Journal of Universal Computer Science, vol. 25, no. 4 (2019), 361-389
submitted: 20/4/18, accepted: 15/4/19, appeared: 28/4/19 © J.UCS

Survey on Ranking Functions in Keyword Search over Graph-Structured Data

Asieh Ghanbarpour
(University of Sistan and Baluchestan, Zahedan, Iran
ghanbarpour@ece.usb.ac.ir)

Hassan Naderi
(Iran University of Science and Technology, Tehran, Iran
naderi@iust.ac.ir)

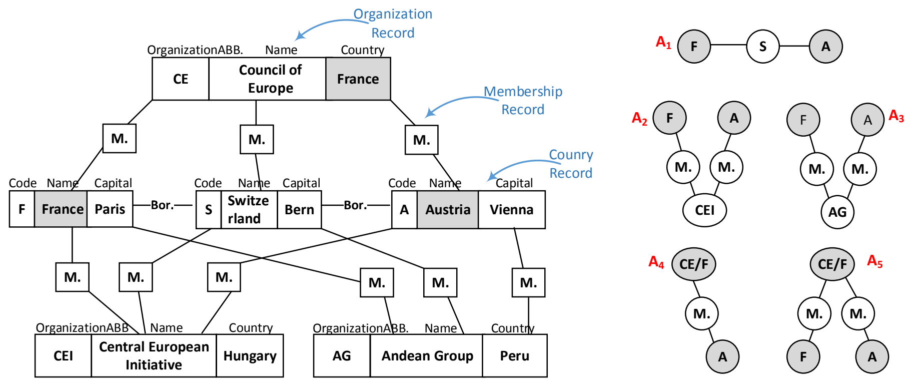
Abstract: Keyword search is known as an attractive alternative for structured query languages in querying over graph-structured data. A keyword query is expressed by a set of keywords and respond by a set of connected structures from the database, which totally or partially cover the queried keywords. These results show how the queried keywords are related in the database. Since there may be numerous results to a given query, a ranking function is essential to present top-k more relevant results to the user. The effectiveness of this function directly affected the effectiveness of the keyword search system. In this paper, we survey the proposed ranking functions in the context of keyword search. First, the proposed models for the results of a keyword query are discussed and a categorization of them is presented. Next, the effective factors in determining the relevance of results are examined. Then, various ranking functions for ordering the results of a query are described and categorized based on their main view in determining the semantic of the results. Finally, we present an analysis of these classes and discuss the evolution of new research strategies to resolve the issues associated with the ranking of results in the keyword search domain.

Keywords: Information retrieval, Database, Keyword search, Query processing, Ranking.
Categories: H.3.3, M.3, M.4, M.7, H.2.4, H.2.5

1 Introduction

Keyword search has been accepted as a widely used mechanism for querying the text-based systems especially the World Wide Web. The simplicity and effectiveness of this type of search has led to its application in querying over relational databases [Agrawal 2002, Bergamaschi 2013, Kim 2014, Oliveira 2015, Bergamaschi 2016, Bou 2016, Pawar 2016], XML databases [Guo 2003, Chen 2010, Gao 2011, Liu 2011, Nguyen 2012, Le 2015], RDF databases [Tran 2009, Bikakis 2013, Le 2014, Wang 2015, Han 2017], graph databases [Bron 1973, Hao 2015, Mass 2016], and any graph-structured heterogeneous data sources [Nguyen 2012]. Keyword search is considered as a user-friendly alternative for structured and semi-structured query languages (e.g. SQL, XQuery, SPARQL) and is attractive for common users who are not familiar with a complex querying language and have no prior knowledge about the underlying data.

A keyword query is expressed as a set of keywords over a database modeled as a graph with labeled elements. A relevant result to this query is a connected structure of the graph which covers whole or part of the queried keywords. For example, suppose [Fig. 1 (a)] shows a part of Mondial database with some modifications. The nodes of this graph are associated with three types of entities, organization, country, and membership. Assume a user poses a keyword query $Q = \{France, Austria\}$ on the graph to search the relationships among the query's keywords. Some of relevant results to the query is shown in [Fig. 1 (b)]. For example, answer A_1 says that "France and Austria are both on the border with Switzerland", answer A_2 means that "France and Austria are both members of Central European Initiative organization", and answer A_5 means that "Austria is a member of Council of Europe organization which is held in France".



(a) a part of Mondial graph (with some modification)

(b) Some of results to query Q={France, Austria}

Figure 1: An example of a keyword query and its relevant results over Mondial dataset

One of the main challenges in processing a keyword query is to develop an efficient algorithm to retrieve relevant results to a given query. Any keyword query may be associated with numerous of relevant results. The other challenge in keyword search is to estimate the relevance degree of results to the query to present top-k most relevant ones to the user. The first challenge has been extensively studied in the literature [Wang 2010, Yu 2010, Liu 2011, Park 2011]. However, the ranking of results is the problem which has been superficially addressed by the existing works. This problem has usually been handled by focusing on the structural properties of results such as the number of nodes or the total weight of edges. However, this is an oversimplified viewpoint to rank the keyword search results because it ignores the texts stored in the result's nodes, which are explicitly targeted by the queries. A group of works [Hristidis 2003a, Liu 2006, Li 2010, Xu 2013, Oliveira 2015] borrowed the text-dependent ranking factors used in the Web search to rank the keyword search results. However, the results of a keyword search (sets of interconnected nodes) are more complex than the results of a Web search (single nodes). Therefore, ranking them requires more expert techniques than what is used in the ranking of the Web results.

For ranking the results of a keyword query, the main problem is how the textual and structural properties of a result can be employed in combination to determine the relevance of a result to the query. In this paper, we first outline a set of textual and structural factors which can be used in scoring the results (subtree/subgraph) of a keyword query. The textual factors refer to the matches between the textual content of results and the keywords of the query. The structural factors refer to the minimality and compactness of results. A more compact result shows the more closeness of nodes in the result's structure and consequently the more relatedness of the queried keywords in the result. Homogeneous combination of the textual and structural factors in defining a ranking function is the other matter examined in this paper. We present a categorization of the proposed ranking functions in the literature. These functions are grouped based on their main view of determining the relevance of results to the query. We perform an analysis of these groups, examine their strengths and weaknesses and discuss the evolution of new research strategies to resolve the issues associated with the ranking problem in keyword search domain.

The rest of the paper is organized as follows: [Section 2] discusses the basic data models on which the keyword search was defined and introduces the problem of keyword search, along with its fundamental characteristics. [Section 3] describes the range of proposed effective factors in ranking the results of a keyword query, [Section 4] describes and categorizes the existing approaches for ranking the complex results. [Section 5] concludes this survey by comparing the existing approaches and describing future directions for research related to effective result ranking.

2 Preliminaries

In this section, we introduce the background model of data, the basic problem of keyword search and the forms of results which are attended in response to the keyword queries.

2.1 Data Model

In general, keyword search is applicable on the graph in the form of $G = (N, E, M_N, L_N, L_E, \sigma)$, where $N = \{n_1, n_2, \dots, n_{|N|}\}$ is a set of nodes, $E = \{(n_i, n_j): 1 \leq i, j \leq |N|, i \neq j\}$ is a set of edges, M_N is the metadata of nodes, L_N and L_E are the sets of all the unique words contained in the labels of nodes and in the labels of edges respectively, and σ is a weight function which assign a weight to each edge. This type of graph could be used for modeling any types of interconnected data such as XML documents, relational data, and RDF triples. In what follows, the ways of modeling these data as a tree/graph are described.

Relational database: a relational database is modeled as a graph in two ways: in the first way, the tables are treated as the nodes of the graph. If a foreign key in table T_i references table T_j, an edge is created between T_i and T_j. In this graph, any connected subgraph represents a join sequence among tables and the final answers are obtained by executing the extracted SQL statements from these subgraphs. In the second way of modeling the tuples are considered as the nodes and the connections between the tuples through primary/foreign keys are employed as the edges. Although such graphs are usually large, they could be directly used to extract the final results [Wang 2010].

XML documents: an XML document was generally mapped to a labeled and directed tree. In this mapping, any element, attribute, and data value is considered as a node in the XML tree [Guo 2003, Gao 2011, Liu 2011, Nguyen 2012, Zhao 2014, Le 2015]. An element node is the parent of its attributes and immediate sub-elements. The value nodes are the child of the corresponding element or attribute node. In some of the works [2003, Guo 2003], XML documents are modeled as a graph by considering ID/IDREF relationships. In an XML graph, in addition to the parent-child edges, there are referential edges, each of which connects an IDREF node to the corresponding ID node. XML modeling in the form of a graph prevents the storage of redundant information occurred in the tree when the same entity is referred by different nodes of branches.

RDF Database: An RDF database can be viewed as a directed graph with nodes representing the entities (URLs), types and data values. An edge in such graph shows an entity-entity, entity-type or entity-value relationships. The edges are labeled with the relationship between its endpoints.

2.2 Keyword Search Problem

Definition 1 (Keyword search problem) Given graph G, an integer k as the desirable number of results and a keyword query $Q = \{q_1, q_2, \dots, q_{|Q|}\}$ where each q_i is a keyword. It is expected to find top-k relevant results $A_1, A_2, \dots., A_k$ to Q, whose are enumerated in a ranked order.

2.3 Results

There are two defined semantics for interpreting a keyword query, conjunctive with “AND” semantics and disjunctive with “OR” semantics. A relevant result to a keyword search over graph G is defined as a subtree or subgraph of G, which covers all (according to the “AND” semantics) or a part (according to the “OR” semantics) of the queried keywords. A group of works have imposed more constraints on the properties of results and proposed special forms for the results of a keyword query. According to our studies, the proposed forms of results could be examined under three main groups as it is shown in [Fig. 2].

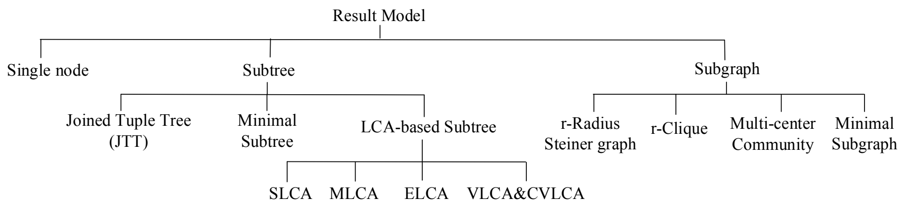


Figure 2: Defined forms of results for a keyword query

A single-node result shows the center of a neighborhood in the graph which discusses the queried keywords. ObjectRank[Balmin 2004] is one of the methods which presents single nodes as the results of a keyword query. This method used the authority transfer paradigm to rank the relevant nodes. Most of works in the literature presented subtrees or subgraphs as the results of a keyword query. These works have usually imposed some constraints on the properties of a result in order to increase the effectiveness of answering. Minimality is one of the most common of these constraints which was defined on both the subtree and subgraph forms of results. A subtree/subgraph is minimal if it does not have a proper subtree/subgraph which still covers the same set of keywords of the query [Mass 2012].

The minimal covered subtrees over relational databases have been referred by different names. The minimal covered subtrees that their nodes show the tuples of the database were named as joined tuple tree (JTT) [Hristidis 2003a, Coffman 2010], minimal joining networks (MJN) [Hristidis 2002] and minimal total joined tuple trees (MTJTT) [Oliveira 2015]. Joined tuple tree is defined in Definition 2. Candidate network (CN) [Agrawal 2002, Hristidis 2002] is the name of minimal covered subtree that its nodes show the relations of the database. These networks are used to instantiate a number of fixed SQL queries whose results are served as the final results of the given keyword query.

Definition 2 (Joined Tuple Tree- JTT): A JTT is a tree of tuples, such that each of its tuples contains at least one of the query keywords, and each pair of its adjacent tuples is connected via a foreign key to a primary key relationship.

The XML data are usually modeled as unweighted trees. An example of an XML tree shows in [Fig. 3]. The common way to find the results of a keyword query over these trees is using the lowest common ancestor (LCA) of matched nodes (the nodes which directly covers at least a query keyword). In what follows, we review the LCA-based concepts including SLCA [Xu 2005], MLCAS [Li 2004], ELCA [Guo 2003], VLCA [Li 2007], and CVLCA [Li 2007]. Any of these concepts determines the root of a relevant result to the query. Before them, we should define the concepts of full coverage and query match.

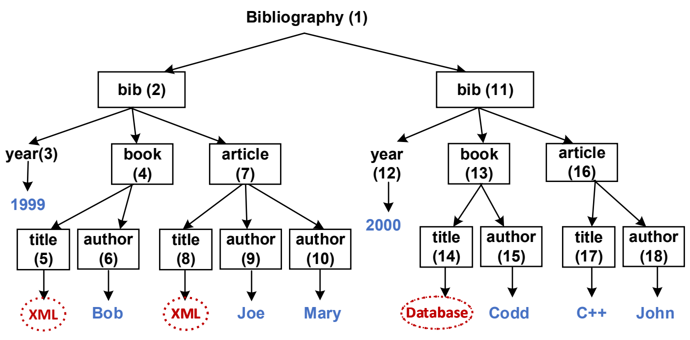


Figure 3: An XML tree [Li 2004]

Definition 3 (Full coverage): an XML node n has full coverage of query Q if there is at least one match to each keywords of Q in the subtree rooted at n.

Definition 4 (Query match): A query match is referred to a set of nodes in the tree consisting of one match to each keyword.

Definition 5 (Smallest Lowest Common Ancestor- SLCA): A node n of an XML tree is an SLCA if it has full coverage of the query and there is no descendant of n that has full coverage of Q.

Although the definition of SLCA is so similar to the definition of minimality in sub-trees, the retrieved results under the SLCA semantics may contain a large number of irrelevant nodes. It is because that any node in the subtree rooted at an SLCA node is considered as a part of the corresponding result. MaxMatch [Liu 2008] is proposed to prune the additional nodes of an SLCA. According to MaxMatch, a node n is pruned if it has a sibling n' which covers in its subtree all of the keywords covered by n. As it is obvious, MaxMatch returns a subset of relevant results in comparison with SLCA. For example, consider query $Q = \{XML,\ 1999\}$. The subtree rooted at bib(2) in [Fig. 3] is considered as a result by SLCA. This result is provided by MaxMatch by pruning the subtree rooted at article(7) because "XML" is also covered by the subtree rooted at book(4).

The concept of MLCAS was proposed based on the relationships among the pairs of nodes as follows:

Definition 6 (Meaningful Lowest Common Ancestor Structure- MLCAS): a node n is considered as a MLCAS if there is a query match P in which all pairs of nodes are meaningfully related, and n is the LCA of query match P. Consider two nodes n_1 and n_2 in P, which cover keywords q_1 and q_2 respectively. These nodes are meaningfully related if there does not exist nodes $n_1{}'$ and $n_2{}'$ covering keywords q_1 and q_2, such that $LCA(n_1{}', n_2{}')$ is a descent of $LCA(n_1, n_2)$.

According to [Liu 2011], the set of retrieved results by MLCAS semantics is a subset of those retrieved by MaxMatch.

The concepts of valuable LCA (VLCA) and compact valuable LCA (CVLCA) were proposed based on the domination of nodes on each other.

Definition 7 (Valuable Lowest Common Ancestor- VLCA): a node n is a VLCA, if there is a query match of which n is the LCA, and each pair of keyword nodes in the query match are interconnected. Two nodes n and n' are interconnected if no nodes on the paths from $LCA(n, n')$ to n and n' have the same label. However, the labels of n and n' could be the same.

Definition 8 (Compact Valuable Lowest Common Ancestor- CVLCA): a node n is considered as a CVLCA, if n is the LCA of a query match P, and it dominates all nodes of P. Node n dominates node n' in P, if the LCA of any other query matches containing n' be an ancestor-or-self of n.

Consider the XML tree shown in [Fig. 3]. Node bib(11) for keyword query $Q =$ "$Database, Author$" is a VLCA, as it is the LCA of the query match $\{Database, Author(18)\}$. However, it is not a CVLCA as it does not dominate node $Database$ because there is another query match $\{Database, Author(15)\}$ whose LCA is a descendant of bib(11).

In [Park 2015], the results covering more number of matched nodes are considered more relevant to query than the minimal results. The ability of generating multi-match results on XML data is provided by the semantics of Exclusive Lowest Common Ancestor (ELCA) in [Guo 2003, Xu 2008, Chen 2010].

Definition 9 (Exclusive Lowest Common Ancestor- ELCA): A node n is considered as ELCA if n has a full coverage of the query even after removing the subtree rooted at each child of n, which has a full coverage of the query.

The set of answers generated by ELCA semantics is a superset of that generated by SLCA semantics. In addition, these answers are better than those generated by SLCA when the data is recursive [Liu 2011]. As an example, consider query "*XML,*

$author$" on the XML tree shown in [Fig. 3]. Node $bib(2)$ is an ELCA because of nodes XML (the child of node 5) and $author(9)$. However, it is not an SLCA since node $article(7)$ is an SLCA.

In the works which present subgraphs as results, how connection of nodes in the result has been usually considered to determine the relevance of the result to the query. The authors in [Kargar 2011, Kargar 2013, Kargar 2014] focused on the high connectivity of nodes to declare a subgraph as a relevant one to the query. This property was embedded in the definition of r-clique as the result of a given query.

Definition 10 (r-clique): an r-clique of graph G with respect to query Q is a set of nodes in G that together cover all the keywords of Q and in which the shortest distance between each pair of nodes is no larger than r.

The r-group Steiner graphs were considered in [Li 2008] as the relevant results to the query. The radius of graph G was defined as the minimal value among the centric distances of every node in G. The centric distance of node v is the maximal value among the distances between v and any node u of G.

Definition 11 (r-Radius Steiner Graph): Given an r-radius graph G and a keyword query Q. Node s in G is called a Steiner node if there exist two matched nodes u and v such that s is on the path between u and v. An r-radius Steiner graph composes of the Steiner nodes, matched nodes and their associated edges. The radius of an r-radius Steiner graph may be smaller than r but cannot be larger than r [Li 2008]. It is obvious that an r-radius Steiner graph is a $2r$-clique. The concept of multi-center community was first introduced in [Qin 2009] with the idea that the minimal connected structures are not always the best results in response to the user queries.

Definition 12 (Multi-Center Community): a multi-center community comprises three types of nodes: the matched nodes which directly contain the query keywords, the center nodes for which there exist at least a single path to every keyword node with the length lower than l, and the path nodes which appears on any path from a center node to a keyword node. Parameter l is a user-defined parameter to control the size of the communities.

3 Ranking Factors

Just as document ranking is a critical component in Web search engines, the ranking of connected structures is a critical component in the keyword search engines. It is due to the numerous relevant results likely to be retrieved for each keyword query. In this section, we review the main ranking factors which have been employed in the literature to rank the results of a keyword query.

3.1 TF/IDF-based Ranking Factors

TF/IDF is one of the main traditional IR measures to rank Web search results, which rates a document based on the occurrences of queried keywords in the document and in the corpus of Web documents. Different versions of TF/IDF have been adapted in the context of keyword search for ranking of results. These adaptations could be categorized into five classes based on the unit (tuple, attribute, resource, node or graph) on which the occurrences of queried keywords are calculated. In what follows, these adaptations are examined.

- **Tuple-level:** In the papers [Liu 2006, Zhou 2007, Cappellari 2012, Xu 2013, Lopez-Veyna 2014] which were defined over relational databases, the text of each tuple was assumed as a document to be used in the keyword weighting. The formula of tuple-based TF/IDF is shown in [Tab. 1].
- **Attribute-level:** When the content of a tuple is considered as a set of (attribute, value) pairs, the value of each attribute could be considered as a document and the frequency of keywords could be calculated over this fine-grained set of documents. TF/IAF (Term Frequency-Inverse attribute Frequency) used in [Calado 2004, Mesquita 2007, Oliveira 2015] is an adapted version of TF/IDF at the attribute level. In this measure, TF shows the frequency of a keyword in the values of an attribute considering all the tuples which have a value for the attribute. Besides, IAF shows how infrequent the keyword is among the values of the attributes. The TF/IAF formulation is shown in [Tab. 1]. The authors of [Hristidis 2003a] also considered the values of attributes as documents. However, they used a measure similar to the basic TF/IDF for weighting keywords in the attribute level. We name this measure TF/IAF2 and show it in [Tab. 1].
- **Resource-level:** When searching over RDF databases, a resource could be considered as a document for calculating a keyword's weight. The concept TF-IRF (Term Frequency-Inverse RDF document Frequency) [Zong 2015] was proposed based on this idea to work over RDF databases. The TF/IRF formulation is shown in [Tab. 1].
- **Node-level:** In [Park 2015], the graph nodes are considered as documents. The relevance of a node n to keyword q contained in the node is estimated using Eq. (1). In this equation, N shows the number of the graph nodes and $N_{q,G}$ shows the number of nodes that contain q.

$$rel_1(n,q) = \sqrt{tf_{q,n}} \cdot \left(1 + log\left(\frac{N}{N_{q,G}+1}\right)\right) \tag{1}$$

  In keyword search over XML documents, the answers are in the form of trees. The works in this domain considered ILF (Inverse Leaf Frequency) instead of IDF to estimate the prevalence of keywords. Using ILF, just the leaf nodes of results are contributed in keyword weighting. The TF/ILF introduced in XSerach [Cohen 2003] is shown in [Tab. 1]. This measure was also used in [Li 2009] in a normalized form to weight keywords.

- **Graph-level**: In Ease [Li 2008] and [Jianhua 2011], the TF/IDF of keywords was measured on the maximal *r*-radius graphs which were retrieved as the results of a keyword query. TF measures the occurrence of keywords in the maximal *r*-radius graphs, and IDF measures how infrequent the keywords are among the maximal r-radius graphs. The formulation of graph-based TF/IDF weighting used in Ease [Li 2008] and [Jianhua 2011] is shown in [Tab. 1]. In XBridge [Li 2010] which was defined over XML databases, the TF/IEF measure was introduced. The IEF (Inverse Element Frequency) of a keyword was defined as the total number of elements of

XML tree divided by the number of elements which cover the keyword in their subtree. The function of TF/IEF weighting is shown in [Tab. 1]. In XBridge, the term frequency of all the keywords was assumed equal to 1.

Unit	**Text-based factor**	**Formula**	**Comments**
Tuple	Tuple-based TF/IDF	$T(q,t) = \frac{1+\ln(1+\ln(tf_{q,t}))}{(1-s)+s\frac{dl_t}{avdl_{t,r(t)}}} \times ln\frac{N_{r(t)}}{df_{q,r(t)}+1}$	$tf_{q,t}$: the term frequency of q in tuple t, $N_{q,r(t)}$: #tuples in relation r(t) containing q, dl_t: the size of tuple t, $N_{r(t)}$: #tuples in r(t), $avdl_{t,r(t)}$: the average size of tuples in r(t), $df_{q,r(t)}$: #tuples in r(t) that contain q, s $(0 \leq s \leq 1)$: a constant.
Attribute	TF/IAF	$T(q,A_i) = TF_{q,A_i} \times IAF_q = \frac{log(1 + tf_{q,A_i})}{log(1 + \|T\|_{A_i})} \times \log\left(1 + \frac{N_A}{N_{q,A}}\right)$	tf_{q,A_i}: the term frequency of q in A_i, $\|T\|_{A_i}$: #distinct terms in A_i, N_A: #attributes in the database, $N_{q,A}$: #attributes in whose keyword q occurs.
	TF/IAF2	$T(q,A_i) = \frac{1+\ln(1+\ln(tf_{q,A_i}))}{(1-s)+s\frac{\|T\|_{A_i}}{avdl_A}} \times \ln\left(\frac{N_{r(A_i)}+1}{N_{q,r(A_i)}}\right)$	$N_{q,r(A_i)}$: #tuples in the relation of A_i, in whose keyword q occurs, $avdl_A$: the average size of attribute-values, $N_{r(A_i)}$: #tuples in the relation of A_i, s $(0 \leq s \leq 1)$: a constant.
Resource	TF/IRF	$T(q,r) = TF_{q,r} \times IRF_q = \frac{tf_{q,r}}{\|T\|_r} \times \frac{N_r}{N_{q,r}}$	$tf_{q,r}$: term frequency of q in RDF doc. r, $\|T\|_r$: #terms in RDF document r, N_r: #RDF documents in the corpus, $N_{q,r}$: #RDF documents that contains q.
Node	TF/ILF	$T(q,n_k) = TF_{q,n_k} \times ILF(k) = \frac{tf_{q,n_k}}{\{\max tf_{q',n_k} \| q' \in n_k\}} \times \log\left(1 + \frac{N_L}{N_{q,L}}\right)$	tf_{q,n_k}: the term frequency of q in node n_k, N_L: # leaf nodes in the XML tree, $N_{q,L}$: #leaf nodes that contains q.
Graph	Graph-based TF/IDF	$T(q,rg) = \frac{1+\ln(1+\ln(tf_{q,rg}))}{(1-s)+s\frac{\|T\|_{rg}}{avdl_{\|T\|}}} \times ln\frac{N_{rrg}+1}{N_{q,rrg}+1}$	$tf_{q,rg}$: the term frequency of q in r-radius graph rg, $avdl_{\|T\|}$: the average #terms among all $rrgs$, $\|T\|_{rg}$: #terms in rg, N_{rrg}: #maximal rrgs, $N_{q,rrg}$: #maximal rrgs that contains q.
	TF/IEF	$T(q,n_k) = \log(1+tf_{q,n_k}) \times \log IEF(n)$	tf_{q,n_k}: the term frequency of q in node n_k, IEF: the inverse element frequency.

Table 1: The adapted versions of TF/IDF used in the context of keyword search

3.2 PageRank-style Ranking Factors

PageRank is an algorithm to measure the relative importance of web pages based on the Web graph. PageRank assigns to a web page a score proportional to the number of times a random surfer would visit the web page if it traversed indefinitely from page to page by randomly following out-links of pages or randomly jumping to other pages. Intuitively, PageRank rates a web page highly if many web pages point to it and many other pages point to those. Although PageRank is proposed for ranking single nodes (web pages), it could be properly adapted for ranking subgraphs. A result (subtree/subgraph) to a keyword query comprises a set of nodes such that the importance of each of which affecting the importance of their group. For example, [Fig. 4] shows some of the results to query $Q = \{\text{Papakonstantinou}, \text{Ullman}\}$ which differ only in the paper node connecting the two authors. Certainly, among these results, the one containing the paper with more citation is more important and should be ranked higher.

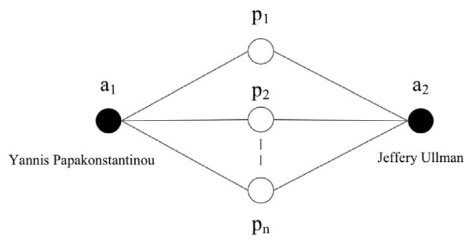


Figure 4: Search results for the query "Papakonstantinou Ullman". Nodes $p_1, \ldots, p_n$ show the common papers written by the two authors [Yu 2017].

In the proposed ranking function of BANKS [Hulgeri 2002], BLINKS[He 2007] and Bidirectional [Kacholia 2005] systems, the use of PageRank values was recommended to consider the individual structural importance of nodes in the ranking of results, although these values were not empirically used by these systems. The works [Xu 2005, Li 2007, Yu 2017] employed modified versions of PageRank by considering both the individual importance of nodes and the cohesiveness of the result nodes. ElemRank is a random walk-based module introduced in XRANK [Guo 2003] to rank XML search results. ElemRank is similar to PageRank, except that it has been formalized on the elements instead of single nodes. In addition, the nested structure of XML has been considered in surfing of the XML tree. After XRANK, the works [Xu 2005, Li 2007] have also employed ElemRank to prepare a ranked list of results.

Inspired by PageRank, CI-Rank [Yu 2017] proposed a variant of the random walk model called Random Walk with Message Passing (RWMP) to assign a score to the nodes of results. One of the objectives of RWMP is to measure the strength of pairwise connection of keyword nodes in a result. In RWMP, a keyword node generates a number of messages as a signal to declare itself. The number of these messages is proportional to the node's importance value. The generated signals are weakened when they pass through intermediate nodes. The strengths of the received signals in the nodes are contributed to estimate the final score of the result.

3.3 Proximity-based Ranking Factors

The use of proximity-based factors in the result ranking rooted at the idea that the more closeness of nodes in the graph shows the more semantic relationship of the nodes. Accordingly, a result with more close nodes is considered more relevant to the given query.

In [Park 2015], the relevance score of a result was measured based on the proximities between the result's nodes and its matched nodes. In this work, the proximity between node n_i and matched node n_j was defined as follows:

$$rel_2(n_i, n_j) = \begin{cases} 1, & n_i = n_j \\ \dfrac{1}{dist(n_i, n_j) + 1}, & n_i \neq n_j \; and \; \exists \; a \; path \; from \; n_i \; to \; n_j \\ 0 & otherwise \end{cases} \tag{2}$$

where $dist(n_i, n_j)$ denotes the length of the shortest path between two nodes n_i and n_j.

The proximity of a result in the SAINT method [Jianhua 2011] was calculated based on the square of the distances between keywords to the result's nodes. Consider node n_i and keyword q_j and let $ku_{(q_j,n_i)}$ is the nearest neighbor of n_i containing q_j. According to the SAINT, the relevance score of node n_i to keyword q_j is calculated using the proximity of nodes as follows:

$$score_{IR}(q_j, n_i) = \frac{T(q_j, ku_{(q_j,n_i)})}{\left(dist\left(n_i, ku_{(q_j,n_i)}\right) + 1\right)^2} \tag{3}$$

The above of fraction in Eq. (3) shows the tuple-based TF/IDF. In SAINT, a proximity function was also used to assign a weight to any pair of keywords. This weight is calculated as follows:

$$Rel(< q_k, q_j > |u) = \sum_{i=1}^{|V|} \frac{1}{\left(dist\left(ku_{(q_k,n_i)}, ku_{(q_j,n_i)}\right) + 1\right)^2} \tag{4}$$

The proximity function in Ease [Li 2008] was defined based on the closeness of matched nodes in the result. In this method, the closeness of two matched nodes was calculated based on all the paths between the nodes as follows:

$$Sim(n_i, n_j) = \sum_{n_i \leftrightsquigarrow n_j} \frac{1}{\left(|n_i \leftrightsquigarrow n_j| + 1\right)^2} \tag{5}$$

where $|n_i \leftrightsquigarrow n_j|$ shows the length of path $n_i \leftrightsquigarrow n_j$ in the corresponding result.
In [Mass 2016], the proximity values were defined under the Gaussian function to specialized the term frequencies. In this method, a node neighborhood was considered

for each node n_i, representing by n_i^*, and a weighted term frequency was assigned to each keyword q_j in n_i^* as follows:

$$wtf\left(q_j, n_i^*\right) = \sum_{u \in n_i^*} e^{\frac{-\left(dist(u,n_i)\right)^2}{2\sigma^2}} \times tf(q_j, u) \tag{6}$$

where σ is a parameter that controls the effect of distance.
In [Ghanbarpour 2018a], a model was estimated for each node of a result and the proximities of the nodes was defined under the Gaussian function. In this method, the calculated proximity values were employed to estimate the effect of a node's content on the models of the other nodes.

3.4 Centrality-based Ranking Factors

The centrality of nodes is one of the structural ranking factors highly considered in the ranking of results. This factor is useful in distinguishing the results which cover equal strengths of keywords and organized under the same structure. Among the different centrality measures, degree-centrality is the most attended one in the keyword search domain due to its simplicity and fast calculations.

In [Mass 2016], the degree of nodes was employed to determine the probability of skeleton forming of a result. In this work, the probability of selecting a node as root is determined proportional to the node's degree in the graph. Besides, the probability of selecting any the other nodes to insert to the result's skeleton is a conditional probability defined based on the node's degree relative to the degrees of its siblings. The joint probability of the result forming is considered as a query-independent score for the result.

3.5 Keyword interdependent-based Ranking Factors

According to observations, humans tend to write queries in which related keywords are close to each other [Kumar 2010]. Therefore, the meaning of a queried keyword could be extracted with the aid of its adjacent keywords in the query. As an example, consider the keyword query "Jack Area Database" expressed on the database shown in [Fig. 5]. In the query, the keyword "Database" is written right next to keyword "Area", showing that the keyword "Database" is more likely relevant to a value of attribute Area in table Person rather than to the table Database.

Person

Name	Area	Email
Aggarwal	Database	aggarwal@aa.cc
Deitel	L.P.	Deitel@mm.cc
Zare	Mineralogy	Zare@dd.nn

Publication

Name	Title	Source
Aggarwal	Graph Databases	DBLP
Zare	Detecting minerals	FAND

Database

Name	Address
DBLP	http://www.informatik.uni-trier.de
FAND	http://www.mineral.kani.cfm
IMDb	http://www.IMDb.com

Figure 5: A part of a database

This intuition was materialized in [Bergamaschi 2011] such that a contextual weight in addition to an intrinsic weight was calculated for each result. Such weighting is to emphasize that the relevance score of a result does not only depend on the isolated score of keywords (intrinsic) but also depend on the co-occurrence of keywords (contextual). In this work, the intrinsic weight of keywords is updated with the values provided by the mappings of the keywords to the database schema terms (the names of tables and attributes). When a keyword is matched to a schema term, the confidence of mapping its adjacent keywords to a value in the schema term's domain is increased. In [Mass 2016] and in the absence of data schema, a bigram language model was used to consider the co-occurrence of keywords. Using this model, a bigram-based weight is assigned to each subsequent of queried keywords $\{q_i, q_{i+1}\}$ in a result. This weight is estimated based on the number of times that the two keywords are observed together in the result. Total of bigram-based weights was considered as a sub-function in the final ranking of results. In [Bergamaschi 2016], the co-occurrence of keywords was considered using mutual information and entropy measures. In this work, the graph nodes represent database terms and the weight of each edge shows the co-occurrence of the two endpoint terms, which is calculated based on the feedback information. The weight of a result is obtained by summing on the weights of its edges. [Tab. 2] shows the set of employed ranking factors used by different keyword search methods to rank results. We organized this table in the order of presenting years of works to illustrate how the maturity of ranking methods in employing the various ranking factors.

4 Ranking Functions

In general, keyword search process could yield more than one result. Keyword search approaches usually use a ranking function to rank the results in decreasing order of their relevance to the query to present top-k of them to the user. We categorize the ranking functions of the literature works into three main groups (as it is shown in [Fig. 6]): structure-based ranking functions which mainly focused on the compactness of results to rank them, text-based ranking functions which were mainly defined based on the textual coverage of results with the ideas derived from the Web search ranking functions, and interaction-based ranking functions which were defined based on the relationship between the nodes of results and their effect on each other. In what follows, we examine the intuition, factors and goals of each group of works. The results of a keyword query may be in the forms of subtree or subgraph. In the following text, R_T is used to show a subtree-form result and R_G is used to show a subgraph-form result. The results are ranked based on a weighting function or a scoring function. The use of these functions are respectively shown by two symbols W and S. When using a weighting function to rank results, the results with smaller weights would be ranked higher and when using a scoring function, the results with greater weights would be ranked higher.

Method	**Database**	**Form of results**	**Ranking function**	**Weight of edges**	**Weight of nodes**	**Cover Window**	**PageRank**	**Size of result**	**Distance**	**TF/IDF**	**|Q|**	**|VD|**	**Co-occurrence**
Discover [Hristidis 2002]	Relational	Tree	Structure-based					*					
Dbxplorer[Agrawal 2002]	Relational	Tree	Structure-based					*					
BANKS [Hulgeri 2002]	Relational	Tree	Structure-based	*	*		P						
XRANK [Guo 2003]	XML	Tree	RW-based		*	*	*	*	*				
XSearch [Cohen 2003]	XML	Tree	Document-based					*		*			
XKeyword [Hristidis 2003b]	XML	Graph	Structure-based						*				
Discover2 [Hristidis 2003a]	Relational	Tree	Document-based					*		*			
ObjectRank [Balmin 2004]	Graph data	Node	RW-based					*		*			
Bidirectional[Kacholia 2005]	Relational	Tree	Structure-based	*	*		*						
Effective [Liu 2006]	Relational	Tree	Document-based					*		*			
Approximate[Kimelfeld 2006]	Graph data	Tree	Structure-based	*									
DPBF [Ding 2007]	Relational	Tree	Structure-based	*									
SPARK [Zhou 2007]	Relational	Tree	Document-based					*		*	*		
BLINKS [He 2007]	Relational	Tree	Structure-based				P		*	P			
Ease [Li 2008]	Graph data	Graph	Document-based						*	*			
SearchWebDB [Tran 2009]	RDF	Graph	Document-based						*	*			
IR-style KS[Ning 2009]	RDF	Tree	Structure-based	*	*								
XBridge [Li 2009]	XML	Tree	Relation-based						*	*	*		
CoverDensity[Coffman 2010]	Relational	Tree	VD-based			*					*	*	
XBridge2 [Li 2010]	XML	Tree	Document-based						*	*			
R-clique [Kargar 2011]	Graph data	Graph	Structure-based						*				
SAINT [Jianhua 2011]	Relational	Tree	Relation-based						*	*			
Metadata[Bergamaschi 2011]	Relational (Schema)	Tree	Relation-based					*		*			*
RG-GRS[Kim 2012]	Graph data	Graph	Relation-based		*					*			
YAANII [Cappellari 2012]	Graph data	Graph	Relation-based						*	*	*	*	
GraphLM[Mass 2012]	Graph data	Graph	VD-based	*	*					*		*	
LP [Xu 2013]	Relational	Tree	Document-based					*		*			
Dup-free [Kargar 2013]	Graph data	Tree	Structure-based						*				
PKI [Yuan 2013]	Uncertain graph data	Tree	Structure-based						*				
SUMM [Le 2014]	RDF	Graph	Structure-based						*				
KESOSASD [Lopez-Veyna 2014]	Relational	Tree	VD-based							*			
SRT-Rank [Kim 2014]	Relational	Tree	Relation-based					*		*			*
BM/EM-RL[Park 2015]	Graph Data	Tree	Relation-based						*	*			
PFC [Zong 2015]	RDF	Graph	Document-based		*		*	*		*	*	*	
EKSG [Hao 2015]	Graph Data	Tree	Structure-based	*									
CNRank [Oliveira 2015]	Relational	Tree	Document-based					*		*		*	
MRF-KS [Mass 2016]	Graph Data	Tree	VD-based	*	*				*	*		*	
QUEST [Bergamaschi 2016]	Relational	Tree	Structure-based	*									*
CI-rank [Yu 2017]	Graph Data	Tree	RW-based	*	*		*		*	*	*		
TKS [Ghanbarpour 2018a]	Graph Data	Graph	Relation-based	*	*		*	*	*	*	*		
				P: it is recommended, but not used.									

Table 2: A fast review of literature works

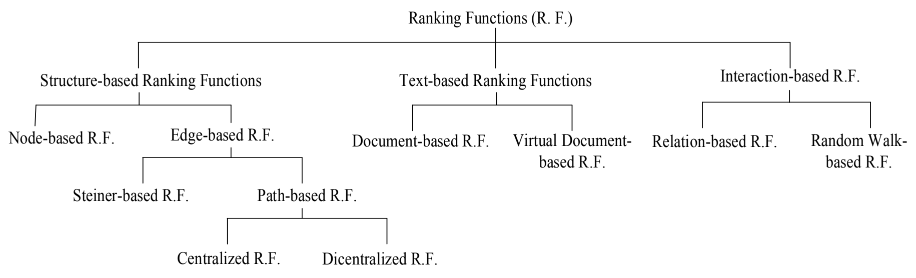


Figure 6: The categorization of ranking functions.

4.1 Structure-based Ranking Functions

The compactness of results is the matter focused by this group of ranking functions. These functions mainly employed the number of nodes and the weight of edges to measure the compactness of results.

4.1.1 Node-based Ranking Functions

Discover [Hristidis 2002] and DBxplorer [Agrawal 2002] simply measured the results compactness based on the size of results (the number of nodes). These works were developed to search over relational databases. They generate results in the form of subtrees in which any node shows a table and any edge represents a join between the corresponding tables. The score of result R_T is determined by reversing its size if it covers all the queried keywords else by zero. The top-ranked subtrees are then mapped to SQL queries to be run on a SQL server and retrieve the final results. The ranking functions in these works were actually designed based on the join cost in the database.

The work [Fakas 2011] was also proposed over relational databases. In this work, the affinity of tuples to the corresponding table, the number of nodes in the result, and the importance of these nodes were contributed in the ranking of results. A result tree R_T is scored using this method as follows:

$$S(R_T, Q) = \frac{\sum_{n_i \in R_T} Im(n_i) \times Af(n_i)}{\log(|R_T|) + 1} \tag{7}$$

where $Im(n_i)$ shows the impotance of node n_i and $Af(n_i)$ shows the affinity of n_i to the corresponding table.

4.1.2 Edge-based Ranking Functions

The ranking functions in [Ding 2007] [Ning 2009], [Hao 2015] [Bergamaschi 2016] were defined in the setting equivalent to the Steiner-tree problem. According to the semantic of Steiner tree, a result is scored by the total edge weight of the result tree; each edge has its weight counted only once. In [Ning 2009], the weight of an edge e_{uv} typed t was defined as $w(e_{uv}) = w(t).\log\big(1 + Degree(u,t)\big)$, where $w(t)$

shows the weight of type t, and $Degree(u, t)$ shows the number of links typed t from u. This work didn't discuss how weighting the types. The weights of edges in [Ding 2007] were determined in the same way as [Ning 2009], except that no type was assumed for the edges. In [Hao 2015], the edges were weighted by a uniform distribution in the range of (0, 1.0]. In QUEST [Bergamaschi 2016], the weight of an edge was defined based on the mutual information and entropy of the endpoint terms to show the co-occurrence of these terms.

Ranking results based on the Steiner weights is in line with ranking based on the node-based weights, but it is more accurate. More precisely, adding a node to a result tree is equivalent to adding an edge to the tree and both lead to increasing the result's weight. However, adding different edges to a result tree increases its Steiner weight differently, while adding a node to the result always increases its node-based weight by one. It means the more distinction power of the Steiner weights than the node-based weights to rank results. However, the insensitivity of the Steiner weighting to how the connectivity of nodes in the result's structure makes this way of weighting also insufficient to rank results. For instance, the Steiner weight of a star-shaped result and a line-shaped result with the same number of nodes is equal if the weights of their edges are taken from the same set.

To score each result in BANKS [Hulgeri 2002], a backward edge $e(v, u)$ is created for each edge $e(u, v)$. The direct edge $e(u, v)$ is assigned a weight of one and its backward edge $e(v, u)$ is weighted proportionally to the number of links to v from the nodes of the same type as u. The weight of each edge is normalized by dividing to the minimum weight of the graph's edges ($w_{\min}$) as $Escore(e) = \log(1 + w(e)/w_{min})$. On the other hand, the prestige of a node v ($N(v)$) is defined proportional to its degree and normalized by the highest degree in the graph as $Nscore(v) = \log(1 + N(v)/N_{max})$. The overall relevance score of a result is calculated in two ways as follows; in both, a factor λ controls the relative effect of the edges and nodes scores.

$$S_1(R_T) = (1 - \lambda)\frac{1}{1 + \sum_{e \in R_T} Escore(e)} + \lambda\frac{\sum_{v \in R_T} Nscore(v)}{|V|} \quad (8)$$

$$S_2(R_T) = \frac{1}{1 + \sum_{e \in R_T} Escore(e)} \times \left(\frac{\sum_{v \in R_T} Nscore(v)}{|V|}\right)^{\lambda} \quad (9)$$

Bidirectional method [Kacholia 2005] used the same function as $Eq. (9)$ to rank results, except that it used a biased version of Pagerank [Brin 2012] to determine the prestige of nodes.

Another group of methods focused on the length of paths in results to score them. These methods could be examined in two groups, the centralized methods and the decentralized ones. In the centralized methods [He 2007, Yuan 2013, Le 2014, Park 2015], a result is weighted by summing on the shortest distances from the root node to the other result's nodes. This way of weighting (named as distinct root-based semantics) is a relaxed form of Steiner-based semantics defined for the tree-shaped results [Yu 2010]. In the decentralized methods, the results are weighted by summing on the shortest distances between different pairs of nodes. In [Hristidis 2003b, Kargar 2011, Kargar 2013], the weight of a result was determined by summing on the shortest distances between the pairs of keyword nodes in the result. In these methods, the weight of each edge was set to one.

4.2 Text-based Ranking Functions

In fact, there is no ranking function in keyword search which exclusively focused on the text of results to score them. However, in this section, we review the ranking functions which their main focus is on the text of results, such that their score to a result could be interpreted as a normalized text score. The main idea of these works is to utilize the successful ranking techniques of Web information retrieval (IR) in the context of keyword search.

4.2.1 Document-based Ranking Functions

In these methods, the nodes of graph are assumed as documents and the keywords are weighted based on their popularity in the documents. TF/IDF and its variants are the main measures employed in these methods for weighting keywords. Assume that $TScore(R, Q)$ shows the TF/IDF-based weight of result R and is defined as follows:

$$TScore(R, Q) = \sum_{q_i \in Q} \sum_{D \in R} T(q_i, D) \tag{10}$$

where D shows a document in the result, on which the TF/IDF scores are calculated and $T(q_i, D)$ shows the TF/IDF weight of keyword q_i in document D, discussed in [Section 3.1].

In [Hristidis 2003a], the results of a keyword query are joining trees of tuples. In this work, the text of any attribute in each tuple was assumed as a document to be used in the calculation of $TScore$ (Eq. 10). Accordingly, result R_T was scored by $TScore(R_T)$ divided by the size of R_T. This score could be interpreted as the result's textual score normalized by the number of nodes. The ranking function in [Xu 2013] is similar to that of [Hristidis 2003a] except that the text of each tuple was assumed as a document to use in the calculation of $TScore$.

The authors of [Liu 2006] believed that using the raw size of results, similar to that used in the two previous methods, can be sub-optimal, especially for the ranking of results involving multiple nodes. For example, consider result R_{T_1} with three nodes each of which covering one of the queried keywords with the weights of w_1. On the other hand, consider answer R_{T_2} with one node covering just one of the queried keywords with the weight of w_2 where $(w_2 > w_1)$. Based on the two previous methods [Hristidis 2003a, Xu 2013], R_{T_2} is ranked incorrectly ahead of R_{T_1}. To solve this problem, the normalized size of results was defined in [Liu 2006] as follows:

$$Nsize(R_T) = (1 - s) + s * \frac{size(R_T)}{avgsize} \tag{11}$$

where $avgsize$ shows the average size of all the retrieved results for the query. This score is then normalized again by dividing to the maximum TF/IDF weight of the keywords covered by the result.

XSearch [Cohen 2003] used the vector space model to calculate the textual relevance of results to the queries of an XML database. In this method, the textual similarity between query Q and result R_T, $Sim(R_T, Q)$, was defined as the sum of the cosine distances between the textual vectors associated with the nodes of R_T and the textual vector associated with the query Q. The overall score of R_T was defined by XSearch as follows:

$$S(R_T, Q) = \frac{Sim(R_T, Q)^{\alpha}}{size(R_T)^{\beta}} \times (1 + \gamma \times anc\_des(R_T)) \tag{12}$$

where $anc\_des(R_T)$ shows the number of unordered pairs of R_T that participate in an ancestor-descendent relationship in the examined XML tree.
In XBridge2 [Li 2010], the textual weight of result was normalized by the distances of its nodes to the root node. The score of result tree R_T was calculated by this method as follows:

$$S(R_T, Q) = \frac{TScore(R_T, Q)}{\sum_{i=1}^{n} (dist^*(r, n_i) - \theta)^{\gamma}} \tag{13}$$

Where θ is the total number of times that the edges repeat on the path from r to each node n_i in the XML tree; γ is a parameter to balance the impact of the structure to the overall score (its default value is 2), and $dist^*(r, n_i)$ is an adapted distance between node n_i and root node r. $TScore$ in this equation is calculated using TF/IEF measure discussed in [Section 3.1].

In [Oliveira 2015], results were ranked based on a probabilistic Bayesian model. In this method, a base vector was considered for each attribute stored in the database to represent its values. The weight of any keyword was determined by TF/IAF weighting discussed in [Section 3.1]. Similarly, a query vector was considered to represent the values of attributes in the query. The overall score of a result was calculated by multiplying the cosine of angles between the result's base vector and the query vector, divided by the size of the result and the number of unique terms in the attributes.

4.2.2 Virtual document-based Ranking Functions

These methods view the graph as a set of overlapping subgraphs. Any of subgraphs is mapped to a virtual document (VD) by concatenating the textual content of its nodes. A relevance score is assigned to each virtual document using the text-based factors (e.g. TF/IDF). The scores of VDs are then combined to determine the final score of result.

In [Lopez-Veyna 2014], the text of a node (tuple) in addition to the texts of its child nodes, the text of its parent, and the texts of its siblings were integrated into a tuple unit (TU). From each tuple unit, several virtual documents were extracted after removing redundant information. Any of virtual documents was indexed as a single document to be used in the search process. In this work, a virtual document covering all of the queried keywords was considered as a relevant result to the query and scored based on summing on the TF/IDF weights of its covered keywords.

In [Mass 2012], the whole of a result was mapped to a virtual document by concatenating the texts of all its nodes. In this work, two fields were imagined for any node: title and content. Accordingly, the constructed VD comprises two fields of title (vd_{tit}) and content (vd_{cnt}). These fields were separately scored using language models as follows:

$$R(Q, vd_f) = \sum_{q_i \in Q} \left(\ln \left((1-\lambda) P(q_i | vd_f) + \lambda P(q_i | G) \right) \right) \tag{14}$$

where f is either content or title and $P(q_i|vd_f)$ is the probability of selecting keyword q_i from vd_f, which is calculated as $P(q_i|vd_f) = tf(q_i, vd_f)/|vd_f|$.

The value of $R(Q, vd_f)$ was then normalized by its maximum value over all of the top-k retrieved results and named as $S_{ir}(Q, vd_f)$. The text-based score of the result (with virtual document vd) was calculated as a linear combination, with parameter α, on the two fields of vd as $S_{ir}(Q, vd) = \alpha \times S_{ir}(Q, vd_{tit}) + (1-\alpha) \times S_{ir}(Q, vd_{cnt})$. This score was then summed up linearly with the structural score of R_G to obtain its overall score. The structural score of a result was calculated by summing on the weights of its nodes and its edges. In this work, a node was weighted according to its degree and an edge was weighted based on the types of its endpoints.

In [Mass 2016], a result R_G also mapped to a virtual document with two fields title and content. The authors of this paper applied a Markov random field (MRF) model to rank results based on their query-dependent and query-independent features. Consider result R_G which was mapped to virtual document vd. The score of R_G was defined using MRF as follows:

$$\begin{aligned} S(R_G, Q) = & \sum_{q_i \in Q} [\lambda_t f_t(q_i, vd) + \lambda_c f_c(q_i, vd)] \\ & + \sum_{\{q_i, q_{i+1}\} \in Q} [\lambda'_t f_t{}'(q_i, q_{i+1}, vd) + \lambda'_c f_c{}'(q_i, q_{i+1}, vd)] \\ & + \lambda_L f_L(R_G) \end{aligned} \tag{15}$$

The parameters λ_t, $\lambda_t{}'$,λ_c, $\lambda_c{}'$and λ_L are nonnegative constants which their sum is 1. The functions $f_t(q_i, vd)$ and $f_c(q_i, vd)$ show the unigrams of title and content of vd. Similarly, $f_t{}'(q_i, q_{i+1}, vd)$ and $f_c{}'(q_i, q_{i+1}, vd)$ show unordered bigrams of title and content of vd. All of these functions are absolutely text-based ones which are calculated based on the frequencies of terms in the fields of vd. $f_L(R_G)$ is a query-independent structural function which was calculated based on the degrees of nodes. According to this function, a result containing nodes with higher degree relative to the degrees of their siblings receives the higher structural score than the other results. This way of thinking is rooted in the underlying PageRank assumption that the more important websites are those that received more links from the other websites. This assumption may be true for the homogenous networks such as Web, but not about heterogeneous ones in which there are nodes of different types. For example, in a relational graph, the degree of intermediary nodes is often very higher than that of entity nodes, while these nodes usually contain no text and only serves as connectors. In such graphs, binding the result's score to the intermediary node seems not logical.

4.3 Interaction-based Ranking Functions

This group of functions scores results based on the distribution of keywords in their nodes and the effects of the nodes on each other. These ranking functions could be grouped into two classes (according to Fig. 6): relation-based ones which were defined based on the pairwise effect of nodes on each other, and random walk-based ranking functions which were defined based on the global effect of nodes on each other in a result.

4.3.1 Relation-based Ranking Functions

In this group of ranking functions, the importance of a keyword in a result is estimated based on its hosting node and the relations of this node to the other nodes of the result.

The authors of [Park 2015] aimed at retrieving results with strong coverage of queried keywords whose are located near to each other. The size of results was not a decisive ranking factor in this method. Accordingly, the relevance score of result R_T rooted at n_r was defined based on the top-p relevant nodes to n_r as follows:

$$rel(R_T, Q) = \sum_{(n_q, q_i) \in Top_p(n_r, Q)} rel_2(n_r, n_q) * rel_1(n_q, q_i) \tag{16}$$

where p is a constant integer greater than or equal to the query's size, and $rel_2(n_r, n_q)$ and $rel_1(n_q, q_i)$ are calculated using Eq. (2) and Eq. (1) respectively. Using this method, the relevance score of a result is calculated by summing on p ($p \geq |Q|$) elements with the highest TF/IDF and the lowest distances to the root node. Therefore, a result with the more number of strong keywords around its root node is ranked higher.

In [Jianhua 2011], the results were ranked based on their structural compactness from the database viewpoint and their textual relevance from the IR viewpoint. Based on this method, the IR-style score of node u with respect to query Q is computed by Eq. (17), where $ku_{(q_i,u)}$ denotes the nearest neighbor of u containing q_i. This score is dependent on the distances of u to its nearest neighbors covering the queried keywords and the strength of keywords in these nodes.

$$score_{IR}(u, Q) = \sum_{q_i \in Q} score_{IR}(u, q_i) = \frac{score_{IR}(ku_{(q_i,u)}, q_i)}{\left(dist(u, ku_{(q_i,u)}) + 1\right)^2} \tag{17}$$

The above of the fraction in Eq. (17) is calculated by Eq. (3).

The DB-point score of node u is computed based on its IR-style relevance score affected by the distances of keywords on each other as follows:

$$score_{DB}(u,Q) = \sum_{1\le i<j\le|Q|} REL(<q_i,q_j>|u) * \left(score_{IR}(u,q_i) + score_{IR}(u,q_j)\right) \quad (18)$$

where $REL(<q_i,q_j>|u)$ is computed by Eq. (4).

According to this method, any graph's node is initially considered as the root of a result and scored by summing up its IR-style score and DB-point score. Top-k nodes with the highest score are selected as the roots of top-k relevant results and ranked accordingly.

In [Ghanbarpour 2018a], the answers were weighted based on the influence of nodes on each other. The Influence of node u on node v was determined based on the content of u, the distance between u and v, and the importance of intermediate nodes located on the shortest path between u and v. This method employed language models to score each node u based on its textual content enriched with the neighborhood information. Using this method, the score of a result was determined by an aggregation on the scores of its nodes.

In [Kim 2012], the results were scored based on their adjacency matrix. To score a result, the eigenvector of its adjacency matrix was normalized by the L_1 norm of this eigenvector. The resulted value was multiplied by a weight vector containing the textual relevancies of the result's nodes to the query. When calculating the eigenvectors, each node affects its neighbors in decreasing order of their distances. Therefore, the presented ranked list of results would be significantly biased toward the star-shaped results.

The ranking function in XBridge [Li 2009] is also a relation-based one defined over the results of an XML database, which are trees covering keywords in their leaves. To score such results, the weight of keywords and the closeness of keyword nodes to the root node were contributed. The weight of a keyword in the root node was computed by dividing the TF/ILF of the keyword in the corresponding leaf node divided by its distance to the root node. The result was scored by summing on the weights of keywords in the root node normalized by the number of queried keywords.

In SRT-Rank [Kim 2014], the semantic relation of nodes was estimated through functional dependencies. This work was defined over relational databases and presented join tuple trees (JTT) as results. The results were ranked based on SRT-scores computed over semantic units named as SRT. An SRT is a maximal subtree of JTT, in which the set of relations has the functional dependency to each other. It means a tuple in one relation uniquely determines tuples in the other relations. Any result was scored in the high-level of scoring based on reversing the number of its SRTs. This work allowed to use any ranking function to rank each group of results with the same SRT-score.

In [Ghanbarpour 2018b], a result was modeled based on a wide range of features including the content of nodes, the proximities of nodes and the importance of both keyword nodes and intermediate nodes. The results of a query are initially ranked based on their models. In the second step, the ranking is modified based on the pseudo-feedback model. In this step, a query model is estimated based on the term locality and importance evidence to assign more weights to terms of relevant results that are in the nodes covering keywords and are more likely to be consistent with the node topic. The results are then ranked again based on the estimated query model.

4.3.2 Random Walk-based Ranking Functions

This group of methods utilized the random walk models to estimate the relevance of results to the query. XRANK [Guo 2003] proposed ElemRank for this purpose. ElemRank is similar to Google's PageRank, but it was calculated at the granularity of an element in the nested structure of the base XML. Suppose result R_T rooted at node r and contains keyword nodes $\{v_1, v_2, \dots, v_{|Q|}\}$ each covering at least a unique keyword of the query. This result is scored based on ElemRank of nodes as follows:

$$S(R_T, Q) = p(r, v_1, v_2, \dots, v_{|Q|}) \times \sum_{i=1}^{|Q|} \left(Elemrank(v_i) \times decay^{dist(v_i, r)+1} \right) \tag{19}$$

where $decay$ is a parameter set to a value in the range of [0,1], and $p(r, v_1, v_2, \dots, v_{|Q|})$ is a measure of keyword proximity that can be any function that ranges from 0 to 1. It should be noted that if some nodes cover the same keyword, a max or sum function is employed for integrating them. The ranking function of XRANK was also employed in [Xu 2005, Li 2007] to rank the result set of queries.
In [Balmin 2004], the score of node n_i with respect to query Q was defined as a combination of two scores: the global ObjectRank of n_i and the keyword-specific ObjectRank of n_i which was calculated based on Q as follows:

$$r^{q_1, \dots, q_{|Q|}}(n_i) = \prod_{i=1,\dots,|Q|} \left(r^{q_k}(n_i) \right)^{g(q_k)} \tag{20}$$

Where $r^{q_k}(n_i)$ is the global ObjectRank of node n_i with respect to keyword q_k, which is calculated based on a similar formula as PageRank one by starting from the keyword nodes containing q_k. In [Balmin 2004], the exponent $g(q_k)$ was set to $1/log(\left|\mathbb{K}_{q_k}\right|)$ to prevent the skewness of ObjectRanks to the high frequent keywords.
CI-Rank [Yu 2017] used a random walk message passing (RWMP) model to score results. In this model, any keyword node v_i generate a number of messages (r_{ii}) proportional to the PageRank value of v_i and the number of keywords which has been covered by this node. The messages of a node stay in the node with probability α and propagate to its neighbors with probability $(1 - \alpha)$. By assuming that a node with the least PageRank value (p_{min}) produces only one message, the survival rate of node v_j is calculated as follows:

$$d_j = 1 - (1 - \alpha)^\wedge(1 + \log\left(\frac{p_j}{p_{min}}\right)) \tag{21}$$

On the other hand, the survival rate of node v_j is equal to $d_j = f_{ij}/r_{ij}$, where r_{ij} shows the number of v_i messages received at node v_j, and f_{ij} shows the number of v_i messages left v_j. The number of v_i messages received in node v_k through node v_j is proportional to the number of v_i messages left v_j and the weight of the link between v_j and v_k. In CI-Rank, the score of node v_i was defined as the minimum value of f_{ij} which received from any keyword node v_i. A result was scored by aggregating the scores of its nodes divided by the number of these nodes. As it is evident, CI-Rank has intelligently employed various factors including the importance of nodes, the weight of edges, the proximity of nodes and the degree of nodes for ranking of results.

5 Discussion

The ranking of results is a challenging problem in the context of keyword search, which has not been discussed well in the literature. The results of a keyword query over a database are connected structures with labeled elements. Structurally processing of these results for the ranking purpose is as important as textually processing of them. The existing structure-based ranking functions such as the functions based on Steiner-based semantics or distinct root-based semantics tried to reflect the size and structure of results in their relevance scores. However, they have not been successful in providing a high-quality ranked list of results. More precisely, the exclusive emphasis of these functions on the structural properties of results made them unable to differentiate among the textually different results which are organized under the same structure. For example, consider the answers shown in [Fig. 7], which have been retrieved for query $Q = \{a, b, c\}$. The subscript of each keyword in this figure shows the term frequency of the keyword in the corresponding node. Two results A_2 and A_3 in this figure receive the same score under the structure-based ranking functions because they are structurally similar. The results of a comprehensive study on the structural factors in [Coffman 2011] also confirm the inadequacy of using structural properties in the ranking of keyword search results.

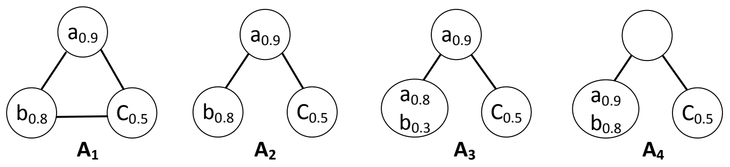


Figure 7: Some sampled results to query $Q = \{a, b, c\}$.

The text-based ranking functions mainly concentrated on the textual properties of results to rank them. Although the effectiveness of such ranking function was proved for keyword search over the Web, they are not successful in the ranking of keyword search results. The relevance score of a result according to these functions is intrinsically a text-based score normalized by the number of textual units contributed in the score calculations. The main weakness of this group of functions is ignoring how the connectivity of nodes in a result. This omission causes that they could not be effective in differentiating among results with different structures, which have the same textual scores and the same number of nodes (or the same number of textual units). For example, results A_1, A_2 and A_4 in [Fig. 7] receive the same score using the text-based ranking functions.

A group of ranking functions were defined based on a combination of structure-based and text-based scores of results. Although a wide range of textual and structural features are employed by these functions to score results, they are unable to distinct results which covers a same set of keywords and organized under the same structure (for example, answers A_2 and A_4 in [Fig. 7]). It is due to ignoring the inter-dependency of text and structure in interpreting the results of a keyword query. According to our studies, the most comprehensive group of ranking functions in the context of keyword search are the interaction-based ones which analyze keywords according to their position in the structure of the result and their distances to the other contained keywords in the result. It should be noted that these functions would be effective if they consider the local importance of keywords, the effect of distance on the relation of keywords, and the environmental factors such as the popularity of nodes in their definitions. The effectiveness of these functions could be improved by applying the semantic extraction tools such as query rewriting and query expansion. It could be also improved by employing the external information such as feedback of users or log stories. These potential issues are useful and interesting to investigate in further researches.

References

[Agrawal, S., Chaudhuri, S. and Das, G. 2002] Agrawal, S., Chaudhuri, S. and Das, G.: "DBXplorer: A System for Keyword-Based Search over Relational Databases"; Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society, 5-16.

[Balmin, A., Hristidis, V. and Papakonstantinou, Y. 2004] Balmin, A., Hristidis, V. and Papakonstantinou, Y.: "Objectrank: authority-based keyword search in databases"; Proceedings of the 30th international conference on Very large data bases, VLDB Endowment/Toronto, Canada, 30, 564-575.

[Bergamaschi, S., Domnori, E., Guerra, F., Lado, R. T. and Velegrakis, Y. 2011] Bergamaschi, S., Domnori, E., Guerra, F., Lado, R. T. and Velegrakis, Y.: "Keyword search over relational databases: a metadata approach"; Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM/Athens, Greece, 565-576.

[Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R. and Velegrakis, Y. 2013] Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R. and Velegrakis, Y.: "QUEST: a keyword search system for relational data based on semantic and machine learning techniques"; Proceedings of the VLDB Endowment, 6, 12 (2013), 1222-1225.

[Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R. and Velegrakis, Y. 2016] Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R. and Velegrakis, Y.: "Combining user and database perspective for solving keyword queries over relational databases"; Information Systems, 55, (2016), 1-19.

[Bikakis, N., Giannopoulos, G., Liagouris, J., Skoutas, D., Dalamagas, T. and Sellis, T. 2013] Bikakis, N., Giannopoulos, G., Liagouris, J., Skoutas, D., Dalamagas, T. and Sellis, T.: "RDivF: Diversifying Keyword Search on RDF Graphs"; 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, Springer Berlin Heidelberg/Berlin, Heidelberg (2013), 413-416.

[Bou, S., Amagasa, T. and Kitagawa, H. 2016] Bou, S., Amagasa, T. and Kitagawa, H.: "An Improved Method of Keyword Search over Relational Data Streams by Aggressive Candidate Network Consolidation"; Proceedings, Part I, 27th International Conference on Database and Expert Systems Applications, Springer-Verlag New York, Inc./Porto, Portugal, 9827, 336-351.

[Brin, S. and Page, L. 2012] Brin, S. and Page, L.: "The anatomy of a large-scale hypertextual web search engine"; Computer Networks, 56, 18 (2012), 3825-3833.

[Bron, C. and Kerbosch, J. 1973] Bron, C. and Kerbosch, J.: "Finding all cliques of an undirected graph"; Communications of the ACM, 16, 9 (1973), 575-577.

[Calado, P., da Silva, A. S., Laender, A. H. F., Ribeiro-Neto, B. A. and Vieira, R. C. 2004] Calado, P., da Silva, A. S., Laender, A. H. F., Ribeiro-Neto, B. A. and Vieira, R. C.: "A Bayesian network approach to searching Web databases through keyword-based queries"; Information Processing & Management, 40, 5 (2004), 773-790.

[Cappellari, P., Virgilio, R. D. and Roantree, M. 2012] Cappellari, P., Virgilio, R. D. and Roantree, M.: "Path-oriented keyword search over graph-modeled Web data"; World Wide Web, 15, 5-6 (2012), 631-661.

[Chen, L. J. and Papakonstantinou, Y. 2010] Chen, L. J. and Papakonstantinou, Y.: "Supporting top-k keyword search in xml databases"; Data Engineering (ICDE), 2010 IEEE 26th International Conference on, IEEE(2010), 689-700.

[Coffman, J. and Weaver, A. C. 2010] Coffman, J. and Weaver, A. C.: "Structured data retrieval using cover density ranking"; Proceedings of the 2nd International Workshop on Keyword Search on Structured Data, ACM/Indianapolis, Indiana, 1-6.

[Coffman, J. and Weaver, A. C. 2011] Coffman, J. and Weaver, A. C.: "Learning to rank results in relational keyword search"; Proceedings of the 20th ACM international conference on Information and knowledge management, ACM/Glasgow, Scotland, UK, 1689-1698.

[Cohen, S., Mamou, J., Kanza, Y. and Sagiv, Y. 2003] Cohen, S., Mamou, J., Kanza, Y. and Sagiv, Y.: "XSEarch: a semantic search engine for XML"; Proceedings of the 29th international conference on Very large data bases, VLDB Endowment/Berlin, Germany, 29, 45-56.

[Ding, B., Yu, J. X., Wang, S., Qin, L., Zhang, X. and Lin, X. 2007] Ding, B., Yu, J. X., Wang, S., Qin, L., Zhang, X. and Lin, X.: "Finding Top-k Min-Cost Connected Trees in Databases"; 23rd International Conference on Data Engineering, IEEE(2007), 836-845.

[Fakas, G. J. 2011] Fakas, G. J.: "A novel keyword search paradigm in relational databases: Object summaries"; Data and Knowledge Engineering, 70, 2 (2011), 208-229.

[Gao, N., Deng, Z.-H., Jiang, J.-J. and YU, H. 2011] Gao, N., Deng, Z.-H., Jiang, J.-J. and YU, H.: "MAXLCA: a new query semantic model for XML keyword search"; Journal of Web Engineering, 11, 2 (2011), 131-145.

[Ghanbarpour, A. and Naderi, H. 2018a] Ghanbarpour, A. and Naderi, H.: "A Model-based Keyword Search Approach for Detecting Top-k Effective Answers"; The Computer Journal, (2018), doi:10.1093/comjnl/bxy056.

[Ghanbarpour, A. and Naderi, H. 2018b] Ghanbarpour, A. and Naderi, H.: "A model-based method to improve the quality of ranking in keyword search systems using pseudo-relevance feedback"; Journal of Information Science, (2018), doi: 10.1177/0165551518799637.

[Guo, L., Shao, F., Botev, C. and Shanmugasundaram, J. 2003] Guo, L., Shao, F., Botev, C. and Shanmugasundaram, J.: "XRANK: ranked keyword search over XML documents"; Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM/San Diego, California, 16-27.

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

[Hao, Y., Cao, H., Qi, Y., Hu, C., Brahma, S. and Han, J. 2015] Hao, Y., Cao, H., Qi, Y., Hu, C., Brahma, S. and Han, J.: "Efficient keyword search on graphs using mapreduce"; Big Data (Big Data), 2015 IEEE International Conference on, IEEE(2015), 2871-2873.

[He, H., Wang, H., Yang, J. and Yu, P. S. 2007] He, H., Wang, H., Yang, J. and Yu, P. S.: "BLINKS: ranked keyword searches on graphs"; Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM/Beijing, China, 305-316.

[Hristidis, V., Gravano, L. and Papakonstantinou, Y. 2003a] Hristidis, V., Gravano, L. and Papakonstantinou, Y.: "Efficient IR-style keyword search over relational databases"; Proceedings of the 29th international conference on Very large data bases, VLDB Endowment/Berlin, Germany, 29, 850-861.

[Hristidis, V. and Papakonstantinou, Y. 2002] Hristidis, V. and Papakonstantinou, Y.: "Discover: keyword search in relational databases"; Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment/Hong Kong, China, 670-681.

[Hristidis, V., Papakonstantinou, Y. and Balmin, A. 2003b] Hristidis, V., Papakonstantinou, Y. and Balmin, A.: "Keyword proximity search on XML graphs"; Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405), (2003), 367-378.

[Hulgeri, A. and Nakhe, C. 2002] Hulgeri, A. and Nakhe, C.: "Keyword Searching and Browsing in Databases using BANKS"; Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society(2002), 431-440.

[Jianhua, F., Guoliang, L. and Jianyong, W. 2011] Jianhua, F., Guoliang, L. and Jianyong, W.: "Finding Top-k Answers in Keyword Search over Relational Databases Using Tuple Units"; IEEE Transactions on Knowledge and Data Engineering, 23, 12 (2011), 1781-1794.

[Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R. and Karambelkar, H. 2005] Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R. and Karambelkar, H.: "Bidirectional expansion for keyword search on graph databases"; Proceedings of the 31st international conference on Very large data bases, VLDB Endowment/Trondheim, Norway, 505-516.

[Kargar, M. and An, A. 2011] Kargar, M. and An, A.: "Keyword search in graphs: finding r-cliques"; Proceedings of the VLDB Endowment, 4, 10 (2011), 681-692.

[Kargar, M. and An, A. 2014] Kargar, M. and An, A.: "Finding top-k, r-cliques for keyword search from graphs in polynomial delay"; Knowledge and Information Systems, 43, 2 (2014), 249-280.

[Kargar, M., An, A. and Yu, X. 2013] Kargar, M., An, A. and Yu, X.: "Efficient Duplication Free and Minimal Keyword Search in Graphs"; IEEE Transactions on Knowledge and Data Engineering, 26, 7 (2013), 1657 - 1669

[Kim, I.-J., Whang, K.-Y. and Kwon, H.-Y. 2014] Kim, I.-J., Whang, K.-Y. and Kwon, H.-Y.: "SRT-rank: Ranking keyword query results in relational databases using the strongly related tree"; IEICE TRANSACTIONS on Information and Systems, 97, 9 (2014), 2398-2414.

[Kim, S., Lee, W., Arora, N. R., Jo, T.-C. and Kang, S.-H. 2012] Kim, S., Lee, W., Arora, N. R., Jo, T.-C. and Kang, S.-H.: "Retrieving keyworded subgraphs with graph ranking score"; Expert Systems with Applications, 39, 5 (2012), 4647-4656.

[Kimelfeld, B. and Sagiv, Y. 2006] Kimelfeld, B. and Sagiv, Y.: "Finding and approximating top-k answers in keyword proximity search"; Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM/Chicago, IL, USA, 173-182.

[Kumar, R. and Tomkins, A. 2010] Kumar, R. and Tomkins, A.: "A characterization of online browsing behavior"; Proceedings of the 19th international conference on World wide web, ACM(2010), 561-570.

[Le, T. N., Bao, Z. and Ling, T. W. 2015] Le, T. N., Bao, Z. and Ling, T. W.: "Exploiting semantics for XML keyword search"; Data & Knowledge Engineering, 99, (2015), 105-125.

[Le, W., Li, F., Kementsietsidis, A. and Duan, S. 2014] Le, W., Li, F., Kementsietsidis, A. and Duan, S.: "Scalable keyword search on large RDF data"; IEEE Transactions on Knowledge & Data Engineering, 26, 11 (2014), 2774-2788.

[Li, G., Feng, J., Wang, J. and Zhou, L. 2007] Li, G., Feng, J., Wang, J. and Zhou, L.: "Effective keyword search for valuable lcas over xml documents"; Proceedings of the 16th ACM Conference on information and knowledge management, ACM/Lisbon, Portugal, 31-40.

[Li, G., Ooi, B. C., Feng, J., Wang, J. and Zhou, L. 2008] Li, G., Ooi, B. C., Feng, J., Wang, J. and Zhou, L.: "EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data"; Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM/Vancouver, Canada, 903-914.

[Li, J., Liu, C., Zhou, R. and Ning, B. 2009] Li, J., Liu, C., Zhou, R. and Ning, B.: "Processing XML Keyword Search by Constructing Effective Structured Queries"; Advances in Data and Web Management, Springer Berlin Heidelberg, Berlin, Heidelberg (2009), 88-99.

[Li, J., Liu, C., Zhou, R. and Wang, W. 2010] Li, J., Liu, C., Zhou, R. and Wang, W.: "Suggestion of promising result types for XML keyword search"; Proceedings of the 13th International Conference on Extending Database Technology, ACM/Lausanne, Switzerland, 561-572.

[Li, Y., Yu, C. and Jagadish, H. V. 2004] Li, Y., Yu, C. and Jagadish, H. V.: "Schema-free XQuery"; Proceedings of the 30th international conference on Very large data bases VLDB Endowment/Toronto, Canada, 30, 72-83.

[Liu, F., Yu, C., Meng, W. and Chowdhury, A. 2006] Liu, F., Yu, C., Meng, W. and Chowdhury, A.: "Effective keyword search in relational databases"; Proceedings of the 2006 ACM SIGMOD international conference on Management of data, ACM/Chicago, USA, 563-574.

[Liu, Z. and Chen, Y. 2011] Liu, Z. and Chen, Y.: "Processing keyword search on XML: a survey"; World Wide Web, 14, 5 (2011), 671-707.

[Liu, Z. and Cher, Y. 2008] Liu, Z. and Cher, Y.: "Reasoning and identifying relevant matches for XML keyword search"; Proceedings of the VLDB Endowment, 1, 1 (2008), 921-932.

[Lopez-Veyna, J. I., Sosa-Sosa, V. J. and Lopez-Arevalo, I. 2014] Lopez-Veyna, J. I., Sosa-Sosa, V. J. and Lopez-Arevalo, I.: "A low redundancy strategy for keyword search in structured and semi-structured data"; Information Sciences, 288, 0 (2014), 135-152.

[Mass, Y. and Sagiv, Y. 2012] Mass, Y. and Sagiv, Y.: "Language models for keyword search over data graphs"; Proceedings of the 5th ACM international conference on Web search and data mining, ACM/Seattle, Washington, USA, 363-372.

[Mass, Y. and Sagiv, Y. 2016] Mass, Y. and Sagiv, Y.: "Virtual Documents and Answer Priors in Keyword Search over Data Graphs"; EDBT/ICDT Workshops, (2016).

[Mesquita, F., Silva, A. S. d., Moura, E. S. d., Calado, P. and Laender, A. H. F. 2007] Mesquita, F., Silva, A. S. d., Moura, E. S. d., Calado, P. and Laender, A. H. F.: "LABRADOR: Efficiently publishing relational databases on the web by using keyword-based query interfaces"; Information Processing & Management, 43, 4 (2007), 983-1004.

[Nguyen, K. and Cao, J. 2012] Nguyen, K. and Cao, J.: "Top-K data source selection for keyword queries over multiple XML data sources"; Journal of Information Science, 38, 2 (2012), 156-175.

[Ning, X., Jin, H., Jia, W. and Yuan, P. 2009] Ning, X., Jin, H., Jia, W. and Yuan, P.: "Practical and effective IR-style keyword search over semantic web"; Information Processing & Management, 45, 2 (2009), 263-271.

[Oliveira, P. d., Silva, A. d. and Moura, E. d. 2015] Oliveira, P. d., Silva, A. d. and Moura, E. d.: "Ranking Candidate Networks of relations to improve keyword search over relational databases"; 2015 IEEE 31st International Conference on Data Engineering, (2015), 399-410.

[Park, C.-S. and Lim, S. 2015] Park, C.-S. and Lim, S.: "Efficient processing of keyword queries over graph databases for finding effective answers"; Information Processing & Management, 51, 1 (2015), 42-57.

[Park, J. and Lee, S.-g. 2011] Park, J. and Lee, S.-g.: "Keyword search in relational databases"; Knowledge and Information Systems, 26, 2 (2011), 175-193.

[Pawar, S. S., Manepatil, A., Kadam, A. and Jagtap, P. 2016] Pawar, S. S., Manepatil, A., Kadam, A. and Jagtap, P.: "Keyword search in information retrieval and relational database system: Two class view"; 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), (2016), 4534-4540.

[Qin, L., Yu, J. X., Chang, L. and Tao, Y. 2009] Qin, L., Yu, J. X., Chang, L. and Tao, Y.: "Querying Communities in Relational Databases"; Proceedings of the 2009 IEEE International Conference on Data Engineering, IEEE Computer Society, 724-735.

[Tran, T., Haofen, W., Rudolph, S. and Cimiano, P. 2009] Tran, T., Haofen, W., Rudolph, S. and Cimiano, P.: "Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data"; IEEE International Conference on Data Engineering (2009), 405-416.

[Wang, D., Zou, L. and Zhao, D. 2015] Wang, D., Zou, L. and Zhao, D.: "Top-k queries on RDF graphs"; Information Sciences, 316, (2015), 201-217.

[Wang, H. and Aggarwal, C. C. 2010] Wang, H. and Aggarwal, C. C.: "A Survey of Algorithms for Keyword Search on Graph Data "; Mining and Managing Graph Data, C. C. Aggarwal and H. Wang, Springer US/Boston, MA (2010), 249-273.

[Xu, Y., Guan, J., Li, F. and Zhou, S. 2013] Xu, Y., Guan, J., Li, F. and Zhou, S.: "Scalable continual top-k keyword search in relational databases"; Data and Knowledge Engineering, 86, (2013), 206-223.

[Xu, Y. and Papakonstantinou, Y. 2005] Xu, Y. and Papakonstantinou, Y.: "Efficient keyword search for smallest LCAs in XML databases"; Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM/Baltimore, Maryland, 527-538.

[Xu, Y. and Papakonstantinou, Y. 2008] Xu, Y. and Papakonstantinou, Y.: "Efficient LCA based keyword search in XML data"; Proceedings of the 11th international conference on Extending database technology: Advances in database technology, ACM/Nantes, France, 535-546.

[Yu, J. X., Qin, L. and Chang, L. 2010] Yu, J. X., Qin, L. and Chang, L.: "Keyword Search in Databases", Morgan & Claypool, 155.

[Yu, X., Yu, Z., Liu, Y. and Shi, H. 2017] Yu, X., Yu, Z., Liu, Y. and Shi, H.: "CI-Rank: Collective importance ranking for keyword search in databases"; Information Sciences, 384, (2017), 1-20.

[Yuan, Y., Wang, G., Chen, L. and Wang, H. 2013] Yuan, Y., Wang, G., Chen, L. and Wang, H.: "Efficient Keyword Search on Uncertain Graph Data"; IEEE Transactions on Knowledge and Data Engineering, 25, 12 (2013), 2767-2779.

[Zhao, Y., Yuan, Y. and Wang, G. 2014] Zhao, Y., Yuan, Y. and Wang, G.: "Nearest Keyword Search on Probabilistic XML Data"; Web Technologies and Applications, L. Chen, Y. Jia, T. Sellis and G. Liu, Springer International Publishing (2014), 485-493.

[Zhou, Q., Wang, C., Xiong, M., Wang, H. and Yu, Y. 2007] Zhou, Q., Wang, C., Xiong, M., Wang, H. and Yu, Y.: "SPARK: adapting keyword query to semantic search"; Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference, Springer-Verlag/Busan, Korea, 694-707.

[Zong, N., Lee, S. and Kim, H.-G. 2015] Zong, N., Lee, S. and Kim, H.-G.: "Discovering expansion entities for keyword-based entity search in linked data"; Journal of Information Science, 41, 2 (2015), 209-227.