the only consumers of such semantic data. Human users could also benefit from graph-structured data by browsing and exploring it to detect interesting associations and draw conclusions. To achieve that, methods that allow for search over knowledge graphs are highly sought after. Keyword search is an intuitive and common way to retrieve relevant data (e.g., documents) and can also be leveraged to search over knowledge graphs.

In this survey paper, we derive the typical architecture of a system for keyword search over graph-shaped data, we formally define the problem, we highlight related challenges, and we compare to existing relevant surveys to identify the gaps. We conduct a comprehensive review of studies dealing with the topic of keyword search over graph-shaped data (e.g., knowledge graphs) following a systematic method. Based on that, we derive and define different aspects for classifying existing works. We also give an overview about how those systems are evaluated and highlight possible future research directions.

Keywords: Keyword Search, Knowledge Graph, Survey

1. Introduction

Knowledge Graphs (KGs) have shown their importance in many application areas (e.g., search, question answering, or recommendations). Thus, the data represented as KGs is continuously increasing - a fact also evidenced by the continuous growth of KGs like Wikidata [1]. While their semantic data representation can be directly leveraged by machines to accomplish a specific task (e.g., inference), end users and domain experts should also be able to access and explore the KGs. This is a first step in allowing to go beyond simple information lookup and enable the discovering of new facts and correlations given by the graph structure. We can distinguish between two scenarios: (1) *KG querying and search* includes the usage of structured queries, interactive query builders, keyword search, or question answering. (2) *KG exploration* comprises visualization and browsing, faceted search, graph analytics and summarization.

[*]Corresponding author. E-mail: leila.feddoul@uni-jena.de.

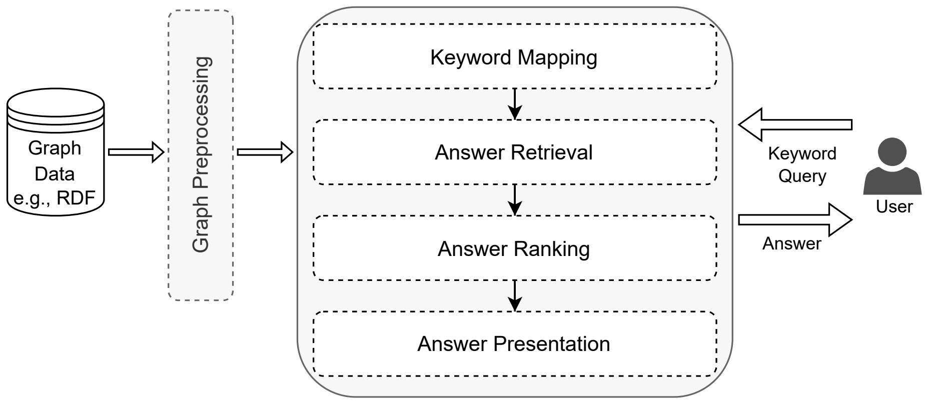

Fig. 1. Typical generic components of a system for keyword search over graph-shaped data

While structured queries allow the formulation of the information need in a precise way, the most familiar and intuitive method to search over data collections is keyword search [2]. This reduces the time to learn new technologies and does not require an intricate knowledge of the underlying domain and the data schema. This technique could also be adapted to work over KGs.

The typical generic components of a system for keyword search over graph-shaped data are depicted in Figure 1:

- *Graph preprocessing*: refers to the operations applied to prepare and index the data graph for efficient query processing (e.g., keyword-node index) and answer retrieval (e.g., shortest paths index or summary graph).
- *Keyword mapping*: this step consists of mapping words or group of words in the query to a graph's nodes and edges.
- *Answer retrieval*: this step consists of finding results relevant to the query (e.g., using graph traversal). The retrieval unit varies depending on the used technique (e.g., tree or entity).
- *Answer ranking*: this step ranks results in the order of relevance and is also dependent on the specific results nature.
- *Answer presentation*: refers to the way the results are provided to the user and the different features that are provided to support the search experience.

Implementing these common system components poses the following main challenges:

- *Finding accurate mappings between keyword terms and graph elements*: this includes a correct interpretation and disambiguation of the query keywords. This is a crucial step since it directly affects the efficiency and effectiveness of the subsequent answer retrieval phase.
- *Scaling to large graphs*: this includes the development of techniques to accelerate the retrieval of answers (e.g., suitable indexing solutions or greedy algorithms).
- *Ranking retrieved answers*: this includes the development of ranking functions that leverage different aspects depending on the type of the answer.

Those challenges lead to increasing research efforts in this area. To the best of our knowledge, there exist four surveys that give an overview about the state of the research on keyword search over graph-shaped data focusing on different aspects. However, two of them [3, 4] are quite outdated (2010), [3] focuses only on relational databases, and another survey [5] dating from 2019 reports only on one aspect (answer ranking). The most recent one [6] classifies existing works based on the type of the returned answer as already proposed in an earlier survey [3]. While the latter also includes an overview of methods depending on the schema of relational databases, [6] only concentrates on schema-free approaches. Both works use a classification that mixes two aspects in one concept, namely the answer type and the way the score of the final answer is calculated. The other survey [4] classifies the works in a rather general way based on the underlying data type, namely *Keyword Search on XML Data* and *Keyword Search over Relational Databases*, which are considered as schema-dependent, and *Keyword Search on Graphs* which refers to schema-free approaches. In [5], authors concentrate on the answer ranking aspect. The proposed factors for the answer ranking slightly overlap with the ones we propose in Section 4, but we adjust and add other ranking types while preserving the distinction between the important aspects. Like [6], they also mix the two factors of answer scoring and ranking. In addition, we also notice the inconsistent use of category names (e.g., "centralized ranking function", to refer to the distinct-root semantics as in [6]).

In addition to the previously mentioned shortcomings, none of the existing surveys report on methodologies used for evaluating the systems including, e.g., the used ground truth, datasets, common benchmarks, and metrics. Furthermore, only one work considered the ranking. Overall, to allow a certain consistency and simplicity of definitions, it would be of benefit to consider each classification aspect separately, as suggested and extended in our proposed categorization (Section 4). In addition, our literature review demonstrates that this area of research is constantly evolving, resulting in new publications that were not included in previous surveys and more works considering KGs. The contributions of this systematic review can be summarized as follows:

- We perform a **systematic literature review** to identify and classify relevant research on keyword search over graph-shaped data.
- We propose a **taxonomy** for classifying existing works based on different aspects: *search space*, *answer type*, *answer ranking*, and *answer scoring*.
- We provide a comprehensive review of **evaluation methodologies** used in the fields.
- We identify remaining research gaps and propose potential **future research directions**.

The remainder of this survey is organized as follows: In Section 2, we define the task of keyword search over graph-shaped data as used within this survey. In Section 3, we describe the methodology for the systematic literature review and in Section 4, we outline the different aspects used for the classification of relevant works. In Section 5, we cluster relevant works based on the *search space* and provide a detailed description for each one while also including the other aspects. In Section 6, we give a summary overview of all the systems and their evaluation methods (cf. Subsection 6.1). Section 7 concludes the review and discusses future research directions.

2. Preliminaries

We consider the task of using keywords to search over graph-shaped data where, given a graph G and a keyword query Q, the goal is to find proper answer(s) $A_1, A_2, ..., A_n$ that satisfy the information need expressed in Q. The response might either be an unranked list of relevant answers, a ranked list by order of relevancy, or just the single-best answer. The relevant inputs and outputs are formally described as follows:

Graph model. We consider the graph $G = (N, E, L_N, L_E)$, where N is a finite non-empty set of nodes, $E \subseteq NxN$ is a finite set of edges connecting two nodes, L_N and L_E are finite sets of textual information (e.g., labels) describing nodes and edges respectively. Depending on the situation, the graph can be directed or undirected and its nodes and/or edges may have weights assigned.

Keyword query. A keyword query Q consists of a non-empty list of terms $Q = k_1, k_2, ..., k_n$. The set of nodes K_i with a textual content matching k_i for $i = 1, ..., n$ are called *keyword nodes*. A keyword node can be associated to multiple keywords and the same keyword can be mapped to various nodes. This matching can be extended to include edges as well, e.g., by also considering them as nodes in the graph [7, 8]. The mapping between keywords and nodes can be provided externally or be included as an additional task that is either a mere string matching or an entity linking [9] where multiple keywords could correspond to a single entity in case of KGs.

Query answer. A relevant answer A_i to the query Q is a connected subgraph $S = (N_S, E_S)$ (for every pair of nodes $n_1, n_2 \in N_S$ there is a path that connects them [10]) that covers all the keywords (the subgraph contains for each keyword at least one corresponding *keyword node*). A relaxed variant will also allow for answers covering only a subset of keywords. This is a general description, without imposing any further restrictions on the characteristics of the answer. In Section 4, we give a detailed overview about possible answer types mentioned in the literature.

3. Survey Methodology

To collect relevant works in the area of keyword search over graph-shaped data, we performed a systematic literature review inspired by the guidelines provided by [11, 12]. This allowed to summarize and classify existing related work as well as determine remaining research gaps. The general process consists of three phases:

1. Plan review: includes the identification of the need for the review and the development of the review protocol which describes the steps that will be followed while conducting the review.
2. Conduct review: includes the concrete steps of performing the review starting from the identification of relevant publications.
3. Document review: includes writing a report about the review and publishing it.

Since the *identification of the need for the review* was already addressed in Section 1, the *review protocol* will be clarified while describing the second phase, and the last step is addressed by this survey, we only describe the *conduct review* phase in more detail in the following:

A. Identify relevant research. In this step potentially relevant candidate papers that match specific predefined keywords are collected. In practice, we first searched the DBLP[1] bibliographic database using the following queries[2]: *keyword search graph*, *keyword search RDF*, *keyword quer RDF*, *keyword quer graph*, *quer generation graph*, *quer generation RDF*, *keyword exploration graph*, and *keyword exploration RDF*. Then, duplicates are automatically detected and removed after saving the original search results for each query. Publication collection and deduplication were performed using the *web search* functionality of the reference management software JabRef 5.7[3] and resulted in 206 candidates.

B. Select primary studies. In this step, a primary selection of relevant papers is performed based on predefined inclusion and exclusion criteria and a first reading of the title and abstract. In our case, this selection resulted in 68 papers. The applied criteria were as follows:

- *Inclusion criteria.*
  - Publication year starting from 2013 (last 10 years).
  - Open/institutional access.
  - English language.
  - Paper type: conference, journal, or workshop paper.
  - Content fits to the topic of *keyword search over graph-shaped data*[4].
- *Exclusion criteria.*
  - Paper type: PhD symposium, Demo/poster, or extended abstract.
  - Not peer reviewed (e.g., only published on an open access archive).
  - Newer paper of the same approach or system exists (keep the newest).

C. Assess study quality and extract/synthesize data. In this step a more detailed reading of the papers takes place to allow their classification based on the criteria defined in Section 4. During this process, we may also decide against including some specific type of works as part of the works overview (cf. Section 5)[5]. The data extraction and synthesis is also done in parallel for each paper by: (1) collecting information (e.g., classification criteria and evaluation method) which allows further analysis, and (2) drafting a short descriptive summary. At the end we selected and classified 35 works.

[1]https://dblp.org/

[2]The keyword *quer* is deliberately used to include both singular and plural forms in the following queries.

[3]https://www.jabref.org/

[4]We did not include works with very specific use cases (e.g., working over multiple data sources or dealing with spatial keywords) or works performing keyword search over XML [13] or over text documents.

[5]Those papers include works dealing with e.g., distributed graphs, incomplete graphs or the usage of parallel processing. They will be just shortly mentioned as possible other special aspects in Section 4.

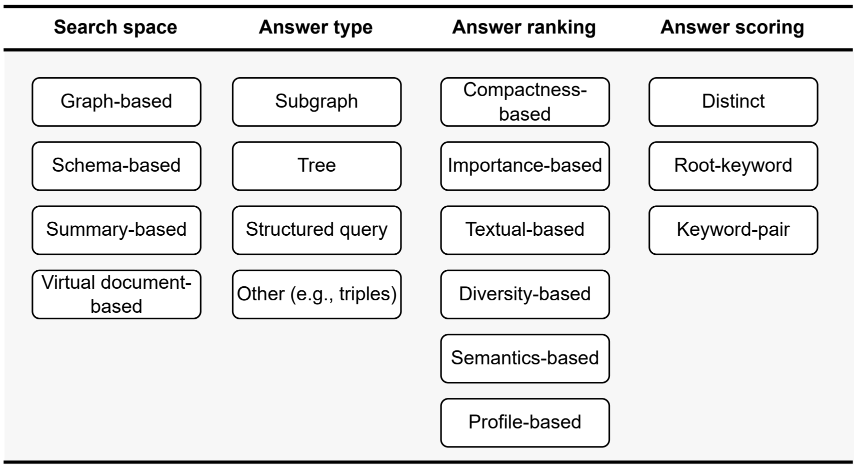

Fig. 2. Classification of approaches for keyword search over graph-shaped data

We also included a subset of works that were published before 2013. This includes popular first works dealing with the keyword search problem over relational data [14–16], in addition, to one of the very first works operating over KGs [7], and other three works [17–19] already read before performing the systematic literature review. These publications were added due to their impact on later works.

4. Taxonomy overview

The literature review allowed us to derive four important aspects for categorizing related works as shown in Figure 2, namely the *search space*, *answer type*, *answer ranking*, and *answer scoring*.

Search space. Represents the nature of the underlying data searched to find relevant answers. We distinguish between four types:

– *Graph-based.* Search for answers over the whole graph.
– *Schema-based.* Operate on a provided data schema to find answers.
– *Summary-based.* Operate on a summarized version of the data graph without relying on a predefined schema.
– *Virtual document-based.* Build documents from the textual content of the graph elements and retrieve relevant ones using information retrieval techniques.

Answer type. Represents the nature of the final answer to be returned by the system. According to the analyzed works, we can distinguish between the following answer types:

– *Subgraph.* The answer is a set of connected keyword nodes. A special case is the *r-clique*, where the answer is a subgraph connecting keywords and the shortest distance between any pair of keyword nodes is equal to or lower than *r* [20].
– *Tree.* The answer is a graph that has no cycles (tree) and that connects all keyword nodes [10]. A tree can be either directed or undirected and further characterized using the following properties:
    ∗ Rooted directed: a rooted directed tree is a tree in which a specific node is distinguished and called *root*, and there exist directed paths (pointing away from the root) from the root to each keyword node [14].
    ∗ Distinct rooted: a distinct rooted tree is a tree with a distinct root with respect to a collection of top-k answers [16].
    ∗ Minimal: the minimality is defined here with respect to the keywords and states that all the leafs of the tree are keyword nodes (In other words, there are no leaf nodes not containing keywords [21], or the answer has no other subtree containing the keywords).

- *Structured query.* The answer is a query equivalent to a previously found intermediate answer (subgraph or tree). When executed, the query returns relevant concrete results (e.g., triples or entities in case of RDF graphs).

- *Other types.* There are also specific cases, where relevant answers are triples or entities of an RDF graph. Those are directly retrieved without having an intermediate step where a subgraph/tree/query are formed. This is usually the case when a graph's textual content is indexed (e.g., triple index), and the answers are retrieved using traditional ranking functions as used in document retrieval. Another specific answer type, introduced by just a single work, is the *Key-core* [22], which represents a subgraph that is formed by connecting two minimal rooted directed trees containing all keywords. Furthermore, for each tree the distance between each keyword node and the root, and the sum of keyword-node distances are smaller than two specific thresholds.

Answer ranking. In the case where more than one relevant result are retrieved, answers are ranked based on different criteria. The ranking also depends on the answer type. In general we distinguish between six ranking categories, that are either individually applied or combined to rank answers:

- *Compactness-based.* Use the size (e.g., tree height or number of nodes) of the retrieved structure as quality criterion.
- *Importance-based.* Use specific criteria (e.g., node in-degree) that may reflect the popularity and importance of the answer elements.
- *Textual-based.* Use matches between query keywords and the textual content of the answer as a ranking factor (e.g., TF/IDF [23]).
- *Diversity-based.* Include criteria that aim at penalizing results showing a certain aspect of redundancy.
- *Semantics-based.* Use semantic characteristics of KGs to rank answers (e.g., semantic distance).
- *Profile-based.* Use user profiles to provide answers closer to the user intent.

Answer scoring. The final score of a graph-shaped answer is usually calculated by aggregating node and edge weights. Since the usual aim is to find answers with minimum weights (or cost) (e.g., Steiner tree [24]), the score of an answer is in general inversely proportional to its weight. Thus, answers with higher scores (lower weights) are considered more relevant and thus ranked higher. We distinguish between three *answer scoring* schemes that can be used to calculate total weights for both nodes and edges (graph elements):

- *Distinct scoring.* The total weight is calculated as the sum of individual weights: $\sum_{x \in S} weight(x)$, where x is a graph element and S is either the set of edges or the set of nodes. Here each included graph element is counted a single time.
- *Root-keyword scoring.* The total weight is calculated as follows: $\sum_{keyN \in N_S} \sum_{x \in path(r,keyN)} weight(x)$, where $keyN$ is a keyword node, N_S is the set of answer's nodes, r is the root, x is a graph element, and $path(r, keyN)$ is the set of the specific elements between the root and a specific keyword node. In summary, the answer weight is the sum of the graph element weights belonging to the path between the root[6] and each keyword node. In this case, graph elements could be counted multiple times, e.g., in the case of having keyword nodes that are not tree leafs. This considers the distribution of each keyword node with respect to the root.
- *Keyword-pair scoring.* The total weight of an answer is calculated as follows $\sum_{i=1}^{|Q|} \sum_{j=i+1}^{|Q|} weight(keyN_i, keyN_j)$, given a query Q. In other words, it is the sum of the weights between each pair of keyword nodes. It is worth mentioning, that in specific cases such as in [25], the weight used in the keyword-pair scoring, is replaced with a value reflecting the connectivity between two nodes. Thus, it is interpreted as directly proportional to the total score.

[6]In the case of retrieving subgraphs, a specific node that connects all the keywords (e.g., connecting element in [7]), is considered as equivalent to the tree root.

In the same final score two different scoring scheme can be combined, one for edges and the other for nodes. Furthermore, some of the works have restricted the nodes incorporated while calculating the edge weight and consider only specific ones (e.g., root and leafs containing keywords).

During the literature review, we also came across several specific groups of works that deal with the following aspects: *parallel processing* [26–30], *probabilistic/uncertain RDF graphs* [31], *temporal graphs* [32, 33], *distributed graphs* [34], *incomplete graphs* [35], or *search results diversification*[7] [36–39]. We considered them as special use cases, and thus decided to only shortly report them here without further analysis and refrain from including them in the overview tables (Section 6).

5. Classification of existing works

In the following, we give a detailed description of the collected relevant works. We structure the next subsections based on the *search space* aspect, since it appeared to best cluster relevant works in distinct groups. The other aspects will be clearly mentioned for each work on the short summary together with the method of evaluation used. In addition, a summary overview including all aspects is given in Section 6. In each subsection, the works are sorted in ascending chronological order.

5.1. *Graph-based methods*

[14] presented BANKS (Browsing ANd Keyword Searching), a system enabling keyword search over relational databases.

Method. The database is modeled as a directed graph by considering the tuples as nodes and the foreign key relations as edges. They define the query answer as a *rooted directed tree*. BANKS models each relation between two nodes using two types of edges: forward (initial edges pointing from foreign key to primary key) and backward (additional edges in the opposite direction). This ensures finding a rooted tree directed away from the root. BANKS aims at finding an approximate set of top-k minimum Steiner trees (hard problem: NP-complete). It starts by mapping query terms to nodes using an index (text contained in graph elements), then the backward expanding search algorithm concurrently starts a shortest path finding algorithm from each keyword node with the aim of finding a node connecting all keywords (information node). To avoid waiting for all answer trees to be generated to sort them, an approximate solution is proposed. Generated trees are incrementally added (after duplicate detection[8]) to a small fixed-size heap (ordered by relevance score). When the latter is full and there are more candidate answers, the highest ranked tree is returned and replaced in the heap. While the proposed heuristic does not necessary ensure the retrieval of trees in decreasing order, the authors state that it works well even using small heaps.

Ranking. An *importance-based* ranking was used where each node is assigned a weight based on its importance (node prestige). The node weight is calculated as the fraction between the number of incoming edges and the maximum node weight giving a higher score to nodes with more incoming edges. For the edges, a weighting based on the edges importance (proximity) is proposed. Each forward edge is given a weight of 1, whereas a backward link (v, u) is given a weight that is directly proportional to the number of links to v coming from nodes having the same type as u. The edge weight is normalized using the minimum edge weight. The total score of an answer is then calculated by combining both node and edge scores by addition (*distinct scoring* for nodes and edges) or multiplication using a factor λ to control the effect of edge and node scores ($\lambda = 0.2$ performed best). The edge score gives higher relevance to smaller trees. Only root and keyword nodes are considered while calculating the total score of the nodes and a node containing multiple search terms is counted multiple times.

[7]This approaches aim to diversity answers at the same time when searching the graphs and thus propose algorithms that are directly geared towards producing diverse results. This is different from the diversity-based ranking, since the latter aims at reducing redundancy of already generated answers using an algorithm that is not necessarily tuned to take this aspect into consideration.

[8]Before adding a result to the heap a duplicate trees detection (e.g., trees with same undirected version) is conducted and the tree with the highest relevance is kept. If the duplicate of the new result have already been output it is discarded even if its relevance is better.

Evaluation. Evaluation of performance, effectiveness together with parameters effect were conducted using a part of the DBLP dataset (paper titles, their authors and citations) and a dataset about the master/PhD dissertations in the IIT Bombay. They used an internally created gold standard with 7 queries and their corresponding relevant answers. The rank difference between the ground truth answers and the actual answers was used as an evaluation metric.

[15] state that the backward search in BANKS may not efficiently perform in the cases where: (1) a query keyword has many matching nodes in the graph, or (2) some of the visited nodes have a large in-degree, since it prevents from exploring other directions until all incoming edges are visited.

Method. To overcome the previous shortcoming, they propose a bidirectional search paradigm to retrieve *rooted directed trees* which does not traverse the graph only backwards but also forwards starting from potential answer roots in the direction of keyword nodes. For this purpose, they propose a node prioritization heuristic (spreading activation) that regulates the traversal. Keyword nodes are assigned an initial score given by $a_{n,i} = \frac{nodePrestige(u)}{|K_i|}, \forall n \in K_i$ where K_i is the set of nodes matching a keyword k_i. This prioritizes keyword nodes having a higher prestige and penalizes the ones matching a large number of nodes. Each node propagates an attenuated activation to its neighbors $\mu = 0.5$ and keeps the remaining $1 - \mu$. In case a node receives activation from different edges starting from a keyword k_i, the maximum activation $a_{(u,i)}$ is considered. The overall activation of a node is calculated as the sum of the activation scores originating from each keyword, to prioritize nodes that are close to a larger number of keywords.

Ranking. Since the focus was on the search algorithm, and not on the ranking of answers, the same *importance-based* scoring as proposed by BANKS was used with the multiplication-based relevance score. The only difference is the used final answer scoring for edges, which is *root-keyword* based in contrast to BANKS.

Evaluation. The evaluation focuses on the efficiency aspect and compares the bidirectional search with BANKS and the Sparse algorithm [40] using the following metrics and 5 manually created queries: number of the nodes explored and the nodes touched as well as the time taken. Results show that the bidirectional search is faster than both baseline algorithms. A very small effectiveness analysis (precision and recall (cf. Subsection 6.1)) was conducted using the same queries and their corresponding results generated using an SQL query. Both backward and bidirectional performed equally with respect to the effectiveness.

Both backward and bidirectional search algorithms performance relies more or less on the graph structure and the position of the different keywords in the graph. This situation makes the performance unpredictable. To deal with that, [16] proposes BLINKS (Bi-Level INdexing for Keyword Search) and aims at exploiting indexes to precompute and store possible shortest paths.

Method. The index is filled using backward search and consists of two kinds of lists : (1) a *keyword-node list* that stores for each keyword the list of nodes that can reach it ordered by distance, and (2) a *node-keyword map* which stores the shortest distance between nodes and keywords, to optimize forward expansions. BLINKS does not aim at indexing all possible paths (single-level index) since it will result in very large indexes for large scale graphs. However, it proposes a Bi-Level index by dividing the whole graph into blocks (subgraphs): the *top-level block index* maps keyword/nodes to blocks, and the *intra-block index* stores detailed information within each block (e.g., intra-block keyword-node lists). They also enhance the backward search strategy using *equi-distance* and *cost-balanced* expansions. BLINKS answers are *distinct rooted directed trees* to avoid answers that deliver only few additional information compared to the rest and technically allow more effective indexing.

Ranking. Again, here the focus was not on the ranking, but on the query processing and indexing strategy. They propose a scoring function of the answer that combines three aspects (*compactness-based , importance-based, and textual-based*): sum of the shortest path distances from the root for each keyword node, sum of the matching score for each keyword with its corresponding node (e.g., TF/IDF), and the score of the answer root (e.g., PageRank [41]). However, for convenience they only consider the sum of the shortest path distances from the root for each keyword node. They use the same aggregation of edge weights (*root-keyword scoring*).

Evaluation. Only efficiency experiments (comparison with the bidirectional algorithm [15]) were conducted including execution time with different parameters (e.g., graph partitioning method) and index performance using 10 queries over the DBLP XML[9] and IMDb[10] datasets. Results reveal that BLINKS is overall faster than the baseline.

[17] proposes different algorithms to efficiently retrieve all answers.

Method. They use threaded enumerators, that enable one enumeration algorithm to directly use elements produced by another one (or itself) instead of waiting until the latter finishes. The desired answers are *minimal trees*, this means that there exist no other subtree in the answer that connects the respective keyword nodes. They consider three types of answers: *directed*, *undirected*, and *strong* (undirected with all keywords as leaves). They focus on the efficiency aspect and aim at proposing answer enumeration algorithms that solve the problem with polynomial delay between two output answers except for the first answer were the delay is exponential. For that, they investigate algorithms for each answer type, with the following different purposes with respect to how the answers are presented: *sorted*, *heuristically sorted*, and *unsorted*. They consider the sorted variant as the most wanted one and propose an algorithm with polynomial delay given an acyclic data graph and a directed answer. They also propose algorithms for the unsorted scenario for all types of answers with polynomial delay. Furthermore, they give a proposition about how can the latter be enhanced to output the answers in a heuristic order. Answer ranking was not addressed and there is no practical evaluation of the proposed approaches, instead theoretical proofs are elaborated.

[19] combines the methods proposed by [14, 15] by using shortest-path iterators to find the first answer, and includes the work done in [17] to generate the rest of not redundant answers using a more practical and faster approach.

Method. [17] retrieves answers (*minimal rooted directed trees*) with a polynomial delay (ranking them by the order of increasing weights) and is based on heuristics for generating Steiner trees. To simplify the problem, [19] proposes the usage of two rankings, a simple one while exploring the graph, and another one after generating candidate answers. A condition for this to work is that the primary ranking should be highly correlated with the final one. They simplify the problem further to increase efficiency and restrict to enumeration of answers in a 2-approximate order ("if one answer precedes another, then the first is worse than the second by at most a factor of 2").

Ranking. In general, they combine *compactness-based, importance-based, and diversity-based* rankings. The primary ranking weight used during the exploration is the height of a subtree ("maximum length of any path from the root to a leaf"). The final ranking of generated answers is calculated as a combination of: an *absolute relevance score* based on the in-degree/out-degree of the nodes belonging to an edge, and the *redundancy penalty* which aims at producing diversified answers by "penalizing answers based on their degree of similarity to the ones that have already been given a final rank". The total score of an answer is inversely proportional to its total weight using a *distinct scoring* for nodes and edges.

Evaluation. The efficiency of the algorithm was evaluated with increasing number of query keywords. The correlation between the primary and final ranking together with the effect of the redundancy penalty on ranking quality are also evaluated. No information was provided on the used evaluation dataset/queries.

[42] aims at retrieving another type of results called *r-clique* which is a subset of connected vertices that should contain all keywords, and the vertices pairwise shortest distance should be at most *r*.

Method. The first step is the weights' assignment to nodes and edges in the graph. After that, the search space is narrowed down by selecting only the top-k nodes that match the keywords using the node weights. Then for each pair of selected nodes the shortest path between them is calculated, and the top-k r-cliques are generated.

Ranking. The node weights are calculated as a function of the keyword frequency and PageRank, and the edge weights are calculated as function of the degree of their respective nodes. The total score of an answer is a linear combination of the sum of the node weights and edge weights (*distinct scoring*), multiplied by other factors (answer size, keywords count in the query). The ranking is a combination between *compactness-based*, *textual-based* and *importance-based* factors.

[9]https://dblp.uni-trier.de/xml/
[10]https://www.imdb.com/

Evaluation. The evaluation was performed using 5 randomly generated queries from each of the DBLP and IMDb datasets. Relevance judgments were carried out by 5 users that were asked to score the nodes in the results based on the relevance to the query. Results show that the proposed approach is faster and more effective compared to the Dup-Free algorithm [43].

[44] proposes a system that exploits indexing and semantic similarity scores to find the triples relevant to a specific keyword (not query).

Method. The first step is the offline preparation of five indexes: predicates, subject/object, incoming/outgoing vertices, literals, and similarity scores between two literals. The approach starts by finding the literal vertex corresponding to a certain keyword, filters candidate nodes without a common predicate with the input keyword node, traverses the graph in a depth-first manner and considers only nodes (literals) with a semantic score greater than a specific threshold. The list of relevant triples is presented to the user.

Ranking. The ranking of target nodes (or triples) is performed based on the similarity score which is calculated as a combination of a distance score (number of subjects relating source and target), and a semantic similarity depending on the most specific class subsuming the classes of the pair of nodes [45], calculated over WordNet [46]. The ranking is a combination between *compactness-based*, and *semantics-based* factors.

Evaluation. Only an efficiency evaluation was performed using 10 manually created keywords over three subsets of DBpedia [47]. The approach was compared with the RDF management system Jena [48] by retrieving the first 2 results (using SPARQL queries for Jena). Results show that the proposed system is faster.

[49] aims to find a practical algorithm to solve the hard problem of retrieving top-k *r-cliques* using an approximation with polynomial delay.

Method. First, needed indexes are prepared which includes keyword-node index and a shortest path index between each pair of nodes within a certain distance. After finding keyword nodes in the graph, top-k r-cliques are calculated by adapting Lawler's procedure [50]. First a polynomial algorithm is called to retrieve the best one answer over the whole search space. Then, the latter is divided into subspaces according to the best answer, and the locally best answers in each subspace are found using the same polynomial algorithm. The subspace from which the best answer originates is further divided to find locally best answers. This process continues until all top-k answers are retrieved. Finally, a Steiner tree is produced over each r-clique.

Ranking. The edge weight used during the evaluation is based on the in-degree of the two edge nodes. The final answer score is calculated using a *keyword-pair scoring* considering the edge weight as pairwise node weight.

Evaluation. The efficiency evaluation is done using 15 queries over 3 datasets (DBLP XML, IMDb (MovieLens[11]) and Mondial[12]) where two approach versions (original and another one starting from nodes containing rarest keywords) were compared with BANKS, Dynamic [51], and an algorithm that produces communities as answers [52]. In addition to the runtime also answer compactness was reported. Both proposed versions are faster than the other systems. The effectiveness was evaluated by first calculating the percentage of produced ground truth (generated using an naive algorithm that produces all r-cliques) answers using their approximate approach. In addition, a small user study was conducted with 4 manually created user queries over DBLP and 8 users were asked to rate the answers. The proposed system, BANKS, and Dynamic achieve better precision than the community algorithm over all queries using the P@k with *k* is equal to 10 and 2. (cf. Subsection 6.1).

[53] proposes an index-based approach that decreases memory consumption while performing keyword search over graphs.

Method. First the RDF graph is transformed to an attributed graph that consists of interlinked subject nodes with text information (predicate, class, and predicate linking to another entity). The approach starts with an offline phase where the index is constructed. The index stores for each keyword in the data graph the corresponding list of pair of nodes (node1 containing the keyword, node2 reachable from node1), and the shortest distance between the pair of

[11]https://grouplens.org/datasets/movielens/
[12]https://www.dbis.informatik.uni-goettingen.de/Mondial/

nodes over 2 hops. The index is leveraged in an online phase to extract the top-k *minimal trees* using a depth-first search algorithm and then transform them to a corresponding SPARQL query.

Ranking. The ranking is *compactness-based* using the tree diameter, which is calculated as the max distance among the shortest distances of all pair of nodes.

Evaluation. The evaluation mainly concentrated on the efficiency aspect comparing the indexing time, memory usage, and the runtime with two other index-based approaches (Rclique [54], Gdensity [55]). Results show that the proposed system is faster and requires less memory usage over the two datasets DBLP XML and DBLP2[13]. The effectiveness was manually checked and compared with Rclique. The keyword queries were manually created, by first randomly selecting 20 subgraphs with diameter 4 over DBpedia, constructing corresponding SPARQL queries, and formulating keyword queries by selecting one or two keywords from each node. Afterwards, they manually check if any of the top-k SPARQL queries generated by the systems matches the ground truth.

[56] proposes an index-based approach for answering keyword queries over graphs that can be modified to also accept answers that have more than one node corresponding to a specific keyword.

Method. Similar to [53], the approach starts by constructing an inverted index that stores for each keyword in the graph corresponding nodes (containing the keyword or connected to other nodes containing the keyword) together with their relevance score to the keyword. This index is used to find top-k *distinct rooted trees* using the Threshold algorithm [57, 58].

Ranking. Each node is assigned a *textual-based* and *compactness-based* score that is calculated as the multiplication between two elements: a matching score calculated based on the number of occurrences of the keyword in the node, and a distance score calculated as the shortest path between the considered node and the one containing the keyword. The total relevance score of an answer tree is the sum of the relevance scores between the root and the keyword nodes (*root-keyword scoring*).

Evaluation. The system was compared (shortest distance set to 4) with BLINKS using 30 queries manually constructed over three datasets (Mondial, IMDb, and DBLP XML). The answer trees of the different approaches were rated by five adjudicators, and the P@10 and P@20 were calculated. The proposed approach outperforms BLINKS for 73% of the queries, but the latter is still faster.

[59] proposes a method that transforms the RDF graph to a bipartite graph.

Method. The method aims at optimizing the efficiency of the keyword search task by taking advantage from the nature of the adjacency matrix of a bipartite graph. First the RDF graph is transformed to a bipartite graph that consists of two sets represented as nodes with labels: one set with entities/classes and another one with properties. First, the keyword query is expanded using synonyms from WordNet. The result is then matched to graph elements and a connecting subgraph is extracted.

Ranking. The list of possible subgraphs is ranked using a *textual-based* score based on the number of keywords contained in a specific subgraph.

Evaluation The evaluation is performed by comparing with 3 other systems (KREAG [60], BLINKS and EASE [61]) using DBLP with 10 manually created queries. Both the execution time and effectiveness are reported using P@5, Average Precision at 5 (AP@5), and the Mean Reciprocal Rank (MRR) as metrics (cf. Subsection 6.1). Results show that the proposed approach is faster and more accurate.

[62] proposes a SPARQL query generation method specific to DBpedia.

Method. The first step is query preprocessing using WordNet followed by the keyword matching of query terms to DBpedia concepts/properties/instances. The next step is a query expansion with semantically related concepts using some properties (e.g., rdfs:seeAlso or owl:sameAs). Afterwards, ambiguous terms are filtered by calculating the pairwise relatedness between keywords. The latter is based on the intersection between the set of matching terms (DBpedia IRIs text in this case) for each keyword. Then, all possible combinations are generated from the matching

[13]https://snap.stanford.edu/data/com-DBLP.html

elements, candidate subgraphs for each combination are constructed using predefined graph patterns, ranked, and the top-1 result is translated to a SPARQL query.

Ranking. The subgraphs are ranked based on a *compactness-based* measure which is calculated as the sum of the shortest paths between each pair of matching elements (*keyword-pair scoring*).

Evaluation. The effectiveness of the system is evaluated using 50 queries, and the following metrics: The fuzzy precision which is calculated as the sum of the average correctness rate for each query divided by the queries returning results. The correctness rate of an answer is calculated as the set of correct terms that match the user intention (subject, predicate, object) divided by the set of all terms. The other used metrics are recall (queries returning results divided by the total number of queries) and the F1-measure (cf. Subsection 6.1) as a combination between the fuzzy precision and recall. No information was provided about the origin of the queries, the used ground truth, and the dataset. The reached results are as follows: 0.52 (F1), 0.43 (recall) and 0.5 (fuzzy precision).

[63] proposes an efficient and accurate keyword search system over RDF data by leveraging bipartite graphs together with translation-based graph embeddings.

Method. The method starts with a *segmentation and annotation* phase where the user query is first tokenized, annotated using types (entity, class, relation), and then matched to a specific graph element. The result is a ranked list of possible query annotations. The second step is the *query graph assembly* where the detected graph elements need to be linked with each other. For that, they model the found possible matches as a bipartite graph where each pair of entities/classes and each set of possible matching relations is represented as a node, and there is a crossing weighted edge between the set of entities/classes and the set of edges. The algorithm tries then to find the optimal connection (subset of crossing edges) called *assembly query graph* with the minimum cost and transforms is to a SPARQL query. The constraint here, is that the query should ideally contain classes, entities and as many needed relations, which is not always present given the nature of keyword queries and in this case the graph elements cannot be connected. The authors point to this issue and shortly state that in this case the missing relation should be first determined using a suitable algorithm for minimum spanning tree finding.

Ranking. Since the goal is to find just one best answer, the algorithm does not output a list of top-k possible answers. However, to know the optimal answer a certain function should be used to score the possible subgraphs. They use a *semantics-based* scoring where edge weights are calculated using TransE [14] [64], a translation-based graph embedding model. The total cost of a candidate subgraph is the sum of its edge costs.

Evaluation. The system is compared in terms of efficiency and effectiveness with two approaches: DPBF [51], a graph-based approach, and SUMG [7] a summary graph based approach. Two datasets were used QALD-6 (6th Open Challenge on Question Answering over Linked Data) [65] with 100 queries, and Free917 [66] with 80 selected queries converted to keywords. They also compared their approach using graph embeddings with two traditional link prediction methods. The system performed better than both other approaches and had the third place in QALD-6 competing with question answering systems. The most time consuming part was the keyword-graph element mapping, but on average the system was faster than the two baselines.

[67] focuses on the problem where keywords cannot be mapped to the underlying graph elements because of missing information (e.g., relation).

Method. The main idea is to leverage external knowledge defined in form of patterns which will add new missing triples. The pattern defines an equivalence between a property and a property path (e.g., two authors are collaborators if they have wrote the same paper). Those patterns are manually defined and are either domain specific or generic by using standard properties (e.g., owl:sameAs). The approach starts with a fragment extraction step that extracts matching graph elements, next the set of elements is expanded using a pattern store. Possible fragment combinations are constructed by applying a cartesian product. For each possible combination the smallest *minimal subgraph* is retrieved.

Ranking. The used ranking is both *textual-based* using a term-frequency based matching score, and *compactness-based* using the size of the subgraph (sum of nodes and edges). Patterns are considered when computing the size of a subgraph by reducing the size by 1 if the subgraphs contains a path from a pattern.

[14]https://github.com/thunlp/Fast-TransX

Evaluation. Two datasets were used: AIFB[15] and a dataset about conferences. The actual approach using patterns was compared with two other configurations without external knowledge. The effectiveness is evaluated with P@k and the normalized Discounted Cumulative Gain at k (NDCG@k) with *k* is equal to 5, 10, and 20 (cf. Subsection 6.1) using 10 randomly generated queries and the ratings of 4 users. The usage of patterns outperforms the other configurations. The effect of the query size on the execution time is also evaluated.

[68] has the same motivation as [67] and focuses on enhancing the keyword to graph element matching by using WordNet as external knowledge base.

Method. The approach starts with keyword matching where first graph elements with exact matching labels are retrieved. If this step fails, related terms are retrieved from WordNet using different relations (e.g., synonyms). Since for each keyword a list of matching elements is retrieved, possible combinations are derived using the cartesian product. For each possible combination the smallest *minimal subgraph* is retrieved using the Dijkstra Algorithm [69].

Ranking. The result subgraphs are ranked based on a function that depends on the number of matching elements and the number of connecting nodes. This function favors small answers with more exact matching nodes (*textual-based* and *compactness-based*).

Evaluation the evaluation was done using 10 queries from each of AIFB and a subset of DBpedia about movies. They show that the approach without external knowledge usage is faster and evaluate effectiveness using a subset of 10 queries and three users. Results reveal that P@10 and P@5 are always above 0.92 for both datasets.

[21] addresses the efficiency issue by ignoring some redundant intermediate results during graph expansion.

Method. They consider directed graphs and define a query answer as a *minimal rooted undirected tree* with a fixed size (number of nodes). The rationale behind their algorithm (Canonical Form-based Search) is to consider only intermediate result trees that have a canonical form [70] while searching the graph. Answer ranking was not addressed.

Evaluation. Experiments are run to evaluate the efficiency of the algorithm using the TPC-H benchmark[16] (decision support database) and the IMDb[17] database and 10 randomly generated queries. They compared with other two systems that generate all answers without redundancy checking [71, 72]. Results reveal that the proposed approach is faster than the baseline for retrieving trees of size six. Furthermore, their algorithm always produces less number of intermediate results compared to the baselines.

[25] is motivated by the fact that shortest path-based methods will assign the same score to connecting subgraphs having the same length.

Method. They propose a function for assigning scores to nodes based on a random walk with restart. They define attributed graphs as underlying data and aim at finding a top-1 or top-k *subgraphs* that cover all keywords. Since this is a rather a hard problem, an approximation is proposed. The algorithm starts from the nodes with the so-called *rarest keyword* (appearing in the fewest nodes in the whole graph) and constructs its surrounding subgraph. This is a greedy approach that reduces the search space from the beginning and thus increases performance. Afterwards, a random walk with restart is run which will assign a score to each neighbor node and the best node is selected until finding the best answer. The entire process is repeated to retrieve the top-k answers. Since the random walks are the bottleneck of the algorithm they propose to approximate it using the Monte Carlo method [73] which does not require knowing the whole graph at each iteration. They also design a parallel version of the same algorithm since each walk is independent from the other.

Ranking. The used ranking is *importance-based* and assigns scores to nodes relative to other by taking the whole graph into consideration. In contrast to PageRank, the proposed approach is query-dependent. The total answer score is calculated using the *keyword-pair scoring* considering node weights.

[15]http://km.aifb.uni-karlsruhe.de/ws/eon2006/ontoeval.zip
[16]https://www.tpc.org/tpch/
[17]https://relational.fit.cvut.cz/dataset/IMDb

Evaluation. They evaluate their method using DBLP XML and IntAct PPI[18] (molecular interaction data) datasets and compare with six systems (shortest-path-based, embedding-based, PageRank based). They evaluated the quality of the answers using different dataset-dependent metrics and the runtime of their distributed version. For IntAct PPI, they calculate the expression level of each protein, where "a cost value is assigned to each protein; the lower the cost, the higher the expression level" using 100 randomly generated queries with 2 to 6 genes. For the DBLP dataset, the h-index is used as metric of quality with 100 randomly generated skill sets (2 to 6 skills). For the discovered subgraph the average h-index of the members is calculated. Their approach results in answers with higher expression level and higher average h-index compared to the others.

[20] aims at retrieving minimal r-cliques called *minimal covered r-clique*. The motivation behind that is to overcome the limitations of finding Steiner trees, distinct root trees (losing results having the same root, considering the distance to the root not between each two keyword nodes), r-cliques and r-radius Steiner graphs [49, 61, 74] (not minimal) based on previous definitions.

Method. They use adapted versions (BKS and its improvement called BKSR) of the Bron-Kerbosch [75] and Tomita [76] algorithms that originally aimed at finding maximal cliques (with the largest possible number of vertices). They also improve the algorithms to return non-duplicate minimal covered r-cliques. Furthermore, they propose algorithm versions (BKSM, BKSRM) that are dedicated to also work on parallel configurations. This is possible since the answers are constructed independently from each other. They also propose an approximate version to generate top-k results with a polynomial delay using an adapted version of [49].

Ranking. They use an *importance-based* edge weight that is a function of the in-degree of its nodes as defined by [15]. They also test using an uniform *compactness-based* scoring, by assigning one to all edges. The final answer score is calculated using a *keyword-pair scoring* considering the edge weight as pairwise node weight.

Evaluation. The different approximate versions of the proposed approach are evaluated against the r-clique and r-clique-rare algorithms [49] over IMDb (MovieLens) and DBLP XML data using the same 5 queries for each dataset. Results show that BKS and BKSR are faster that r-clique and close to r-clique-rare but still does not outperform it for both datasets. However, the parallel setting outperforms all the other algorithms. The other evaluated aspect is the compactness using the average percentage of cliques with the maximum *r*. In contrast to the proposed approach, for r-clique and r-clique-rare not all retrieved results have a maximum distance *r*. To evaluate the quality of the retrieved results, the ground-truth weight is built by selecting the top-50 minimum weight answers generated by the non-approximate version of BKSR and calculating their average weight. Results reveal that BKS and BKSR retrieve results with average uniform weights equal to the ground truth average, in contrast to r-clique and r-clique-rare.

[8] tackles the case where subtrees connecting all the keywords may not exist or they exist, but they are far away from each other resulting in a non-compact answer.

Method. A more relaxed method is proposed that still ensures the compactness of the answers by not requiring that all keywords should be connected. They formulate an optimization problem that aims at finding an optimal answer with a trade-off between compactness (smallest answer diameter) and the degree of relaxation (covering more keywords). For traversing the graph, they use a Best-First strategy (CORE), where the best search direction that may yield a better answer (minimal relaxed) compared to the current one is followed. The exploration terminates if the remaining possible answers cannot be better than the current best answer. The algorithm requires to determine a specific desired answer diameter. Answer ranking was not addressed.

Evaluation. The evaluation was conducted over the RDF versions of Mondial (40 queries from Coffman's benchmark [77]), LinkedMDB[19] (200 random natural language questions transformed to keywords), and DBpedia (438 queries from DBpedia-Entity v2 [78]). The baselines used are: a previous algorithm version of the same authors (CertQR+), and another algorithm for calculating Group Steiner Trees (GST) [79] with edge weights assigned with one. By calculating the compactness of GST answers, they show that it fails finding answers in lot of cases and also finds non-compact answers. In a second step, they show that their relaxed method finds complete answers in case

[18]https://www.ebi.ac.uk/intact/home
[19]https://www.cs.toronto.edu/~oktie/linkedmdb/, linkedmdb-latest-dump.zip

of a number of keywords lower that 10 and 100. The runtime of their approach outperforms CertQR+. Since CORE only computes the top-1 optimal answer, P@1 was used for calculating effectiveness over the DBpedia benchmark ("P@1 = 1 if a computed answer contained a gold-standard answer entity, otherwise P@1 = 0") and compare CORE with CertQR+ and the GST algorithm. Results show that both CORE and CertQR+ have comparable results, and the GST-based approach slightly outperforms both approach with respect to the mean P@1.

[80] proposes a different method that does not rely on traversing the graph.

Method. The very first preprocessing step is to calculate offline the so called *KMV-synopses* over the whole graph, where "A *KMV-synopsis* of a set defines a random sample of size *k*". They are calculated, because they can be used later to estimate the Jaccard similarity of sets. First, keywords are matched to literals' content of the RDF graph. If for one keyword several matches are possible, the one with highest score consisting of the combination of literal and node matching (InfoRank [81]) scores is selected. Then, a so-called initial *query forest* (collection of deconnected trees) is built consisting of keywords and the list of domains and ranges of the properties represented as nodes related to the keyword literals. The goal is to connect it to form a *query tree* (Steiner tree) by applying three types of operations: *node fusion, edge addition, and tree expansion*, before translating it to a SPARQL query. The node fusion step, tries to combine the forest nodes that link to basically similar set of entities. The similarity of those sets is calculated using the Jaccard similarity. If the latter is high, the nodes are combined, which indicates that in the next step the goal is to find entities that have both merged properties that match the initial corresponding keywords. Next, edge addition is applied by identifying object properties that could relate two entities using an estimated set similarity that is computed using the KMV-synopses. The last step is the tree expansion that is a relaxed version of the edge addition, that requires to have either domain or range, not necessary both, to be similar to the other sets. The two last steps are repeated until more connected trees are formed. A cleaning step removes unnecessary edges so that only leaves corresponding to matching nodes are left. Only one tree having the larger score is selected and translated into a *structured query* (SPARQL).

Ranking. The used ranking is both *textual-based* using keyword matching scores and *importance-based* using InfoRank.

*Evaluation.*The evaluation compares the approach with: (1) a schema-based RDF keyword search tool [82] (previous work of some of the same authors), and (2) a virtual document-based approach [83]. For (1), an adapted version of Coffman's benchmark using Mondial and IMDb databases with a total queries of 64 is used. The ground-truth answers were generated using a tool for automatic benchmark construction for keyword search [84]. The effectiveness is measured using the following metrics: Mean Average Precision (MAP) (cf. Subsection 6.1), Top-1, and the MRR. The reached scores are as follows for Mondial and IMDb respectively: MAP of 0.96 and 0.76, Top-1 of 0.96 and 0.76, and an MRR of 0.79 and 0.72, and thus also outperforms the baseline. The average execution time was 11.4 s and 0.60 s for IMDb and Mondial, with no big differences compared to the baseline. For (2), the same benchmark was used as in the virtual document-based baseline, including the following datasets: LUBM[20], BSBM[21], IMDb[22] and DBpedia[23] and also using the same metrics. The proposed approach outperforms the baseline using all metrics.

[85] aims at improving the performance of algorithms that calculate semantically cohesive *minimal trees*. The latter should consist of entities with minimal pairwise semantic distance.

Method. They state that using the semantic distance as an additional factor increases the difficulty of the Steiner tree problem and thus propose two approximation algorithms: quality-oriented and efficiency-oriented. The answer is defined as a *minimal tree* following the same definitions by some of the previous works and the aim is to find the single best answer. Each vertex is assigned with a weight and each pair of vertices is assigned with a semantic distance function. They approximate the problem of computing an optimum answer by first computing a set of relevant paths that lead to each keyword from a specific root node. After that, the candidate paths are transformed into an answer by merging the paths and removing unnecessary edges and vertices to make it a minimal tree.

[20]http://swat.cse.lehigh.edu/projects/lubm/
[21]http://wbsg.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/
[22]https://datasets.imdbws.com/
[23]http://downloads.dbpedia.org/wiki-archive/data-set-37.html

This tree is not necessary optimal but they show that it has a guaranteed approximation ratio. The quality-oriented algorithm finds a minimum cost path for each root, whereas the efficiency-oriented algorithm relaxes this definition by computing the path only for roots that are keywords.

Ranking. The total cost of an answer is calculated using a weighted additive function that aggregates the total PageRank-based vertices weights (*importance-based*) with the total pairwise semantic distance (*semantics-based*) of vertices following a *distinct scoring*. A concrete implementation of the semantic distance is out of scope, but it should be independent from the graph structure.

Evaluation. An evaluation was conducted using three synthetic knowledge graphs generated by the LUBM benchmark and three DBpedia subgraphs including the whole graph. For each LUBM datasets, 250 queries were generated by varying two parameters: the number of keywords and the average number of vertices matched with each keyword. For DBpedia, queries from DBpedia-Entity v2 where filtered by removing keywords without matches. For the evaluation, existing metrics were reused: (1) a PageRank based vertex scoring, and (2) a semantic distance function calculated as the Jaccard distance between the sets of types of entities [86]. The latter requires entities having multiple types, which is not available for LUBM. For that the angular distance between entity embedding vectors using RDF2Vec [87] is calculated (structural semantics). The metrics used are: the runtime compared to BANKS-II [15] and DPBF [51], the ratio between the cost of the approximate and optimum answers (only on small graphs, using an exact version from their algorithm) and the cohesiveness ratio between a baseline answer (BANKS-II and DPBF) and their answer, where cohesiveness is defined as the sum of the pairwise semantic distance. The runtime of the efficiency-oriented algorithm was comparable to BANKS-II. The quality-oriented algorithm showed poor performance by reaching timeouts (200 s) most of the time. The effectiveness was evaluated by conducting a user study using 14-20 queries from the DBpedia-Entity v2 dataset with 16 participants (281 queries). Each participant was presented with the query and corresponding answers retrieved by different methods (DPBF, efficiency-oriented approach) and was asked to rate each answer on a scale (1-4).

[22] does not focus on finding top-k answers but aims at presenting the user with a view that better shows how the answers are correlated and how they may overlap. Furthermore, they also aim at extending the way a user can interact with the retrieved answers by allowing answer manipulation, e.g., intersection or union.

Method. To achieve the previous purpose, they defined a specific answer type. They represent a so called *key-core* which is a subgraph containing highly related answers. Answer ranking was not addressed.

Evaluation. The evaluation is performed using two datasets, DBpedia and DBLP, using 500 randomly selected queries from the datasets' vocabularies. The effectiveness is manually evaluated on 5 example queries. For testing efficiency they compare their approach based on graph traversal against a defined naive baseline (retrieve all key-components and then compute the key-core by union of all key-components) while also varying the graph size, the number of keywords, and other used thresholds like the distance maximal to the keywords.

5.2. *Schema-based methods*

[88] proposes a bidirectional approach for keyword search with defined answer types over RDF graphs with an available ontology.

Method. The proposed system constructs two indexes: the keyword-element index, and an extended version of the single level index by BLINKS that stores for each node the distance of the other node and the direction. The algorithm takes as input a set of nodes corresponding to keywords and a desired answer type (root of interest). The first step is the node merging where it is checked whether each pair of keyword nodes can be directly connected, have same class, or one is a subclass of the other. In the latter cases, the nodes are marked as non-active. The remaining active nodes are bidirectionally expanded to connecting nodes in a round-robin manner until finding a connection with a root. The results are ranked and the best *rooted tree* is transformed to a SPARQL query.

Ranking. The results are ranked by combining *compactness-based*, *textual-based*, and *importance-based* factors based on a previous version of the same authors [89]. The overall tree score is given by the multiplication of three scores each is function of: shortest path between the root and keywords, keyword frequency, and the predicate frequency.

Evaluation. The effectiveness is evaluated with 20 manually created queries and five ontologies. The result of the current approach and its previous version (not bidirectional) are compared with a manually created SPARQL query using the MRR as metric. Results show that the new approach outperforms the old one.

[82] proposes an industry tool to simplify access to specific data from an industry use case.

Method. The overall aim is to construct one SPARQL query to answer a keyword query. The algorithm operated on the data schema where keywords are matched to literals after removing stop words. The matched structure is called *nucleus* and consists of a class, a list of properties, and property values. Each *nucleus* is assigned with a matching score that is used to construct a *minimal tree* that covers all keywords. While a prototypical user interface was proposed, answer ranking was not addressed.

Evaluation. The evaluation was conducted over an industrial database (hydrocarbon exploration) and using Coffman's benchmark (Mondial and IMDb (IMDb-MO)[24]). The relational databases where first converted to triples, then retrieved results using the proposed approach were compared with the ground truth. Results were only analyzed without using any evaluation metric, where 64% (Mondial), 75% (IMDb) of the 50 queries were correctly answered. Using 6 queries for the industrial database, they show that the execution time is reasonable.

[81] focuses on improving the ranking of the results by proposing a new definition of node importance in RDF graphs.

Method. The method aims at generating one structured query (SPARQL) to produce a list of ranked entities. First, the system finds nodes (instances and classes) whose labels match the keywords. Next, for each query keyword the node with higher *accum* score is selected (number of keywords that appear in the label). In the case of equal *accum* scores the disambiguation is performed using another specific node scoring function (*info_score*). The next step is the linking of the classes corresponding to the selected nodes over the graph schema. This consists of finding the *minimal Steiner tree*, but no details are given about how this tree can be found. They only claim that this step will give the information that e.g., two selected nodes are connected through one specific property. This connected tree is transformed to a SPARQL query and the results are ranked based on the sum of the *info_score* of the tree nodes.

Ranking. They use an *importance-based* ranking of entities by proposing a new metric *InfoRank* for scoring instances, classes, and object properties that is based on the *informativeness* of an instance which is defined as the number of its data properties (literals). Each object property is assigned a score that is the maximum of the sum of the *InfoRank* of the subject and object (over all triples involving the property), divided by the sum of the *InfoRank* of all other existing properties. Next, the PageRank of each instance is calculated using the object property score as edge weight. The final score assigned to an instance (*info_score*) is the PageRank after x iterations multiplied by the *InfoRank* of the instance. No ranking of answer trees is proposed.

Evaluation. The proposed ranking score was evaluated by comparing it with the PageRank of some classes, instances, and properties over two datasets[25]: IMDb (IMDb-MO) and MusicBrainz[26] (enriched with DBpedia). This has shown that the *info_score* gives higher scores to the most important classes in both considered domains compared to PageRank. Other experiments were conducted using Coffman's IMDb (50 queries adapted to RDF) and QALD-2[27] MusicBrainz (25 queries) with different ranking measures (e.g., PageRank or HITS [90]). *InfoRank* based ranking achieved the highest MAP.

[91] aims at generating the set of *structured queries* (SPARQL) that answer a keyword query.

Method. Their approach starts with an entity identification module based on the Stanford Regexner Annotator [92] which annotates keywords with candidate classes/individuals from the underlying ontology/RDF repositories. Filter expressions are also detected using a gazetteer. Afterwards, entity combinations are generated, since a keyword can be associated with multiple entities. Less specific annotations (ontology hierarchy) are considered as redundant and thus removed. The answer search algorithm operates on the ontology of the target KG extended with the detected

[24]https://sites.google.com/site/ontopiswc13/home/imdb-mo
[25]https://sites.google.com/view/quira/
[26]http://musicbrainz.org)
[27]https://github.com/ag-sc/QALD

individuals and filters. For each different candidate entity of a specific keyword, such an ontology-based graph model will be built. The graph exploration is done by starting Breadth-First Search from a specific keyword node and the neighbors exploration continues until all candidate keyword elements are included. No cost function is used during the graph exploration. The result is a *rooted tree* that is further cleaned by removing unwanted edges and leaves. Each keyword interpretation results in a corresponding tree. Finally, each tree is translated to a SPARQL query. Only one representative is selected in case of redundant SPARQL queries. Answer scoring is not addressed.

Evaluation. The evaluation was conducted using the QALD-7 dataset for question answering over DBpedia, where each question is associated with a set of keywords. Some types of questions were discarded (e.g., boolean) and the keywords were manually reviewed and adapted. The final test set contained 136 queries. Precision, recall and F1-score are used as metrics to evaluate the generated SPARQL queries against the provided ones. In this case they define different cases (e.g., correct/imprecise/partial answer). The evaluation reveals a precision of 0.52 and a recall of 0.60 considering the best answer for each query. The runtime was also reported with an average time of 57 ms over all queries (without including synonyms from WordNet in the index).

5.3. Summary-based methods

[18] proposes an approach to deal with large graphs that may not fit into memory.

Method. They represent the graph in an hybrid manner (multi-granular) and aim at retrieving *minimal rooted directed trees*: (1) an in-memory summary graph, where nodes are clustered to create supernodes, and (2) a disk-resident part that contains the nodes appertaining to each cluster, together with their adjacency information. Furthermore, they suggest two alternative algorithms that adapt existing exploration algorithms to work over multi-granular graphs, with the goal to reduce IO calls.

Ranking. The same *importance-based* ranking as in [15] was used.

Evaluation. They implemented their approaches by extending BANKS. They used DBLP (8 queries) and IMDb (4 queries) as evaluation datasets and compared different configurations of the proposed algorithm with a schema-based approach (the Sparse algorithm [40]) with respect to the execution time, recall, and cache misses per query.

[93] concentrates on improving the efficiency for complex queries where the keywords have a lot of matching nodes in the graph (*frontier*).

Method. They propose a solution to reduce the frontier that aims at performing search only on parts where actually the candidate answers are more likely to reside. The rationale behind their approach is the fact that if we have multiple matching nodes for each keyword, only few of the different keyword nodes will be tightly related to each other. They use a graph index that maps keywords to a set of subgraphs from the whole graph, together with additional indexes such as keyword-vertex and vertex-subgraph indexes. While creating the subgraph index, they set a bound for the size of the extracted subgraphs. In this way, answering a query would mean searching only subgraphs that actually contain all keywords. They consider the data graph as undirected and define possible answers as *distinct rooted trees*. They used a so called Composed Subgraph Search that it performed over a combination of matched subgraphs. The graph exploration algorithm is similar to the backward search.

Ranking. A *compactness-based* ranking is used with the distance as a ranking factor, and the total score of an answer is defined as the sum of the shortest distances between the root and each keyword node (*root-keyword scoring*) without scoring the nodes.

Evaluation. They evaluated effectiveness and efficiency and compared with BLINKS using the DBLP XML datasets with 12 queries. As a metric for calculating effectiveness, they compare the minimum scores and average scores of the top-10, 25, 50 answers found by the baseline and their approach, which shows comparable results. The analysis of the execution time shows that their method is overall faster compared to the baseline.

[7] proposes another paradigm for keyword search over graphs (RDF).

Method. Instead of presenting the user with the top-k final answers that may originate from various queries, top-k possible *structured queries* are returned. Furthermore, the returned answers are not trees, but subgraphs and user keywords could also be mapped to edges. They use a backward-based search algorithm that operates on a summary graph instead of the whole graph to improve efficiency. There is no distinction between incoming and ongoing edges during the exploration and when reaching a node all its neighbors are taken into consideration preventing cyclic expansion. However, the direction of the edges is important while generating the SPARQL queries corresponding to the subgraphs. The summary graph is derived from the data graph by representing each collection of entities by their corresponding class and aggregating all entities without types using a class called *Thing*. The edges between the summary graph nodes mirror the edges between their corresponding instances.

Ranking. The score of an answer subgraph is calculated as the sum of its path scores following the *root-keyword scoring* while aggregating both edge and node weights. The cost of a path is computed as the aggregation of its element costs. Three different ranking function are introduced: (1) a *compactness-based* ranking using the path length that is defined as the number of elements in a path, (2) an *importance-based* ranking using the popularity of path elements (nodes and edges), where node/edge popularity is given as a function of the number of original instances/edges that were clustered into the corresponding class/edge divided by the number of nodes/edges in the summary graph. The latter is defined in a way that elements with higher popularity should have lower cost and thus contribute to answers with higher scores, and (3) *textual-based* ranking using the keyword matching score which is incorporated by dividing one of the previously defined cost functions by a matching score that is either ranging between 0 and 1 if the element is corresponding to a keyword or is set to one otherwise. This should reflect the fact that higher matching scores reduce the path cost. They use a matching score (between keywords and labels in the graph) that combines both syntactic and semantic similarity (e.g., WordNet) and recommend TF-IDF in case of labels with sufficient number of terms.

Evaluation. The approach was evaluated using 30 queries together with information needs for DBLP and 9 for TAP (knowledge graph describing sports, geography, music and other domains). Twelve participants rated the returned structured queries (query is correct if it matches the information need) and the MRR was used as a metric for the assessment of effectiveness. Results reveal that the textual-based ranking function using keyword matching performed best compared to the other two previously proposed functions. They also perform an evaluation of the query processing time comparing it with other approaches (e.g., [15] and other indexing-based methods) together with an investigation of the impact (on the time) of the number of queries to retrieve, and index performance (also using LUBM). Results show that overall, the proposed approach is faster than the other baselines.

[94] proposes a method that takes advantage of the user profile information.

Method. The proposed approach starts by extracting a structural summary graph from the data that consists of classes, relation between them derived from the relations of corresponding entities, and the union of entity labels. First, keywords are matched to corresponding labels in the graph and for each possible matching a *minimal subgraph* that connects corresponding classes is constructed. For each possible matching, only the smallest connecting subgraph is considered. Subgraphs producing empty results are discarded.

Ranking. The returned subgraphs are ranked based on their similarity with available profile graphs. The latter are either added explicitly or implicitly after a search activity. The similarity is a combination of four metrics: concept similarity, relation similarity, entity property similarity, and entity connection similarity.

Evaluation. The evaluation was conducted using 10 queries for each of the Jamendo (music)[28] and DBpedia[29] datasets. The ground truth was constructed by two users that rated every pattern graph based on its relevance to the result subgraph. The metrics used for evaluation are Kendall-Tau [95], P@k and Rank-DCG (cf. Subsection 6.1). Results reveal a Kendall-Tau above 0.5 and a Rank-DCG above 0.6 for most queries. The average P@5 is at least 0.5 for both datasets. In addition, efficiency experiments show that the system responds in a reasonable time that allows user interaction.

[96] proposes an index (BiG-Index) to speed up finding answers for keywords over KGs.

Method. The rational behind the approach is first to replace the labels of instances with their corresponding generalized ontology labels. Afterwards, this generalized graph version is summarized by merging similar subgraphs into one representative. The last two steps are recursively repeated to build a hierarchy of graphs (indexes). Furthermore, they provide details on how to answer a query using the proposed index. Answer ranking was not addressed.

[28]http://dbtune.org/jamendo/
[29]http://downloads.dbpedia.org/wiki-archive/dbpedia-dataset-version-2015-10.html

Evaluation. the performance of some algorithms (BLINKS and r-clique [54]) for keyword search over graphs is compared both with and without using the index. The evaluation is performed using both real and synthetic datasets (YAGO3[30], DBpedia and IMDb with a total of 18 synthetic queries). Since the methodology requires the usage of an ontology, both DBpedia and IMDb were used with an ontology generated from YAGO3. Results reveal a decrease of the runtime by 50.5% and 29.5% for BLINKS and r-clique respectively.

5.4. Virtual document-based methods

[97] proposes a virtual document-based method that tries to deal with efficiency by reducing the search space and enhance effectiveness by using a scoring function that combines both query dependent and independent scores.

Method. The method starts by preparing a virtual document representation for the whole graph. For each node a virtual document is created by concatenating the textual information (title and attributes) of the node itself and its neighbors in a specific distance. First, a two level node filtering is performed to reduce the search space. Nodes whose virtual document contains every query keyword are ranked based on a specific score and the top-k are selected (*selected roots*). For each keyword, the same filtering is performed over the virtual documents of the roots to end up with a list of top-k *selected keyword nodes*. Next, *minimal trees* connecting the selected roots and keyword nodes and constructed and duplicate trees are removed. The answer trees are then transformed to virtual documents by grouping the textual information for each node and then ranked.

Ranking. The ranking used by the selection of nodes and ranking final answers is calculated using a Markov random field model [98] by combining query dependent and independent features. Those features are function of the frequency of a query term in a virtual document, the node importance which is proportional to its degree, and the edge weights that are set to one to favor smaller answers.

Evaluation. The evaluation was performed using Coffman's benchmark where the MAP (top-1000) was compared with four other systems (e.g., BANKS). Their approach outperforms all the baselines, and shows a good trade-off between effectiveness and efficiency.

[83] proposes an approach based on virtual documents, since they claim that this is a promising method in terms of efficiency.

Method. They introduce Topological Syntactical Aggregator+BM25 (TSA+BM25 [99]) and Topological Syntactical Aggregator+Virtual Documents Pruning (TSA+VDP). TSA is a very first offline phase where the virtual documents are created by clustering triples with related concepts. In practice, it first builds subgraphs having a specific distance around single topics (classes) together with their corresponding text documents (literals, IRIs, predicate strings). Only predicates with a frequency higher than a threshold are taken into account. In the online phase, an initial list of ranked documents relevant to the user query is retrieved. The next step aims at merging overlapping subgraphs corresponding to the top-k candidate documents. If the intersection of the subgraphs' triples is greater than a threshold they are merged. Next, virtual documents corresponding to the merged graphs are created, a second BM25 ranking is performed, and the relevant subgraphs are returned to the user (TSA+BM25). The next step is VDP, that considers the union of the top-k subgraphs of the last step as a new graph. *Minimal trees* with a defined radius containing all keywords are produced using Breadth-First Search, and non-relevant triples (not containing any keyword) are pruned. The last step, is a final ranking of results.

Ranking. The initial and second rankings are both *textual-based* using BM25. The final ranking combines *compactness-based* and *textual-based* factors using the Markov random field model that considers unigrams and bigrams within the virtual document and the distance of the words from the root in the graph.

Evaluation. The evaluation was conducted using real (LinkedMDB and IMDb with 100 queries, and DBpedia-Entity v1 [100] with 50 queries) and synthetic databases (LUBM with 14 queries and BSBM [101] with 13 queries). They consider only one best answer as ground-truth answer. The latter is the result of a manually created SPARQL query corresponding to the keywords. They propose a new definition of precision, recall and the DCG based on a new metric *Signal-to-Noise Ratio* that gives a score to a returned graph based on the intersection of its triples and

[30]http://www.mpi-inf.mpg.de/yago

the ones from the ground truth. Their approach is compared with other three [97, 102, 103] virtual-document based baselines, using average triple based-DCG (tp-DCG) (cf. Subsection 6.1), recall, P@1, P@5, and runtime over all queries. Overall TSA-based systems outperform the baselines, but all in general have a low precision. Considering the online runtime, the proposed approach was among the fastest systems.

Instead of traversing the underlying graph or generating structured queries, [104] aims at adapting an existing information retrieval system (Elasticsearch[31]) and thereby use traditional document indexing and retrieval.

Method. The textual content of a triple is considered as a virtual document that is indexed and retrieved. Two index versions are evaluated depending on the extent of the stored textual content: (1) a baseline index stores only textual content of the considered triple e.g., text of the URI, and (2) an extended index considering also information on other properties of the triple's resources e.g., rdfs:label.

Ranking They use the *textual-based* ranking functions provided by Elasticsearch to rank triples and derive a ranking for the entities involved in a triple using DCG formula.

*Evaluation.*The DBpedia-Entity v2 was used as a test collection using the nDCG@100 and nDCG@10 as metrics. Its goal is testing the influence of different configurations of Elasticsearch (e.g., query type) together with the index content on the effectiveness of the search. Their system (Elas4RDF) was compared to the unsupervised methods tested in the context of DBpedia-Entity v2 and reaches comparable results when used with the extended index and the BM25 ranking. The average query time was also reported: 0.7 s and 1.6 s for the baseline and the extended index respectively. A user interface for the same system is proposed in [105] with a focus on functionality and usability evaluation.

[106] proposes a pipeline consisting of an offline and online stage. The key contribution is the usage of community detection techniques to create group of entities (subgraphs) belonging to the same topic (e.g., computer security in DBpedia). This is used as index to accelerate answer retrieval.

Method. During the offline phase the two indexes are created: (1) the entity index is built by transforming each entity together with its data properties into a virtual document whose terms are indexed, and (2) the community index which maps each entity to its community of entities. The online phase starts with mapping query keywords to candidate entities using the entity index. For each entity, the common community containing all entities is constructed which is at the end a subgraph of the whole RDF entity graph. Finally, a ranked list of trees connecting all keywords (Steiner tree) is computed. No further details are given on the trees construction algorithm or the specific structural *compactness-based ranking*.

Evaluation. Three aspects were compared with an index-based system (EASE) using 5 queries for each dataset (DBLP, KMap[32]): index building (time and storage), runtime, and the effectiveness. The latter is based on answer completeness with respect to the relations connecting keywords using the answer of a method that directly works over the whole graph (not index-based) as a baseline. Both systems have comparable results.

6. Summary

Table 1 and Table 2 provide an overview of all systems described in the previous sections (sorted in ascending chronological order.). In addition to the aspects introduced in Section 4, the following criteria are also reported: The underlying *data*, the *goal* in terms of returned results, the availability of an *index*, the *search algorithm* used, the type of the *intermediate result*, and the specific *ranking*. Most of the approaches either work over any graph-structured data or over KGs (RDF). Furthermore, some of the works also require the existence of an ontology to answer the query, and only one work operates on attributed graphs. The majority of works are graph-based (22), followed by summary-based (5), schema-based (4), and virtual document-based (4) approaches. Overall the aim is to retrieve the top-k relevant answers mostly ranked in the order of relevance to the query. However, a fraction of works only seek to return one relevant result, or deal with the naive scenario of retrieving all relevant answers.

[31]https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
[32]https://old.datahub.io/dataset/knowledge-map

Even if we notice an inconsistency in the naming of the search algorithms used, works that aim at extracting tree or subgraph-shaped results usually rely on graph traversal algorithms (e.g., backward, breadth-first, depth-first, or random walk). The latter are either adapted to work with a specific index structure [15, 18, 93], or a specific result [20–22] or data graph shape [59, 63]. Some works also exploit approximation algorithms that aim at computing the top-k best solutions of a specific optimization problem [49, 56] or optimized enumeration algorithms [17, 19]. Most of the systems that aim at generating SPARQL queries, construct a graph-shaped intermediate result that is translated to a structured query. The final answer type is in most of the cases a tree (15), followed by structured queries (9), and subgraphs (7). Only two works returns triples and one retrieves entities in addition to triples. A newly defined answer type (key-core) is used by a single system.

Most of the works use a compactness-based ranking (18), followed by importance-based (15), and textual-based (12) ones. However, semantics-based (3), diversity-based (1), and profile-based (1) rankings are only rarely used. Almost half (16) of the works combine at least two ranking methods. The distinct answer scoring is mostly used (7), followed by the root-keyword (6) and the keyword-pair (4) scoring. Here also, we notice that two scoring types maybe combined, when for each graph element a different scoring is used. In addition, the specific answer scoring type may be applied only for either nodes or edges. Furthermore, for each system, we also document the type of index used and the specific rankings (cf. Table 2). An index is usually used to support keyword to graph element matching, or to accelerate the graph traversal by storing a specific set of shortest paths between two nodes. Almost half of the works (14) use a keyword to graph element index (keyword-element).

Table 1

Overview of surveyed methods for keyword search over graph-shaped data. ***?***: not mentioned in the paper, **-**: does not apply, ***Onto***: Ontology, ***Attr***: Attributed, ***E***: Edge, ***N***: Node, ***VD***: Virtual document, ***COMP***: Compactness, ***IMP***: Importance, ***TXT***: Textual, ***DIV***: Diversity, ***SEM***: Semantics, ***PROF***: Profile

System	Data	Search space	Goal	Search algorithm	Intermediate result	Final answer type	Answer ranking	Answer scoring
[14]	Graph	Graph-based	Top-k	Backward	-	Rooted directed tree	IMP	Distinct (E, N)
[15]	Graph	Graph-based	Top-k	Bidirectional	-	Rooted directed tree	IMP	Root-keyword (E), Dictinct (N)
[16]	Graph	Graph-based	Top-k	Index-based Bidirectional	-	Distinct rooted directed tree	COMP, IMP, TXT	Root-keyword (E), Dictinct (N)
[17]	Graph	Graph-based	All	Threaded enumeration	-	Minimal tree	-	-
[18]	Graph	Summary-based	Top-k	Multi-granular graph-based	-	Minimal rooted directed tree	IMP	Root-keyword (E), Distinct (N)
[19]	Graph	Graph-based	Top-k, All	Enumeration (polynomial delay)	-	Minimal rooted directed tree	COMP, IMP, DIV	Distinct (E, N)
[7]	RDF	Summary-based	Top-k	Backward	Subgraph	Structured query	COMP, IMP, TXT	Root-keyword (E, N)
[93]	Graph	Summary-based	Top-k	Composed Subgraph	-	Distinct rooted tree	COMP	Root-keyword (E)
[88]	RDF + Onto	Schema-based	Top-1	Node merging/expansion	Rooted tree	Structured query	COMP, IMP, TXT	-
[42]	Graph	Graph-based	Top-k	Shortest path	-	R-clique	COMP, IMP, TXT	Distinct (E, N)
[44]	RDF	Graph-based	Top-k	Depth-First	-	Triple	COMP, SEM	-
[49]	Graph	Graph-based	Top-k	Lawler's procedure	R-clique	Tree	COMP, IMP	Keyword-pair (N)
[53]	RDF	Graph-based	Top-k	Depth-First	Minimal tree	Structured query	COMP	-
[56]	Graph	Graph-based	Top-k	Threshold algorithm	-	Distinct rooted tree	COMP, TXT	Root-keyword (N)
[97]	Graph	VD-based	Top-k	?	-	Minimal tree	COMP, IMP, TXT	-
[59]	RDF	Graph-based	Top-k	Bipartite graph-based	-	Subgraph	TXT	-
[62]	RDF + Onto	Graph-based	Top-1	-	Subgraph	Structured query	COMP	Keyword-pair (E, N)
[63]	RDF	Graph-based	Top-1	Best-First	Subgraph	Structured query	SEM	Distinct (E)
[67]	RDF + Onto	Graph-based	Top-k	Shortest path	-	Minimal subgraph	COMP, TXT	-
[82]	RDF	Schema-based	Top-1	?	Minimal tree	Structured query	-	-
[94]	RDF	Summary-based	Top-k	?	-	Minimal subgraph	COMP, PROF	-
[68]	RDF	Graph-based	Top-k	Dijkstra	-	Minimal subgraph	COMP, TXT	-
[81]	RDF	Schema-based	Top-1	?	Minimal Tree	Structured query	IMP	-
[21]	Graph	Graph-based	Top-k	Canonical Form-based	-	Minimal rooted undirected tree	-	-
[83]	RDF	VD-based	Top-k	Breadth-First	-	Minimal tree	COMP, TXT	-
[25]	Attr. graph	Graph-based	Top-k	Random walk with restart	-	Subgraph	IMP	Keyword-pair (N)
[20]	Graph	Graph-based	Top-k, All	R-clique finding	-	Minimal covered r-clique	COMP, IMP	Keyword-pair (E)
[104]	RDF	VD-based	All	-	-	Triple, entity	TXT	-
[8]	RDF	Graph-based	Top-1	Best-First	-	Minimal tree	-	-
[96]	RDF + Onto	Summary-based	Top-k	*Test different algorithms*	-	-	-	-
[80]	RDF	Graph-based	Top-k	-	Tree	Structured query	IMP, TXT	-
[91]	RDF + Onto	Schema-based	Top-k	Breadth-First	Subgraph	Structured query	-	-
[106]	RDF	VD-based	Top-k	?	-	Tree	COMP	Distinct (E)
[85]	RDF	Graph-based	Top-1	Shortest path	-	Minimal tree	IMP, SEM	Distinct (E, N)
[22]	Graph	Graph-based	Top-1	Key-core computation	-	Key-core	-	-

Table 2
Overview of surveyed methods for keyword search over graph-shaped data (indexing and ranking). ?: not mentioned in the paper, -: does not apply, *: recommended but not used

System	Index	Ranking
[14]	Keyword-element	Node prestige (in-degree), Edge proximity (in-degree)
[15]	Keyword-element	Node prestige (PageRank), Edge proximity (in-degree)
[16]	Shortest paths	Distance, PageRank*, and TF/IDF*
[17]	?	no
[18]	Keyword-element	Node prestige (PageRank), Edge proximity (in-degree)
[19]	?	Tree height, Node in-degree/out-degree, Redundancy penalty
[7]	Keyword-element, Summary graph	Path length, Popularity, Keyword matching (TF/IDF*)
[93]	Keyword-subgraph, Keyword-element, Element-subgraph	Distance
[88]	Shortest paths (BLINKS) with distance, Keyword-element	Shortest path between root and keyword, Term frequency, Property frequency
[42]	no	Term frequency, PageRank, Node degree, Answer size
[44]	Predicates, Subject/object, Literal, Incoming/outgoing relations, Similarity score	Distance (number of subjects in the path), Semantic similarity (WordNet)
[49]	Shortest paths, Keyword-element	Node degree, Shortest path
[53]	Keyword-element, Shortest paths	Tree diameter
[56]	Keyword-element, Shortest paths with node relevance	Node relevance, Shortest path
[97]	Graph nodes/edges, Node-text , Virtual documents	Term frequency, Distance, Node degree
[59]	no	Term frequency
[62]	?	Shortest path
[63]	no	Triple Assembly Cost
[67]	Keyword-element	Term Frequency, Answer size
[82]	Keyword-element	-
[94]	no	Distance
[68]	no	Distance, Keyword matching score
[81]	no	InfoRank
[21]	no	no
[83]	Subgraph-text, Subject	BM25, Markov random field model
[25]	no	Node score
[20]	no	Node degree, Distance
[104]	Triple-text	BM25, DFR, LM-Dirichlet, LM Jelinek-Mercer
[8]	no	no
[96]	Hierarchy of summary graphs	-
[80]	Keyword-element	Keyword matching, InfoRank
[91]	Keyword-element	no
[106]	Keyword-element, Community	Structural compactness
[85]	?	Semantic distance, PageRank
[22]	no	no

6.1. *Evaluation methods*

In general, two aspects are evaluated: effectiveness and efficiency. While the latter deals with the execution time and memory usage, effectiveness judges the ability of the system to retrieve results that are relevant to the query and requires an underlying dataset, a set of user queries, and corresponding results together with their relevance judgments. We also notice that some systems have used approach-specific aspects and metrics for the evaluation: #nodes explored/touched [15], index performance [7, 16], cache misses [18], repetition rate [42], answer compactness (diameter) [49], or #intermediate results [21].

Table 3 gives an overview of the data and metrics used for evaluating the effectiveness of the surveyed systems (sorted in ascending chronological order). For all the works except [62, 88], at least one of the two aspects is evaluated, mostly the execution time. Furthermore, for one system [17] no evaluation was conducted and the other works with no entries did not evaluate effectiveness.

Queries and datasets. We observe that the number of queries used for evaluation varies between 5 and 467. The number of queries is larger when they are randomly generated or an existing benchmark is used, while we notice only dozens of queries by manual creation. Furthermore, the most used datasets are: DBLP, IMDb, and DBpedia.

Ground truth. For each query, a list of potentially relevant results is needed. This is either manually created by result rating from users or by manually creating a corresponding structured query, or automatically generated (other system, structured query, or a tool for benchmark construction). In general we observe that only 10 systems evaluated their approaches using existing test collections (Coffman, DBpedia-Entity v1/v2, QALD) that provide both queries and relevant results.

Metrics. Most of the metrics used to measure the effectiveness are: *Precision*, *Recall*, $F1$-*measure*, $P@k$, *DCG* and its variants, *AP*, *MAP*, and *MRR*. However, we notice that some single systems used other metrics that were shortly mentioned while summarizing the works in the previous sections (e.g., Rank difference, the fraction of queries with results, Kendall-Tau, or the mean of the user scores). In the following, we briefly define the most popular metrics :

Precision, Recall and F-measure. In general, the precision is defined as the ratio between the number of retrieved elements that are relevant to a specific query and the total number of retrieved elements [107]:

$$P = \frac{|relevant \cap retrieved|}{|retrieved|} \tag{1}$$

Recall is defined as the ratio between the number of retrieved elements that are relevant and the total number of all existing relevant elements [107]:

$$R = \frac{|relevant \cap retrieved|}{|relevant|} \tag{2}$$

These definitions may be slightly adapted by some works depending on the nature of the retrieved answer. For example, [83] defines a triple-based variant of precision and recall to compare a generated and a ground truth subgraph, by defining recall as the ratio between the relevant triples and the total number of triples in the ground truth. Another adaptation is given by [91] where precision and recall are calculated for each generated SPARQL answer. Here, recall is a function of the intersection between the generated and ground truth query.

The $F1$-*measure* is calculated by combining both precision and recall as follows [108]:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{3}$$

The previous metrics are *set-based* measures. Next, we define *rank-based* metrics that also consider the order in which the documents are retrieved.

Precision at k (P@k). A good fraction of works used the precision at a specific rank position k as metric [109]. This is more practical since it does not evaluate all retrieved results, but only the top-k:

$$P@k = \frac{|relevant \cap retrieved@k|}{k} \tag{4}$$

Average Precision (AP). The AP is based on the calculation of precision and recall at every position, and is given by the following formula [110], where n is the number of retrieved elements and rel_k is the relevance of the element at position k (1 or 0):

$$AP = \frac{1}{|relevant|} \sum_{k=1}^{n} P@k \cdot rel_k \tag{5}$$

Mean Average Precision (MAP). The MAP is calculated as the ratio between the sum of average precision scores for each query and the total number of queries (Q) [111]:

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{6}$$

Discounted Cumulative Gain (DCG). This measure also takes into account the case where the relevance judgments are given by a graded scale (instead of a binary one) and considers both the relevance and position [111, 112]. The *DCG* at a rank position *n* is defined as follows:

$$DCG_n = \sum_{i=1}^{n} \frac{rel_i}{\log_2(i+1)} \tag{7}$$

where rel_i is the relevance of the retrieved element at the position *i*.

However, the *DCG* is not normalized which makes it difficult to compare different queries with varying result sizes. To deal with this, the normalized *DCG* is introduced. The normalization ($nDCG_n$) is performed by dividing the *DCG* with an ideal *DCG* ($IDCG_n$). The latter is derived by taking the top-*n* relevance judgments ranked in decreasing order of relevance and calculating the DCG_n of this ideal order. The $nDCG_n$ measures could be averaged over all queries to obtain an average $nDCG_n$ of the system.

One work [94] used an improved variant of the *DCG*, called Rank-DCG [111]. [83] proposes a triple-based *DCG* (tp-DCG) based on a new notion of relevance (graph-based) that is calculated as the fraction of relevant triples in a subgraph-shaped answer.

Mean Reciprocal Rank (MRR). The reciprocal rank aims at evaluating the system with respect to it capability of rapidly finding the first relevant result. The *MRR* is the average of reciprocal ranks over all queries *Q*, where pos_i is the position of the very first relevant result to the query *i* [113]:

$$MRR = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{pos_i} \tag{8}$$

7. Conclusion and future directions

In this survey, we have systematically selected, classified and provided an overview of 35 research papers. We have derived four overall aspects for classifying related works: (1) search space, (2) answer type, (3) answer ranking, and (4) answer scoring. Each of those aspects is further specified by defining different possible aspect types (e.g., compactness-based answer ranking). In the following, we highlight some potential directions for future research using a structure that is based on the typical components of a system for keyword search over graph-shaped data.

7.1. Keyword mapping

Enhancing entity linking. Most of the works attempt to map keywords with all possible graph elements that contain one of the keywords in their textual description (string matching). This approach does not only affect efficiency by inducing a huge number of possible candidate nodes/edges (especially for big graphs) that should be connected in the next step, but also induces a lot of non-relevant results. Future research can focus on leveraging entity linking techniques that drastically reduces the number of candidate graph elements in an early stage.

Handling ambiguous queries. Disambiguation is considered as a subtask in a typical entity linking pipeline [9], where a specific entity is selected based on the context. This is usually challenging, given the short nature of

Table 3

Overview of the data and metrics used for evaluating the effectiveness of surveyed systems ***?***: not mentioned in the paper, *****: no evaluation, **-**: does not apply, ***M***: Manual, ***R***: Random, ***G***: Generated

System	#Queries	Query creation	Ground truth	Dataset	Metrics
[14]	7	M	M	DBLP, IIT Bombay dataset	Rank difference
[15]	5	M	G (SQL)	DBLP, IMDB, US patent database	Precision, Recall
[16]	-	-	-	-	-
[17]*	-	-	-	-	-
[18]	12	M	M	DBLP, IMDB	Recall
[19]	?	?	?	?	Ranking correlation
[7]	39	M	M	DBLP , TAP	MRR
[93]	12	M	?	DBLP	Minimum/Avg. scores of top-10, 25, 50
[88]	20	M	M	OntoLife, RDFResume, DBLP, FOAF, Researcher	MRR
[42]	10	R	M	DBLP, IMDb	DCG
[44]	-	-	-	-	-
[49]	19	M,R	M	DBLP, IMDb, Mondial	P@10, P@2, %ground truth answers
[53]	20	M,R	M	DBpedia	1-match (top-10)
[56]	30	M	M	DBLP, IMDb, Mondial	P@10, P@20
[97]	50	Coffman	Coffman	Mondial, Wikipedia, IMDb	MAP (top-1000)
[59]	10	M	?	DBLP	P@5, AP@5, MRR
[62]	50	M	?	?	Fuzzy precision, Fraction queries with result, F1
[63]	180	QALD-6, Free917	QALD-6, Free917	DBpedia, Freebase	Precision, Recall, F1
[67]	10	R	M	AIFB, The Conference dataset	P@k, nDCG@k (k=10,20,5)
[82]	50	Coffman	Coffman	Mondial, IMDb	-
[94]	20	M	M	Jamendo, DBpedia	Kendall-Tau, P@5, Rank-DCG
[68]	20	M	M	AIFB, DBpedia (subset)	P@10, P@5
[81]	75	Coffman, QALD-2	Coffman, QALD-2	IMDb, MusicBrainz	MAP
[21]	-	-	-	-	-
[83]	177	M, DBpedia-Entity v1 (QALD-2)	M, DBpedia-Entity v1 (QALD-2)	LinkedMDB, IMDB, DBpedia, BSBM, LUBM	Recall, P@1, P@5, tb-DCG
[25]	200	R	-	DBLP, IntAct PPI	Expression level, h-index
[20]	10	[49]	G (system)	IMDb, DBLP	Average answer weight
[104]	467	DBpedia-Entity v2	DBpedia-Entity v2	DBpedia	nDCG@100, nDCG@10
[8]	438	DBpedia-Entity v2	DBpedia-Entity v2	Dbpedia	P@1
[96]	-	-	-	-	-
[80]	191	Coffman, [83]	Coffman, [83]	LUBM, BSBM, IMDb, Mondial, DBpedia	MAP, MRR, Top-1
[91]	136	QALD-7	QALD-7	DBpedia	Precision, Recall, F1
[106]	10	?	G (system)	DBLP, KMap	Relationship completeness
[85]	281	DBpedia-Entity v2	DBpedia-Entity v2	DBpedia	Mean of user scores
[22]	5	R	M	DBPedia, DBLP	-

keywords. The enhancement of entity disambiguation techniques would also help reducing the number of candidates and thus indirectly improving efficiency and effectiveness.

Incorporating user intent. Predicting the user intent can help to better understand the query and thus provides guidance for an accurate mapping of keywords. Keyword search systems that work over KGs can in addition leverage the connected nature of the graphs to support techniques for user intent prediction.

7.2. Answer retrieval

Improving efficiency. As KGs grow in size and complexity, current keyword search algorithms may not be able to scale. To deal with that the following directions could be investigated: distributed indexing, parallel processing, and graph summarization. To deal with that, techniques for graph summarization [114] can be used. The latter should have the potential of reducing the size of the graph while preserving important information. However, only few systems [7, 18, 93, 94, 96] follow a summary-based approach and this without evaluating the quality of the generated summaries. Therefore, there is still room for further investigations to deal with some common summarization challenges e.g., information loss. Other directions are the usage of distributed indexing and parallel processing. The latter is still also not sufficiently studied [26–30].

Leveraging large language models. With the recent emergence of large language models [115], a potential research direction could investigate the ability to automatically generate answers such as structured queries e.g., SPARQL given a keyword query and a target KG.

7.3. Answer ranking

Incorporating semantic information. Existing surveys emphasize the general need of enabling semantic search over graph data [4], and propose to, e.g., leverage ontologies to improve efficiency and effectiveness [6]. Furthermore, most of existing techniques working over KGs still consider only structural or textual metrics to judge the relevance of a result to a certain query. Future works can focus on leveraging the semantic information encapsulated in KGs and propose ranking functions that go beyond structural properties of the graph using e.g., semantic similarity or KG embeddings [116].

7.4. Answer presentation

Existing works usually focus on proposing functioning systems aiming at increasing efficiency and effectiveness. However, one aspect is not sufficiently studied namely finding new and suitable ways to present the results to the end user and improve the browsing experience.

Supporting explainability. The aim is to make the search results more transparent by explaining why a specific relevant result was generated. This will allow a better understanding of the relation between the different pieces of information corresponding to the different keywords. A potential future direction is to think about new functionalities to enable explainable information retrieval over KGs.

Enabling exploratory search. Users are not always familiar with the domain in question and they do not always have a specific goal in mind [117]. Therefore, they usually start with a tentative query and continue exploring to better understand a topic or discover new interesting insights and relations. Future research can focus on providing a range of features to support exploration and creative information-seeking behavior using simple keywords and KGs.

7.5. Evaluating search performance

As already mentioned in Subsection 6.1, only few systems were evaluated using existing test collections. On the other hand, we also notice a lack of standardized benchmarks dedicated for keyword search over KGs (only three). Two of them should be adapted before usage since they were originally created to serve other tasks (question answering) or to work over other data formats (relational databases). Future research can focus on developing established evaluation datasets for keyword search over KGs.

7.6. Other specific cases

The literature review also revealed the existence of some additional niche areas with very few contributions, namely *probabilistic/uncertain graphs*, *distributed graphs*, *incomplete graphs*, or *temporal graph*. Future research can investigate more the application of keyword search over the previously mentioned special graphs.

Acknowledgements

This work has been funded by the German Federal Ministry of the Interior and Community and the Thuringian Ministry of Finance. We thank all members and partners of the involved projects: simpLEX, Canarėno, and KollOM-Fit. We would further like to thank Birgitta König-Ries for the guidance and feedback.

References

[1] M.M. Cantallops, S. Sánchez-Alonso and E. García-Barriocanal, A systematic literature review on Wikidata, *Data Technol. Appl.* **53**(3) (2019), 250–268. doi:10.1108/DTA-12-2018-0110.

[2] G. Agarwal, G. Kabra and K.C.-C. Chang, Towards Rich Query Interpretation: Walking Back and Forth for Mining Query Templates, in *WWW '10*, Association for Computing Machinery, New York, NY, USA, 2010, pp. 1–10–. ISBN 9781605587998. doi:10.1145/1772690.1772692.

[3] J.X. Yu, L. Qin and L. Chang, Keyword search in relational databases: A survey., *IEEE Data Eng. Bull.* (2010).

[4] H. Wang and C.C. Aggarwal, A Survey of Algorithms for Keyword Search on Graph Data, in: *Managing and Mining Graph Data*, C.C. Aggarwal and H. Wang, eds, Advances in Database Systems, Vol. 40, Springer, 2010, pp. 249–273. doi:10.1007/978-1-4419-6045-0_8.

[5] A. Ghanbarpour and H. Naderi, Survey on Ranking Functions in Keyword Search over Graph-Structured Data, *J. Univers. Comput. Sci.* **25**(4) (2019), 361–389. http://www.jucs.org/jucs_25_4/survey_on_ranking_functions.

[6] J. Yang, W. Yao and W. Zhang, Keyword Search on Large Graphs: A Survey, *Data Sci. Eng.* **6**(2) (2021), 142–162. doi:10.1007/s41019-021-00154-4.

[7] T. Tran, H. Wang, S. Rudolph and P. Cimiano, Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data, in: *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, Y.E. Ioannidis, D.L. Lee and R.T. Ng, eds, IEEE Computer Society, 2009, pp. 405–416. doi:10.1109/ICDE.2009.119.

[8] G. Cheng, S. Li, K. Zhang and C. Li, Generating Compact and Relaxable Answers to Keyword Queries over Knowledge Graphs, in: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I*, J.Z. Pan, V.A.M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne and L. Kagal, eds, Lecture Notes in Computer Science, Vol. 12506, Springer, 2020, pp. 110–127. doi:10.1007/978-3-030-62419-4_7.

[9] K. Balog, *Entity Linking*, in: *Entity-Oriented Search*, Springer International Publishing, Cham, 2018, pp. 147–188. ISBN 978-3-319-93935-3. doi:10.1007/978-3-319-93935-3_5.

[10] G. Valiente, *Algorithms on Trees and Graphs: With Python Code*, Texts in Computer Science, Springer International Publishing, 2021. ISBN 9783030818852.

[11] B.A. Kitchenham and S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report, EBSE 2007-001, Keele University and Durham University Joint Report, 2007. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf.

[12] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner and M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, *Journal of Systems and Software* **80**(4) (2007), 571–583, Software Performance. doi:https://doi.org/10.1016/j.jss.2006.07.009.

[13] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler and F. Yergeau, Extensible Markup Language (XML) 1.0 (Fifth Edition). https://www.w3.org/TR/REC-xml/.

[14] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti and S. Sudarshan, Keyword searching and browsing in databases using BANKS, in: *Proceedings 18th International Conference on Data Engineering*, 2002, pp. 431–440. doi:10.1109/ICDE.2002.994756.

[15] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai and H. Karambelkar, Bidirectional Expansion For Keyword Search on Graph Databases, in: *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, K. Böhm, C.S. Jensen, L.M. Haas, M.L. Kersten, P. Larson and B.C. Ooi, eds, ACM, 2005, pp. 505–516. http://www.vldb.org/archives/website/2005/program/paper/wed/p505-kacholia.pdf.

[16] H. He, H. Wang, J. Yang and P.S. Yu, BLINKS: ranked keyword searches on graphs, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, C.Y. Chan, B.C. Ooi and A. Zhou, eds, ACM, 2007, pp. 305–316. doi:10.1145/1247480.1247516.

[17] B. Kimelfeld and Y. Sagiv, Efficiently enumerating results of keyword search over data graphs, *Inf. Syst.* **33**(4–5) (2008), 335–359. doi:10.1016/j.is.2008.01.002.

[18] B.B. Dalvi, M. Kshirsagar and S. Sudarshan, Keyword search on external memory data graphs, *Proc. VLDB Endow.* **1**(1) (2008), 1189–1204. doi:10.14778/1453856.1453982. http://www.vldb.org/pvldb/vol1/1453982.pdf.

[19] K. Golenberg, B. Kimelfeld and Y. Sagiv, Keyword proximity search in complex data graphs, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, J.T. Wang, ed., ACM, 2008, pp. 927–940. doi:10.1145/1376616.1376708.

[20] A. Ghanbarpour, K. Niknafs and H. Naderi, Efficient keyword search over graph-structured data based on minimal covered r-cliques, *Frontiers Inf. Technol. Electron. Eng.* **21**(3) (2020), 448–464. doi:10.1631/FITEE.1800133.

[21] X. Lu, D. Theodoratos and A. Dimitriou, Leveraging Pattern Mining Techniques for Efficient Keyword Search on Data Graphs, in: *Web Information Systems Engineering - WISE 2019 Workshop, Demo, and Tutorial, Hong Kong and Macau, China, January 19-22, 2020, Revised Selected Papers*, L.H. U, J. Yang, Y. Cai, K. Karlapalem, A. Liu and X. Huang, eds, Communications in Computer and Information Science, Vol. 1155, Springer, 2019, pp. 98–114. doi:10.1007/978-981-15-3281-8_10.

[22] Z. Zhang, J.X. Yu, G. Wang, Y. Yuan and L. Chen, Key-core: cohesive keyword subgraph exploration in large graphs, *World Wide Web* **25**(2) (2022), 831–856. doi:10.1007/s11280-021-00926-y.

[23] J. Leskovec, A. Rajaraman and J.D. Ullman, *Mining of Massive Datasets*, 3rd edn, Cambridge University Press, 2020. doi:10.1017/9781108684163.

[24] S.E. Dreyfus and R.A. Wagner, The Steiner problem in graphs, *Networks* **1**(3) (1971), 195–207. doi:10.1002/net.3230010302.

[25] S. Bryson, H. Davoudi, L. Golab, M. Kargar, Y. Lytvyn, P. Mierzejewski, J. Szlichta and M. Zihayat, Robust keyword search in large attributed graphs, *Inf. Retr. J.* **23**(5) (2020), 502–524. doi:10.1007/s10791-020-09379-9.

[26] Y. Yang, D. Agrawal, H.V. Jagadish, A.K.H. Tung and S. Wu, An Efficient Parallel Keyword Search Engine on Knowledge Graphs, in: *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, IEEE, 2019, pp. 338–349. doi:10.1109/ICDE.2019.00038.

[27] J. Guan, J. Wang and L. Yu, Multi-keyword Parallel Search Algorithm for Streaming RDF Data, in: *Big Data - 6th CCF Conference, Big Data 2018, Xi'an, China, October 11-13, 2018, Proceedings*, Z. Xu, X. Gao, Q. Miao, Y. Zhang and J. Bu, eds, Communications in Computer and Information Science, Vol. 945, Springer, 2018, pp. 494–511. doi:10.1007/978-981-13-2922-7_33.

[28] C. Liu, L. Yao, J. Li, R. Zhou and Z. He, Finding smallest k-Compact tree set for keyword queries on graphs using mapreduce, *World Wide Web* **19**(3) (2016), 499–518. doi:10.1007/s11280-015-0337-1.

[29] Y. Hao, H. Cao, Y. Qi, C. Hu, S. Brahma and J. Han, Efficient keyword search on graphs using MapReduce, in: *2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, CA, USA, October 29 - November 1, 2015*, IEEE Computer Society, 2015, pp. 2871–2873. doi:10.1109/BigData.2015.7364106.

[30] R.D. Virgilio and A. Maccioni, Distributed Keyword Search over RDF via MapReduce, in: *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab and A. Tordai, eds, Lecture Notes in Computer Science, Vol. 8465, Springer, 2014, pp. 208–223. doi:10.1007/978-3-319-07443-6_15.

[31] X. Lian, L. Chen and Z. Huang, Keyword Search Over Probabilistic RDF Graphs, *IEEE Trans. Knowl. Data Eng.* **27**(5) (2015), 1246–1260. doi:10.1109/TKDE.2014.2365791.

[32] Z. Liu, C. Wang and Y. Chen, Keyword Search on Temporal Graphs, in: *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, IEEE Computer Society, 2018, pp. 1807–1808. doi:10.1109/ICDE.2018.00261.

[33] K. Gkirtzou, K. Karozos, V. Vassalos and T. Dalamagas, Keywords-To-SPARQL Translation for RDF Data Search and Exploration, in: *Research and Advanced Technology for Digital Libraries - 19th International Conference on Theory and Practice of Digital Libraries, TPDL 2015, Poznań, Poland, September 14-18, 2015. Proceedings*, S. Kapidakis, C. Mazurek and M. Werla, eds, Lecture Notes in Computer Science, Vol. 9316, Springer, 2015, pp. 111–123. doi:10.1007/978-3-319-24592-8_9.

[34] Y. Yuan, X. Lian, L. Chen, J.X. Yu, G. Wang and Y. Sun, Keyword Search over Distributed Graphs with Compressed Signature, *IEEE Trans. Knowl. Data Eng.* **29**(6) (2017), 1212–1225. doi:10.1109/TKDE.2017.2656079.

[35] Y. Hao, X. Cao, Y. Sheng, Y. Fang and W. Wang, KS-GNN: Keywords Search over Incomplete Graphs via Graphs Neural Network, in: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y.N. Dauphin, P. Liang and J.W. Vaughan, eds, 2021, pp. 1700–1712. https://proceedings.neurips.cc/paper/2021/hash/0d7363894acdee742caf7fe4e97c4d49-Abstract.html.

[36] M. Zhong, Y. Wang and Y. Zhu, Coverage-Oriented Diversification of Keyword Search Results on Graphs, in: *Database Systems for Advanced Applications - 23rd International Conference, DASFAA 2018, Gold Coast, QLD, Australia, May 21-24, 2018, Proceedings, Part II*, J. Pei, Y. Manolopoulos, S.W. Sadiq and J. Li, eds, Lecture Notes in Computer Science, Vol. 10828, Springer, 2018, pp. 166–183. doi:10.1007/978-3-319-91458-9_10.

[37] C. Park, Effective keyword search on graph data using limited root redundancy of answer trees, *Int. J. Web Inf. Syst.* **14**(3) (2018), 299–316. doi:10.1108/IJWIS-10-2017-0070.

[38] Y. Wang, M. Zhong, Y. Zhu, X. Li and T. Qian, Diversified Top-k Keyword Query Interpretation on Knowledge Graphs, in: *Web and Big Data - First International Joint Conference, APWeb-WAIM 2017, Beijing, China, July 7-9, 2017, Proceedings, Part I*, L. Chen, C.S. Jensen, C. Shahabi, X. Yang and X. Lian, eds, Lecture Notes in Computer Science, Vol. 10366, Springer, 2017, pp. 541–555. doi:10.1007/978-3-319-63579-8_41.

[39] N. Bikakis, G. Giannopoulos, J. Liagouris, D. Skoutas, T. Dalamagas and T.K. Sellis, RDivF: Diversifying Keyword Search on RDF Graphs, in: *Research and Advanced Technology for Digital Libraries - International Conference on Theory and Practice of Digital Libraries, TPDL 2013, Valletta, Malta, September 22-26, 2013. Proceedings*, T. Aalberg, C. Papatheodorou, M. Dobreva, G. Tsakonas and C.J. Farrugia, eds, Lecture Notes in Computer Science, Vol. 8092, Springer, 2013, pp. 413–416. doi:10.1007/978-3-642-40501-3_49.

[40] V. Hristidis, L. Gravano and Y. Papakonstantinou, Efficient IR-Style Keyword Search over Relational Databases, in: *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*, J.C. Freytag, P.C. Lockemann, S. Abiteboul, M.J. Carey, P.G. Selinger and A. Heuer, eds, Morgan Kaufmann, 2003, pp. 850–861. doi:10.1016/B978-012722442-8/50080-X.

[41] L. Page, S. Brin, R. Motwani and T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web., Technical Report, 1999-66, Stanford InfoLab, 1999, Previous number = SIDL-WP-1999-0120. http://ilpubs.stanford.edu:8090/422/.

[42] Z. Zhang, D. Xia and X. Xie, Keyword Search on Graphs Based on Content and Structure, in: *Wireless Algorithms, Systems, and Applications - 9th International Conference, WASA 2014, Harbin, China, June 23-25, 2014. Proceedings*, Z. Cai, C. Wang, S. Cheng, H. Wang and H. Gao, eds, Lecture Notes in Computer Science, Vol. 8491, Springer, 2014, pp. 760–772. doi:10.1007/978-3-319-07782-6_68.

[43] M. Kargar, A. An and X. Yu, Duplication free and minimal keyword search in large graphs, *York Univ., Toronto, ON, Canada, Tech. Rep. CSE-2013-02* (2013).

[44] M. Bae, S. Kang and S. Oh, Semantic similarity method for keyword query system on RDF, *Neurocomputing* **146** (2014), 264–275. doi:10.1016/j.neucom.2014.04.062.

[45] F. Lin and K. Sandkuhl, A Survey of Exploiting WordNet in Ontology Matching, in: *Artificial Intelligence in Theory and Practice II, IFIP 20th World Computer Congress, TC 12: IFIP AI 2008 Stream, September 7-10, 2008, Milano, Italy*, M. Bramer, ed., IFIP, Vol. 276, Springer, 2008, pp. 341–350. doi:10.1007/978-0-387-09695-7_33.

[46] C. Fellbaum and A.M. George, *WordNet: an electronic lexical database*, MIT Press, 1998.

[47] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer and C. Bizer, DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195. doi:10.3233/SW-140134.

[48] B. McBride, Jena: Implementing the RDF Model and Syntax Specification, in: *Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001, Hongkong, China, May 1, 2001*, S. Decker, D.A. Fensel, A.P. Sheth and S. Staab, eds, CEUR Workshop Proceedings, Vol. 40, CEUR-WS.org, 2001. http://CEUR-WS.org/Vol-40/mcbride.pdf.

[49] M. Kargar and A. An, Finding top-*k*, *r*-cliques for keyword search from graphs in polynomial delay, *Knowl. Inf. Syst.* **43**(2) (2015), 249–280. doi:10.1007/s10115-014-0736-0.

[50] E.L. Lawler, A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem, *Management Science* **18**(7) (1972), 401–405. doi:10.1287/mnsc.18.7.401.

[51] B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang and X. Lin, Finding Top-k Min-Cost Connected Trees in Databases, in: *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, R. Chirkova, A. Dogac, M.T. Özsu and T.K. Sellis, eds, IEEE Computer Society, 2007, pp. 836–845. doi:10.1109/ICDE.2007.367929.

[52] L. Qin, J.X. Yu, L. Chang and Y. Tao, Querying Communities in Relational Databases, in: *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, Y.E. Ioannidis, D.L. Lee and R.T. Ng, eds, IEEE Computer Society, 2009, pp. 724–735. doi:10.1109/ICDE.2009.67.

[53] Y. Wang, K. Wang, A.W. Fu and R.C. Wong, KeyLabel algorithms for keyword search in large graphs, in: *2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, CA, USA, October 29 - November 1, 2015*, IEEE Computer Society, 2015, pp. 857–864. doi:10.1109/BigData.2015.7363833.

[54] M. Kargar and A. An, Keyword Search in Graphs: Finding r-cliques, *Proc. VLDB Endow.* **4**(10) (2011), 681–692. doi:10.14778/2021017.2021025.

[55] N. Li, X. Yan, Z. Wen and A. Khan, Density index and proximity search in large graphs, in: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, X. Chen, G. Lebanon, H. Wang and M.J. Zaki, eds, ACM, 2012, pp. 235–244. doi:10.1145/2396761.2396794.

[56] C. Park and S. Lim, Efficient processing of keyword queries over graph databases for finding effective answers, *Inf. Process. Manag.* **51**(1) (2015), 42–57. doi:10.1016/j.ipm.2014.08.002.

[57] R. Fagin, A. Lotem and M. Naor, Optimal aggregation algorithms for middleware, *J. Comput. Syst. Sci.* **66**(4) (2003), 614–656. doi:10.1016/S0022-0000(03)00026-6.

[58] U. Güntzer, W. Balke and W. Kießling, Towards Efficient Multi-Feature Queries in Heterogeneous Environments, in: *2001 International Symposium on Information Technology (ITCC 2001), 2-4 April 2001, Las Vegas, NV, USA*, IEEE Computer Society, 2001, pp. 622–628. doi:10.1109/ITCC.2001.918866.

[59] Z. Zheng, Y. Ding, Z. Wang and Z. Wang, A Novel Method of Keyword Query for RDF Data Based on Bipartite Graph, in: *22nd IEEE International Conference on Parallel and Distributed Systems, ICPADS 2016, Wuhan, China, December 13-16, 2016*, IEEE Computer Society, 2016, pp. 466–473. doi:10.1109/ICPADS.2016.0069.

[60] H.-Y. Li and Y.-Z. Qu, KREAG: Keyword query approach over RDF data based on entity-triple association graph, *Jisuanji Xuebao(Chinese Journal of Computers)* **34**(5) (2011), 825–835.

[61] G. Li, B.C. Ooi, J. Feng, J. Wang and L. Zhou, EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, J.T. Wang, ed., ACM, 2008, pp. 903–914. doi:10.1145/1376616.1376706.

[62] M. Kharrat, A. Jedidi and F. Gargouri, SPARQL Query Generation based on RDF Graph, in: *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 1: KDIR, Porto - Portugal, November 9 - 11, 2016*, A.L.N. Fred, J.L.G. Dietz, D. Aveiro, K. Liu, J. Bernardino and J. Filipe, eds, SciTePress, 2016, pp. 450–455. doi:10.5220/0006091904500455.

[63] S. Han, L. Zou, J.X. Yu and D. Zhao, Keyword Search on RDF Graphs - A Query Graph Assembly Approach, in: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, E. Lim, M. Winslett, M. Sanderson, A.W. Fu, J. Sun, J.S. Culpepper, E. Lo, J.C. Ho, D. Donato, R. Agrawal, Y. Zheng, C. Castillo, A. Sun, V.S. Tseng and C. Li, eds, ACM, 2017, pp. 227–236. doi:10.1145/3132847.3132957.

[64] A. Bordes, N. Usunier, A. García-Durán, J. Weston and O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[65] C. Unger, A.N. Ngomo and E. Cabrio, 6th Open Challenge on Question Answering over Linked Data (QALD-6), in: *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, H. Sack, S. Dietze, A. Tordai and C. Lange, eds, Communications in Computer and Information Science, Vol. 641, Springer, 2016, pp. 171–177. doi:10.1007/978-3-319-46565-4_13.

[66] Q. Cai and A. Yates, Large-scale Semantic Parsing via Schema Matching and Lexicon Extension, in: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, The Association for Computer Linguistics, 2013, pp. 423–433. https://aclanthology.org/P13-1042/.

[67] H. Ouksili, Z. Kedad, S. Lopes and S. Nugier, Using Patterns for Keyword Search in RDF Graphs, in: *Proceedings of the Workshops of the EDBT/ICDT 2017 Joint Conference (EDBT/ICDT 2017), Venice, Italy, March 21-24, 2017*, Y.E. Ioannidis, J. Stoyanovich and G. Orsi, eds, CEUR Workshop Proceedings, Vol. 1810, CEUR-WS.org, 2017. http://ceur-ws.org/Vol-1810/GraphQ_paper_08.pdf.

[68] M. Rihany, Z. Kedad and S. Lopes, Keyword Search Over RDF Graphs Using WordNet, in: *Proceedings of the 1st International Conference on Big Data and Cyber-Security Intelligence, BDCSIntell 2018, Hadath, Lebanon, December 13-15, 2018*, M. Hojeij, B. Finance, Y. Taher, K. Zeitouni, R. Haque and M. Dbouk, eds, CEUR Workshop Proceedings, Vol. 2343, CEUR-WS.org, 2018, pp. 75–82. http://ceur-ws.org/Vol-2343/paper15.pdf.

[69] E.W. Dijkstra, A Note on Two Problems in Connexion with Graphs, *Numer. Math.* **1**(1) (1959), 269–271–. doi:10.1007/BF01386390.

[70] M.J. Zaki, Efficiently Mining Frequent Embedded Unordered Trees, *Fundam. Informaticae* **66**(1–2) (2005), 33–52. http://content.iospress.com/articles/fundamenta-informaticae/fi66-1-2-03.

[71] V. Hristidis and Y. Papakonstantinou, DISCOVER: Keyword Search in Relational Databases, in: *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*, Morgan Kaufmann, 2002, pp. 670–681. doi:10.1016/B978-155860869-6/50065-2.

[72] M. Kargar, A. An, N. Cercone, P. Godfrey, J. Szlichta and X. Yu, Meaningful keyword search in relational databases with large and complex schema, in: *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, J. Gehrke, W. Lehner, K. Shim, S.K. Cha and G.M. Lohman, eds, IEEE Computer Society, 2015, pp. 411–422. doi:10.1109/ICDE.2015.7113302.

[73] B. Bahmani, A. Chowdhury and A. Goel, Fast Incremental and Personalized PageRank, *Proc. VLDB Endow.* **4**(3) (2010), 173–184. doi:10.14778/1929861.1929864.

[74] M. Kargar, A. An and X. Yu, Efficient Duplication Free and Minimal Keyword Search in Graphs, *IEEE Trans. Knowl. Data Eng.* **26**(7) (2014), 1657–1669. doi:10.1109/TKDE.2013.85.

[75] C. Bron and J. Kerbosch, Algorithm 457: Finding All Cliques of an Undirected Graph, *Commun. ACM* **16**(9) (1973), 575–577–. doi:10.1145/362342.362367.

[76] E. Tomita, A. Tanaka and H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, *Theoretical Computer Science* **363**(1) (2006), 28–42, Computing and Combinatorics. doi:https://doi.org/10.1016/j.tcs.2006.06.015.

[77] J. Coffman and A.C. Weaver, An Empirical Performance Evaluation of Relational Keyword Search Techniques, *IEEE Trans. Knowl. Data Eng.* **26**(1) (2014), 30–42. doi:10.1109/TKDE.2012.228.

[78] F. Hasibi, F. Nikolaev, C. Xiong, K. Balog, S.E. Bratsberg, A. Kotov and J. Callan, DBpedia-Entity v2: A Test Collection for Entity Search, in: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, N. Kando, T. Sakai, H. Joho, H. Li, A.P. de Vries and R.W. White, eds, ACM, 2017, pp. 1265–1268. doi:10.1145/3077136.3080751.

[79] R. Li, L. Qin, J.X. Yu and R. Mao, Efficient and Progressive Group Steiner Tree Search, in: *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, F. Özcan, G. Koutrika and S. Madden, eds, ACM, 2016, pp. 91–106. doi:10.1145/2882903.2915217.

[80] Y. Izquierdo, G.M. García, E. Menendez, L.A.P.P. Leme, A.B. Neves, M. Lemos, A.C. Finamore, C. Oliveira and M.A. Casanova, Keyword search over schema-less RDF datasets by SPARQL query compilation, *Inf. Syst.* **102** (2021), 101814. doi:10.1016/j.is.2021.101814.

[81] E.S. Menendez, M.A. Casanova, L.A.P.P. Leme and M. Boughanem, Novel Node Importance Measures to Improve Keyword Search over RDF Graphs, in: *Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part II*, S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A.M. Tjoa and I. Khalil, eds, Lecture Notes in Computer Science, Vol. 11707, Springer, 2019, pp. 143–158. doi:10.1007/978-3-030-27618-8_11.

[82] G. García, Y. Izquierdo, E. Menendez, F. Dartayre and M.A. Casanova, RDF Keyword-based Query Technology Meets a Real-World Dataset, in: *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, V. Markl, S. Orlando, B. Mitschang, P. Andritsos, K. Sattler and S. Breß, eds, OpenProceedings.org, 2017, pp. 656–667. doi:10.5441/002/edbt.2017.86.

[83] D. Dosso and G. Silvello, Search Text to Retrieve Graphs: A Scalable RDF Keyword-Based Search System, *IEEE Access* **8** (2020), 14089–14111. doi:10.1109/ACCESS.2020.2966823.

[84] A.B. Neves, L.A.P.P. Leme, Y.T. Izquierdo and M.A. Casanova, Automatic Construction of Benchmarks for RDF Keyword Search Systems Evaluation, in: *Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS 2021, Online Streaming, April 26-28, 2021, Volume 1*, J. Filipe, M. Smialek, A. Brodsky and S. Hammoudi, eds, SCITEPRESS, 2021, pp. 126–137. doi:10.5220/0010519401260137.

[85] Y. Shi, G. Cheng, T. Tran, E. Kharlamov and Y. Shen, Efficient Computation of Semantically Cohesive Subgraphs for Keyword-Based Knowledge Graph Exploration, in: *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang and L. Zia, eds, ACM / IW3C2, 2021, pp. 1410–1421. doi:10.1145/3442381.3449900.

[86] G. Cheng, F. Shao and Y. Qu, An Empirical Evaluation of Techniques for Ranking Semantic Associations, *IEEE Trans. Knowl. Data Eng.* **29**(11) (2017), 2388–2401. doi:10.1109/TKDE.2017.2735970.

[87] P. Ristoski, J. Rosati, T.D. Noia, R.D. Leone and H. Paulheim, RDF2Vec: RDF graph embeddings and their applications, *Semantic Web* **10**(4) (2019), 721–752. doi:10.3233/SW-180317.

[88] S. Sitthisarn, A Semantic Keyword Search Based on the Bidirectional Fix Root Query Graph Construction Algorithm, in: *Semantic Technology - 4th Joint International Conference, JIST 2014, Chiang Mai, Thailand, November 9-11, 2014. Revised Selected Papers*, T. Supnithi, T. Yamaguchi, J.Z. Pan, V. Wuwongse and M. Buranarach, eds, Lecture Notes in Computer Science, Vol. 8943, Springer, 2014, pp. 387–394. doi:10.1007/978-3-319-15615-6_29.

[89] S. Sitthisarn, L. Lau and P.M. Dew, Semantic keyword search for expert witness discovery, in: *2011 International Conference on Semantic Technology and Information Retrieval*, 2011, pp. 18–25. doi:10.1109/STAIR.2011.5995759.

[90] J.M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, *J. ACM* **46**(5) (1999), 604–632–. doi:10.1145/324133.324140.

[91] F.A. Navarro, C. Martínez-Costa and J.T. Fernández-Breis, Semankey: A Semantics-Driven Approach for Querying RDF Repositories Using Keywords, *IEEE Access* **9** (2021), 91282–91302. doi:10.1109/ACCESS.2021.3091413.

[92] C.D. Manning, M. Surdeanu, J. Bauer, J.R. Finkel, S. Bethard and D. McClosky, The Stanford CoreNLP Natural Language Processing Toolkit, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, The Association for Computer Linguistics, 2014, pp. 55–60. doi:10.3115/v1/p14-5010.

[93] M. Zhong and M. Liu, Efficient keyword proximity search using a frontier-reduce strategy based on *d*-distance graph index, in: *International Database Engineering and Applications Symposium (IDEAS 2009), September 16-18, 2009, Cetraro, Calabria, Italy*, B.C. Desai, D. Saccà and S. Greco, eds, ACM International Conference Proceeding Series, ACM, 2009, pp. 206–216. doi:10.1145/1620432.1620453.

[94] S.B. Sinha, X. Lu and D. Theodoratos, Personalized Keyword Search on Large RDF Graphs based on Pattern Graph Similarity, in: *Proceedings of the 22nd International Database Engineering & Applications Symposium, IDEAS 2018, Villa San Giovanni, Italy, June 18-20, 2018*, B.C. Desai, S. Flesca, E. Zumpano, E. Masciari and L. Caroprese, eds, ACM, 2018, pp. 12–21. doi:10.1145/3216122.3216167.

[95] A. Agresti, *Analysis of ordinal categorical data*, Vol. 656, John Wiley & Sons, 2010. doi:10.1002/9780470594001.

[96] J. Jiang, B. Choi, J. Xu and S.S. Bhowmick, A Generic Ontology Framework for Indexing Keyword Search on Massive Graphs, *IEEE Trans. Knowl. Data Eng.* **33**(6) (2021), 2322–2336. doi:10.1109/TKDE.2019.2956535.

[97] Y. Mass and Y. Sagiv, Virtual Documents and Answer Priors in Keyword Search over Data Graphs, in: *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016*, T. Palpanas and K. Stefanidis, eds, CEUR Workshop Proceedings, Vol. 1558, CEUR-WS.org, 2016. http://ceur-ws.org/Vol-1558/paper20.pdf.

[98] D. Metzler and W.B. Croft, A Markov random field model for term dependencies, in: *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, R.A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat and J. Tait, eds, ACM, 2005, pp. 472–479. doi:10.1145/1076034.1076115.

[99] S. Robertson and H. Zaragoza, The Probabilistic Relevance Framework: BM25 and Beyond, *Found. Trends Inf. Retr.* **3**(4) (2009), 333–389–. doi:10.1561/1500000019.

[100] K. Balog and R. Neumayer, A test collection for entity search in DBpedia, in: *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, G.J.F. Jones, P. Sheridan, D. Kelly, M. de Rijke and T. Sakai, eds, ACM, 2013, pp. 737–740. doi:10.1145/2484028.2484165.

[101] C. Bizer and A. Schultz, The Berlin SPARQL Benchmark, *Int. J. Semantic Web Inf. Syst.* **5**(2) (2009), 1–24. doi:10.4018/jswis.2009040101.

[102] S. Elbassuoni and R. Blanco, Keyword search over RDF graphs, in: *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, C. Macdonald, I. Ounis and I. Ruthven, eds, ACM, 2011, pp. 237–242. doi:10.1145/2063576.2063615.

[103] W. Le, F. Li, A. Kementsietsidis and S. Duan, Scalable Keyword Search on Large RDF Data, *IEEE Trans. Knowl. Data Eng.* **26**(11) (2014), 2774–2788. doi:10.1109/TKDE.2014.2302294.

[104] G. Kadilierakis, P. Fafalios, P. Papadakos and Y. Tzitzikas, Keyword Search over RDF Using Document-Centric Information Retrieval Systems, in: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, A. Harth, S. Kirrane, A.N. Ngomo, H. Paulheim, A. Rula, A.L. Gentile, P. Haase and M. Cochez, eds, Lecture Notes in Computer Science, Vol. 12123, Springer, 2020, pp. 121–137. doi:10.1007/978-3-030-49461-2_8.

[105] C. Nikas, G. Kadilierakis, P. Fafalios and Y. Tzitzikas, Keyword Search over RDF: Is a Single Perspective Enough?, *Big Data Cogn. Comput.* **4**(3) (2020), 22. doi:10.3390/bdcc4030022.

[106] H. Zhang, B. Dong, B. Feng and B. Wei, A Keyword Query Approach Based on Community Structure of RDF Entity Graph, in: *IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC 2021, Madrid, Spain, July 12-16, 2021*, IEEE, 2021, pp. 1143–1148. doi:10.1109/COMPSAC51774.2021.00157.

[107] C. Cleverdon, The Cranfield tests on index language devices, in: *Aslib proceedings*, Vol. 19, MCB UP Ltd, 1967, pp. 173–194.

[108] E. Zhang and Y. Zhang, *F-Measure*, in: *Encyclopedia of Database Systems*, L. LIU and M.T. ÖZSU, eds, Springer US, Boston, MA, 2009, pp. 1147–1147. ISBN 978-0-387-39940-9. doi:10.1007/978-0-387-39940-9_483.

[109] H. Schütze, C.D. Manning and P. Raghavan, *Introduction to information retrieval*, Vol. 39, Cambridge University Press Cambridge, 2008.

[110] T.-Y. Liu, Learning to Rank for Information Retrieval, *Foundations and Trends® in Information Retrieval* **3**(3) (2009), 225–331. doi:10.1561/1500000016.

[111] D. Katerenchuk and A. Rosenberg, RankDCG: Rank-Ordering Evaluation Measure, in: *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*, N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk and S. Piperidis, eds, European Language Resources Association (ELRA), 2016.

[112] K. Järvelin and J. Kekäläinen, Cumulated Gain-Based Evaluation of IR Techniques, *ACM Trans. Inf. Syst.* **20**(4) (2002), 422–446–. doi:10.1145/582415.582418.

[113] N. Craswell, *Mean Reciprocal Rank*, in: *Encyclopedia of Database Systems*, L. LIU and M.T. ÖZSU, eds, Springer US, Boston, MA, 2009, pp. 1703–1703. ISBN 978-0-387-39940-9. doi:10.1007/978-0-387-39940-9_488.

[114] Y. Liu, T. Safavi, A. Dighe and D. Koutra, Graph Summarization Methods and Applications: A Survey **51**(3) (2018). doi:10.1145/3186727.

[115] S. Ozdemir, *Quick Start Guide to Large Language Models: Strategies and Best Practices for Using ChatGPT and Other LLMs*, Addison-Wesley Data and Analytics Series, Pearson Education (US), 2023. ISBN 9780138199197.

[116] Q. Wang, Z. Mao, B. Wang and L. Guo, Knowledge Graph Embedding: A Survey of Approaches and Applications, *IEEE Transactions on Knowledge and Data Engineering* **29**(12) (2017), 2724–2743. doi:10.1109/TKDE.2017.2754499.

[117] R.W. White and R.A. Roth, *Exploratory Search: Beyond the Query-Response Paradigm*, Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan & Claypool Publishers, 2009. doi:10.2200/S00174ED1V01Y200901ICR003.

[118] J. Pound, P. Mika and H. Zaragoza, Ad-Hoc Object Retrieval in the Web of Data, in: *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 771–780–. ISBN 9781605587998. doi:10.1145/1772690.1772769.

[119] G. Smits, O. Pivert, H. Jaudoin and F. Paulus, An Autocompletion Mechanism for Enriched Keyword Queries to RDF Data Sources, in: *Flexible Query Answering Systems - 10th International Conference, FQAS 2013, Granada, Spain, September 18-20, 2013. Proceedings*, H.L. Larsen, M.J. Martín-Bautista, M.A. Vila, T. Andreasen and H. Christiansen, eds, Lecture Notes in Computer Science, Vol. 8132, Springer, 2013, pp. 601–612. doi:10.1007/978-3-642-40769-7_52.