Summary Models for Routing Keywords to Linked Data Sources

Thanh Tran, Lei Zhang, Rudi Studer

Institute AIFB, Karlsruhe Institute of Technology, Germany
{dtr,lzh,studer}@kit.edu

Abstract. The proliferation of linked data on the Web paves the way to a new generation of applications that exploit heterogeneous data from different sources. However, because this Web of data is large and continuously evolving, it is non-trivial to identify the relevant link data sources and to express some given information needs as structured queries against these sources. In this work, we allow users to express needs in terms of simple keywords. Given the keywords, we define the problem of finding the relevant sources as the one of keyword query routing. As a solution, we present a family of summary models, which compactly represents the Web of linked data and allows to quickly find relevant sources. The proposed models capture information at different levels, representing summaries of varying granularity. They represent different trade-offs between effectiveness and efficiency. We provide a theoretical analysis of these trade-offs and also, verify them in experiments carried out in a real-world setting using more than 150 publicly available datasets.

1 Introduction

The Web is no longer only a collection of textual documents but also a *Web of linked data*. One prominent project which largely contributes to this development is the Linking Open Data project. Collectively, linked data comprises hundreds of sources containing over 13.1 billions RDF triples, which are connected by 142 millions links (November 2009, `http://linkeddata.org/`).

This development offers new opportunities for addressing complex information needs. Instead of documents, complex results ranging over different sources of linked data can be returned to Web users. To exploit this, users can specify complex queries using structured query languages such as SPARQL[1]. While such a query language is powerful, it requires users to know not only the query syntax and semantics but also the schema as well as the underlying data.

Problem So far, these requirements have proven to be a large burden. Given the amount of linked data is large and continuously evolving, it is inherently difficult to know what is in there (i.e., the data and the schema) and to formulate the corresponding structured queries for addressing some given information needs. Hence, it is desirable to have a mechanism, which allows users to express information needs in their own words. Another aspect of dealing with the large Web of linked data is scalability. Processing the needs against the entire Web might

[1] http://www.w3.org/TR/rdf-sparql-query/

be too time consuming and not needed, especially when users are interested in and want to choose some particular sources of information. Processing against a relevant subset of linked data identified by the user is more scalable and possibly the only practical solution for the large Web of linked data. Concerning these problems, the question we deal with is *given the needs expressed by users as sets of keywords, are there corresponding answers in linked data and what combination of data sources shall be used to produce them?*.

Existing Work In the Semantic Web community, there exists a large body of work on processing queries against RDF and linked data. Given structured queries, RDF stores such as RDF-3X [5] and YAR2 [3] can compute structured results and in the context of linked data query processing [2], can also identify relevant sources. They however do not apply when the information need is provided as keywords. While keyword search is supported by some Semantic Web search engines such as SWSE [1] and Sig.ma [9], they are limited to processing simple list of keywords that refer to entities. This work deals with *complex information needs*, which may involve complex results providing information about sets of entities and relations between them, i.e., result tuples that may form *graphs*. Further, the aim is not to directly compute results but to quickly identify and let users and system focus on the combination of sources that produce non-empty results.

To this end, work in information retrieval (IR) and database research dealing with keyword search constitutes the starting point. Keyword search has become the most widely used IR paradigm on the Web, enabling lay users without knowledge of the schema and data to search for a priori unknown documents. This kind of schema-agnostic search is not limited to textual data but can also be used for querying structured data. In database research, solutions have been proposed that allow for the retrieval of the most relevant, possibly graph-structured results [4, 6, 7]. Unlike IR approaches, which consider only keywords for finding matching documents (we called *keyword coverage*), database approaches also use structure information. Possible join sequences in the data are explored to ensure that matching result tuples not only "contain" the keywords but also, represent meaningful connections between these keywords (called *structure coverage*). Given the data graph in Fig. 1a and the query "Stanford, John, Award" for instance, an IR-style approach might return none (AND-semantics) or all the entities $uni1, per2, ...$ in the graph because they all partially match the keyword query (OR-semantics) whereas the DB approach would return the subgraph that connects $uni1$ with $per2, per1, per3$ and $prize1$. However, computing complex results in this way is expensive, especially in a multi-source setting like the linked data Web. Authors of state-of-the-art work explicitly considered only the setting where "number of databases that can be dealt with is up to the tens" [6].

Thus, database researchers started to look at a problem we consider most related, namely the of finding the single most relevant databases [11, 10]. They recognized the fact that the computational complexity resulting from a large-scale setting can be partially addressed when allowing users to choose and retrieve answers from only some particular databases. Given a set of keywords, the goal is to find and rank the single most relevant databases that contain the answers. Following this line, we propose specific solutions for the linked data context. The differ-

ences to this work called *database selection* will be discussed in detail throughout the paper.

Contributions While existing approaches select single databases, we deal with the Web of linked data where results are not bounded by a single source but may encompass several linked data sources. Instead of computing the most relevant single sources, we extend the work in [11, 10] to compute the most relevant combinations of sources. The goal is to produce *keyword routing plans* which capture combinations of sources that contain non-empty results. This novel *keyword query routing* problem raises additional challenges. Most notably, query keywords may be covered by several linked sources, resulting in a large search space. The size of this search space grow exponentially with the number of sources and their associated links. Targeting this problem of *scale*, we report the following contributions in this paper:

- We propose *solutions for keyword query routing* which enable the exploitation of linked data. Without putting any burden on the users, this kind of approaches help to find relevant sources containing complex answers to ad-hoc information needs in the large and evolving Web of linked data.
- We propose a *multi-level relationship graph* to capture the search space of the keyword query routing problem. Based on this, we elaborate on *a family of summary models*, which compactly represent the Web of linked data. These models capture information at different levels, representing summaries of different granularities. In a theoretical analysis, we prove that finer grained models can improve the result quality. This however, comes at the expense of higher complexity. Thus, the models represent different trade-offs between effectiveness and efficiency.
- In the experiments, we investigate these trade-offs by analyzing the precision and the processing time needed using different models. The experiments were carried out in a real-world setting using more than 150 publicly available datasets, and an open-source implementation we made available at `http://code.google.com/p/rdfstores/`. Results of using summaries are promising. While the "best" one shall be determined w.r.t a concrete application, there is one model that seems to represent the most practical trade-off: the D-KERG model, which summarizes elements according to sources, produces results in less than 10ms, out of which every second is a valid one.

Outline Section 2 introduces the readers to the concepts of linked data and keyword query routing. The search space and its summary models are presented in Section 3. Strategies for computing routing plans using these models and ramifications for result quality and performance are discussed in Section 4. Evaluation results are provided in Section 5 before we conclude in Section 6.

2 Preliminary

In this section, we discuss the underlying data and problem.

2.1 Web of Linked Data

Linked data can be conceived as a set of data graphs, each represents a particular source. As a working definition, we present a simple graph-based model of linked data called the *Web graph*. In that model, we distinguish between the *Web data graph* representing relationships between individual data elements, the *Web schema graph*, which captures information about group of elements, and the *Web source graph* that contains information at the level of data sources.

Definition 1 (Web Graph). *The Web of linked data is modeled as a* Web Graph $\mathcal{W}^*(\mathcal{G}^*, \mathcal{M}^*, \mathcal{N}^*, \mathcal{E}^*)$ *where* $\mathcal{G}^*$ *denotes the set of* data graphs, $\mathcal{M}^*$ *is the set of edges also called mappings or* links*, which establish connections between elements of two different graphs,* $\mathcal{N}^*$ *is the set of all nodes and* $\mathcal{E}^*$ *is the set of all edges, i.e.,* $\mathcal{G}^* = \{g_1(\mathcal{N}^*_1, \mathcal{E}^*_1), g_2(\mathcal{N}^*_2, \mathcal{E}^*_2), \ldots, g_n(\mathcal{N}^*_n, \mathcal{E}^*_n)\}$, $\mathcal{N}^* = \bigcup_{i=1}^{n} \mathcal{N}^*_i$, $\mathcal{M}^* = \{m(n_i, n_j) | n_i \in \mathcal{N}^*_i, n_j \in \mathcal{N}^*_j, \mathcal{N}^*_i, \mathcal{N}^*_j \subseteq \mathcal{N}^*, i \neq j\}$ *and* $\mathcal{E}^* = \bigcup_{i=1}^{n} \mathcal{E}^*_i \cup \mathcal{M}^*$. *We use* $\mathcal{W}(\mathcal{G}, \mathcal{M}, \mathcal{N}, \mathcal{E})$ *to distinguish the* Web data graph *from the* Web schema graph $\mathcal{W}'(\mathcal{G}', \mathcal{M}', \mathcal{N}', \mathcal{E}')$ *and the* Web source graph $\mathcal{W}''(\mathcal{N}'', \mathcal{E}'')$. *We have* $n \in \mathcal{N}$ *representing a data element,* $n' \in \mathcal{N}'$ *stands for a group of elements, and* $n'' \in \mathcal{N}''$ *denotes a data source. For simplicity, we use* $n \in n'$ *to denote that an element* n *belongs to the group* n' *and* $n, n' \in n''$ *to assert the element* n *and the group* n' *belongs to the source* n''. *Elements in* $\mathcal{N}$ *and* $\mathcal{N}'$ *are labeled, i.e., there is a function label* $: \mathcal{N} \cup \mathcal{N}' \mapsto 2^{\mathcal{V}}$ *that associates an element with a set of labels drawn from* $\mathcal{V}$*, the vocabulary of words. We have* $m(n'_i, n'_j) \in \mathcal{M}'$ *iff there is* $m(n_i, n_j) \in \mathcal{M}$ *where* $n_i \in n'_i$ *and* $n_j \in n'_j$. *Analogously,* $e(n''_i, n''_j) \in \mathcal{E}''$ *iff there is* $m(n'_i, n'_j) \in \mathcal{M}'$ *where* $n'_i \in n''_i$ *and* $n'_j \in n''_j$. *We use the Web graph* $\mathcal{W}^*(\mathcal{G}^*, \mathcal{M}^*, \mathcal{N}^*, \mathcal{E}^*)$ *to refer to the union set of elements of the Web data graph, the Web schema graph and the Web source graph.*

This is a simple model of linked data that omits details not necessary for this work. In particular, data elements may correspond to RDF resources, blank nodes or literals. Schema elements might stand for classes or data types. For keyword query routing, these distinctions are not relevant but the fact that the elements can be recognized via their labels. While different kinds of links can be established, the ones frequently found are *sameAs* links, which denote that two RDF resources or two classes are the same. There is also no need to distinguish the types of links. Only the fact that sources can be reached via some kinds of link $m \in \mathcal{M}^*$ matters. An example of this model is illustrated in Figs. 1.

2.2 Keyword Query Routing

Given the need expressed as keywords, we aim to identify sources containing results. A DB-style result to a keyword query is typically a Steiner graph, which in the linked data scenario, may combine data from several sources:

Definition 2 (Keyword Query Result). *A Web data graph* $\mathcal{W}(\mathcal{G}, \mathcal{M}, \mathcal{N}, \mathcal{E})$ *contains a result for a query* $\mathcal{K} = \{k_1, k_2, \ldots, k_{|\mathcal{K}|}\}$ *if there is subgraph also called* Steiner graph $\mathcal{W}_{\mathcal{K}}(\mathcal{G}_{\mathcal{K}}, \mathcal{M}_{\mathcal{K}}, \mathcal{N}_{\mathcal{K}}, \mathcal{E}_{\mathcal{K}})$*, where for all* $k_i \in \mathcal{K}$*, there is an* $n^{\mathcal{M}}_{\mathcal{K}} \in \mathcal{N}^{\mathcal{M}}_{\mathcal{K}} \subseteq \mathcal{N}_{\mathcal{K}} \subseteq \mathcal{N}$ *with a label that matches* k_i *(*$\mathcal{N}^{\mathcal{M}}_{\mathcal{K}}$ *is called the set of* keyword elements*), and there is path* $n_i \leftrightsquigarrow n_j$ *for all* $n_i, n_j \in \mathcal{N}^{\mathcal{M}}_{\mathcal{K}}$. *In a* d-max Steiner graph*, the length of the paths* $n_i \leftrightsquigarrow n_j$ *is d-max or less.*

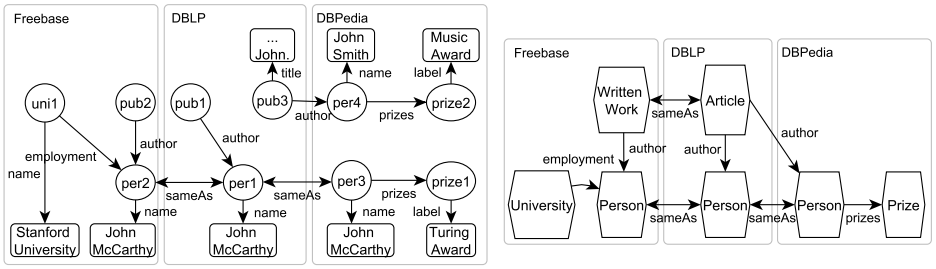


Fig. 1: (a) A Web data graph (left) and (b) its Web schema graph (right).

Typical for keyword search is the pragmatic assumption that users are only interested in compact results such that a threshold d_{max} can be used to constrain the connections to be considered. Thus, instead of general Steiner graphs, keyword search solutions proposed so far and the work presented here consider *d-max Steiner graphs* as results. For our example query "Stanford, John, Award", we have $\mathcal{N}_{\mathcal{K}}^{\mathcal{M}} = \{uni1, per2, per1, per3, prize1\}$; the subgraph that connects these keyword elements is a 1-Steiner graph because the maximum distance between keyword elements is 1; and since there are no other elements between keyword elements, $\mathcal{N}_{\mathcal{K}}^{\mathcal{M}} = \mathcal{N}_{\mathcal{K}}$.

Definition 3 (Keyword Routing Plan). *Given the Web data graph* $\mathcal{W} = (\mathcal{G}, \mathcal{M}, \mathcal{N}, \mathcal{E})$ *and a set of keyword queries* $\mathcal{SK}$*, the mapping* $\mu : \mathcal{SK} \mapsto 2^{\mathcal{G}}$ *that associates a query with a set of data graphs is called a* keyword routing plan $\mathcal{RP}$*. A plan* $\mathcal{RP} = \{g_1, \ldots, g_{|\mathcal{K}|}\}$ *for a query* $\mathcal{K} \in \mathcal{SK}$ *is considered* valid *when there is a combination of data graphs* $g_i \in \mathcal{RP}$ *that produces non-empty results for* $\mathcal{K}$*.*

A valid plan in our example is $\mathcal{RP} = \{Freebase, DBLP, DBPedia\}$. Note that validity does not imply relevance. That is, a valid plan ensures that results can be produced, but for the users, these results may differ in relevance. A proper account of relevance and the ranking of routing plans based on the relevance of their results go beyond the scope of this paper, which is focused on *efficiency* aspects of computing valid plans. We assume a fixed ranking function, which equally applies to all summaries discussed in this paper. We refer the interested readers to our report [8], which discusses relevance and the ranking function.

3 Summary Models for Keyword Query Routing

We now discuss the most related work in detail and introduce the models we use for keyword query routing.

3.1 Keyword Query Routing Search Space

For database selection, the search space is composed of a set of databases. The idea behind previous work [11, 10] is to model every database using a keyword relationship model. A keyword relationship $\langle k_i, k_j \rangle$ is a pair of keywords, which

can be connected via a sequence of join operations, i.e., there exists two data elements $n_i \leftrightsquigarrow n_j$ that contain k_i and k_j. For instance, $\langle Stanford, Award \rangle$ is a keyword relationship because there is a path between FB:$uni1$ and DBP:$prize1$ in Fig. 1a. The state-of-the-art [10] employs a keyword relationship graph (KRG), with keywords being nodes and keyword relationships being edges. A database is relevant when all pairs of query keywords match some edges of the KRG.

In our example, we have the keyword pairs $(Stanford, John)$, $(Stanford, Award)$ and $(John, Award)$. It is clear that when using keyword relationships in every source to form separate KRGs, none of them matches all the 3 keyword pairs. To match the pair $(Stanford, Award)$, relationships across sources from $Freebase$ to $DBPedia$ have to be incorporated into in the model. In keyword query routing, the search space does not comprise single databases but constitutes one integrated Web data graph. Instead of computing a set of summary models, this problem requires the construction of one *integrated summary model*. It shall allow for answers capturing relationships across sources. Thus, not only single sources but also combinations of sources might be relevant. Another aspect not addressed by current work is efficiency. Instead of capturing all possible relationships, we aim to use a more *compact* representations of the search space.

We conceive the search space as a *multi-level inter-relationship graph* (MIRG), as illustrated in Fig. 2. For clarity, this figure does not show the labels and also, omits some data and schema elements of our running example. At the lowest level, it models relationships between keywords. In the upper-levels, there is the Web data graph $\mathcal{W}$ followed by $\mathcal{W}'$ and $\mathcal{W}''$. Elements and relationships at the upper level represent sets of elements and sets of relationships at the lower level: a node at the source level represents a set of schema elements; every schema node represents a set of data elements; and every data element n is composed of a set of keywords K. We say $k \in K$ is mentioned in n, denoted $mentionedIn(k, n)$.

3.2 Summary Models

Thus, MIRG provides different perspectives on the search space and different views on the data. The lower levels capture more fine-grained views of the data. In order to extend the KRG [10] to deal with keyword query routing, the keyword level and keyword relationships at this level that also capture links between sources have to be taken into account. We will now discuss such an extension of the KRG, and introduce further summary models that capture relationships at different levels of granularity. Examples of the models are shown in Fig. 3.

Definition 4 (Keyword Sets). *The* keyword sets *(KS) of a Web graph* $\mathcal{W}^*(\mathcal{G}^*, \mathcal{M}^*, \mathcal{N}^*, \mathcal{E}^*)$ *is* $\mathcal{W}_{\mathcal{K}}^{KS} = \mathcal{N}_{\mathcal{K}}^{KS}$*, where* $\mathcal{N}_{\mathcal{K}}^{KS}$ *stands for all the keywords that are mentioned in elements of the graphs* $\mathcal{G}^*$*. Every* $n_k^{KS} \in \mathcal{N}_{\mathcal{K}}^{KS}$ *is in fact a tuple* $(k, \mathcal{G}_k)$ *that represents a keyword* k *and the graphs* $\mathcal{G}_k \subset \mathcal{G}^*$ *mentioning* k*.*

This is a simple model that contains only keywords but no relationships between them. It captures all nodes at the keyword level of MIRG.

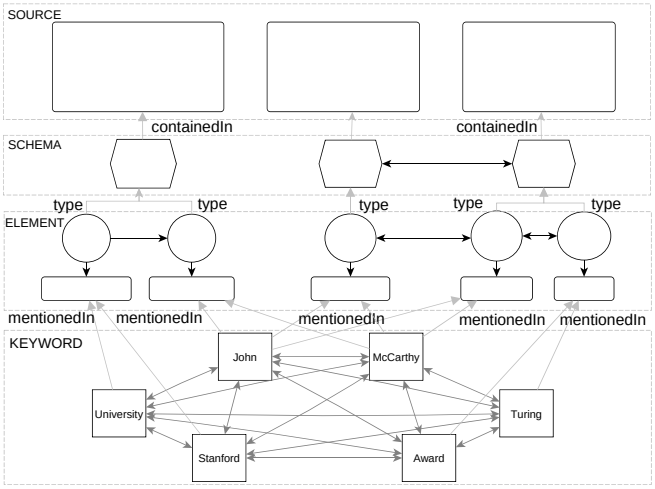


Fig. 2: Multi-level inter-relationship graph

Definition 5 (Element-level KERG). *An* element-level keyword-element relationship graph (E-KERG) *of a Web graph* $\mathcal{W}^*(\mathcal{G}^*, \mathcal{M}^*, \mathcal{N}^*, \mathcal{E}^*)$ *is a tuple* $\mathcal{W}_\mathcal{K} = (\mathcal{N}_\mathcal{K}, \mathcal{E}_\mathcal{K})$. *Every keyword-element* $n_\mathcal{K} \in \mathcal{N}_\mathcal{K}$ *is a tuple* $(n, g, \mathcal{K})$ *where* $n \in \mathcal{N}$ *is the corresponding element node it represents,* $g \in \mathcal{G} \subset \mathcal{G}^*$ *is the data graph containing* n*, and* $\mathcal{K}$ *is the set of all keywords that are mentioned in* n*, i.e.,* $K = \{k | mentionedIn(k, n)\}$. *There is a relationship* $e_\mathcal{K} = (\langle k_i, n_{\mathcal{K}_i}(n_i, g_i, K_i)\rangle, \langle k_j, n_{\mathcal{K}_j}(n_j, g_j, K_j)\rangle) \in \mathcal{E}_\mathcal{K}$, *iff* $mentionedIn(k_i, n_i)$, $mentionedIn(k_j, n_j)$, *and* $n_i \leftrightsquigarrow n_j$.

This can be seen as an extension of the KRG because it captures all keywords and relationships. As shown in Fig. 3a, it also represents the data elements in which the keywords are mentioned. Hence, we use "keyword-element" to make clear that a node captures both the data element and its keywords. This model captures elements at the keyword and element level of the MIRG.

Definition 6 (Schema-level KERG). *A* schema-level keyword-element relationship graph (S-KERG) *is a tuple* $\mathcal{W}'_\mathcal{K} = (\mathcal{N}'_\mathcal{K}, \mathcal{E}'_\mathcal{K})$. *It captures elements at the keyword and schema level of the MIRG. For a keyword-element node* $n'_\mathcal{K}(n', g, \mathcal{K}) \in \mathcal{N}'_\mathcal{K}$, *we have* $n' \in \mathcal{N}'$ *being a schema-level node,* $g \in \mathcal{G}' \subset \mathcal{G}^*$ *is the schema graph containing* n'*, and* $\mathcal{K}$ *comprises keywords that are mentioned in the elements* $n \in n'$*, i.e.,* $\mathcal{K} = \{k | n \in n', mentionedIn(k, n)\}$. *There is a relationship* $e'_\mathcal{K} = (\langle k_i, n'_{\mathcal{K}_i}(n'_i, g_i, K_i)\rangle, \langle k_j, n'_{\mathcal{K}_j}(n'_j, g_j, K_j)\rangle) \in \mathcal{E}'_\mathcal{K}$, *iff* $mentionedIn(k_i, n_i)$, $mentionedIn(k_j, n_j)$, $n_i \in n'_i$, $n_j \in n'_j$, *and* $n_i \leftrightsquigarrow n_j$.

As opposed to E-KERG, this one is indeed a summary model because it clusters two element-level relationships $(\langle k_i, n_{\mathcal{K}_i}(n_i, g_i, K_i)\rangle, \langle k_j, n_{\mathcal{K}_j}(n_j, g_j, K_j)\rangle)$ and $(\langle k_v, n_{\mathcal{K}_v}(n_v, g_v, K_v)\rangle, \langle k_w, n_{\mathcal{K}_w}(n_w, g_w, K_w)\rangle)$ to one schema-level relationship when they capture the same keyword relationships (i.e., $k_i = k_v$ and $k_j = k_w$) between the same classes (i.e, $n'_i = n'_v$ and $n'_j =$

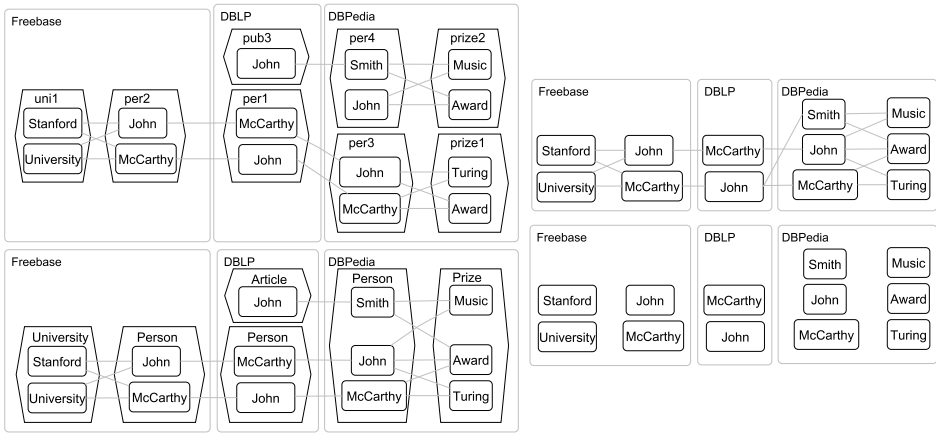


Fig. 3: (a)The 1-E-KERG (top left), (b) the 1-S-KERG (bottom left), (c) the 1-D-KERG (top right) and the (d) KS for our running example.

n'_w). For instance, ($\langle John,\ (Person, DBPedia, \{Smith, John, McCarthy\})\rangle$, $\langle Award,\ (Prize, DBPedia,\ \{Music, Award, Turing\})\rangle$) in Fig. 3b is an aggregation of the relationships ($\langle John,\ (per4, DBPedia, \{Smith, John\})\rangle$, $\langle Award,\ (prize2, DBPedia, \{Music, Award\})\rangle$) and ($\langle John, (per3, DBPedia, \{McCarthy, John\})\rangle, \langle Award, (prize1, DBPedia, \{Turing, Award\})\rangle$) in Fig. 3a. These E-KERG relationships are aggregated because they represent the same relationships $(John, Award)$ between the classes $(Person, Prize)$.

Definition 7 (Source-level KERG). *A* source-level keyword-element relationship graph (D-KERG) *is a tuple* $\mathcal{W}''_{\mathcal{K}} = (\mathcal{N}''_{\mathcal{K}}, \mathcal{E}''_{\mathcal{K}})$. *For a keyword-element node* $n''_{\mathcal{K}}(n'', \mathcal{K}) \in \mathcal{N}''_{\mathcal{K}}$, *we have* $n'' \in \mathcal{N}''$ *being a source-level node, i.e., a graph, and* $\mathcal{K}$ *is the set of all keywords that are mentioned in elements of the graph* n''. *There is a relationship* $e''_{\mathcal{K}} = (\langle k_i, n''_{\mathcal{K}_i}(n''_i, K_i)\rangle, k_j, n''_{\mathcal{K}_j}(n''_j, K_j)\rangle) \in \mathcal{E}''_{\mathcal{K}}$, *iff* k_i *is mentioned in some elements* n_i *of the graph* n''_i, k_j *mentioned in some elements* n_j *of the graph* n''_j, *and* $n_i \leftrightsquigarrow n_j$.

Thus, this model is conceptually similar to S-KERG but aggregate elements at the level of sources. It combines schema-level relationships when they capture the same keyword relationships between the same sources. As shown in Fig. 3b, there are only distinct keyword relationships in S-KERG. Thus, no further aggregation is needed in this case.

As keyword search results, we consider d-max Steiner graphs where paths between keyword elements are of length d_{max} or less. Accordingly, we actually employ a d_{max}-*KERG* versions where the maximum distance to be considered between n_i and n_j is d_{max} ($n_i \leftrightsquigarrow^{d_{max}} n_j$). Note that the summaries illustrated in Figs. 3 resemble the structure of the underlying data and schema graphs because relationships in the summaries in fact correspond to graph edges, i.e., only paths with length 1 are considered such that $d_{max} = 1$ (1-KERG models). Clearly, a higher value for d_{max} would result in a blowup of paths. In particular, the E-KERG model would contain much more relationships than there are edges in the

data graph. Hence, summarizing relationships is essential for efficient keyword query routing.

3.3 Computing Summary Models

The computation of d_{max}-KERG models is performed in three steps. Firstly, the relationships between entities are computed for various distances within a threshold d_{max}. Then, connected term pairs are extracted based on the computed relationships. They are used for computing E-KERG. For computing S-KERG and D-KERG, term pairs are further grouped according to schema and source-level elements, respectively.

All information are finally stored in an specialized index that enables the lookup of keyword-element relationships, given a pair of keywords. In particular, for (k_i, k_j), we have (1) I_{E-KERG} returning the relationships $e_{\mathcal{K}} = (\langle k_i, n_{\mathcal{K}_i}\rangle, \langle k_j, n_{\mathcal{K}_j}\rangle)$, (2) I_{S-KERG} returning $e'_{\mathcal{K}} = (\langle k_i, n'_{\mathcal{K}_i}\rangle, \langle k_j, n'_{\mathcal{K}_j}\rangle)$ and (3) I_{D-KERG} returning $e''_{\mathcal{K}} = (\langle k_i, n''_{\mathcal{K}_i}\rangle, \langle k_j, n''_{\mathcal{K}_j}\rangle)$. Also, we construct a KS model based on keywords extracted from the data graphs and build the index (4) I_{KS}, which returns the elements n_k^{KS}, given the keyword k.

4 Computing Keyword Routing Plans

For computing valid query routing plans, the idea behind existing work on keyword search [4, 6, 7] and database selection [11, 10] applies: we search for Steiner graphs to discover sources that produce answers. Recall that a Steiner graph is basically a graph that connects keyword elements. The existence of such a graph indicates that there are answers to the keyword query. In our approach, the search is not performed directly on the Web data graph but on the summary models. Specifically, we search for Steiner graphs in either (1) $\mathcal{W}^{ks}_{\mathcal{K}}$, (2) $\mathcal{W}_{\mathcal{K}}$, (3) $\mathcal{W}'_{\mathcal{K}}$ or (4) $\mathcal{W}''_{\mathcal{K}}$. Since KS (1) do not capture relationships, the results that can be derived from it do not completely adhere to the notion of Steiner graph. Also for the KERG models (2-4), Steiner graphs that can be computed are different in granularities. We will now elaborate on strategies for searching Steiner graphs using different summaries.

4.1 Routing Plan Computation Using KS

Using the KS model and its index, routing plans can be computed as follows:

- Given the keyword query $\mathcal{K} = \{k_1, k_2, \ldots, k_i\}$, retrieve the elements $n_{k_i}^{KS}(k_i, \mathcal{G}_{k_i})$ for every $k_i \in \mathcal{K}$ using the I_{KS} index.
- For every $n_{k_i}^{KS}$ retrieved before, put the sources $\mathcal{G}_{k_i}$ that is associated with $n_{k_i}^{KS}$ into the set of relevant sources $\mathcal{G}_{\mathcal{K}}$.
- Compute all $|\mathcal{K}|$-combinations for the set $\mathcal{G}_{\mathcal{K}}$.
- Output these combinations as the set of routing plans $\mathcal{SRP}$.

Intuitively speaking, this procedure simply retrieves sources that cover the keywords and in order to cover all $|\mathcal{K}|$ query keywords, it uses $|\mathcal{K}|$-combinations of these sources as routing plans.

4.2 Routing Plan Computation Using KERGs

Since KERG models capture relationships, we retrieve data for pair of keywords (k_i, k_j), instead of single keywords. Retrieved data are joined to compute Steiner graphs. It is necessary to ensure that all keyword elements in a Steiner graph are pairwise connected through a path of length d_{max} or less. Thus, it is necessary to join all possible keyword pairs. Given a query with three keywords k_1, k_2, k_3 for instance, we need to retrieve keyword elements and perform the joins $(k_1, k_2) \bowtie_{k_2} (k_2, k_3) \bowtie_{k_3,k_1} (k_1, k_3)$ to verify that (1) the elements n_2 matching k_2 are connected with both n_1 that match k_1 and n_3 that match k_3 over a distance of d_{max} or less (by means of the first join $((k_1, k_2) \bowtie_{k_2} (k_2, k_3)$ on k_2), (2) the elements n_3 just found to be connected with n_2, are also connected with n_1 (by means of the second join on k_3), and (3) the n_1 found to be connected with n_2, is also connected with n_3 (by means of the third join on k_1). The complete procedure can be summarized as follows:

- Given the keyword query $\mathcal{K} = \{k_1, k_2, \ldots, k_i\}$, compute all 2-combinations of $\mathcal{K}$ to get all possible keyword pairs, resulting in a total of $N = |K|(|K|-1)/2$ different pairs. Subsequently, retrieve relationships for these pairs and perform joins according to a random[2] or alternatively, optimized order.
- In particular, inputs for every keyword pair are obtained using the underlying index. Given (k_1, k_2) and I_{E-KERG} for instance, relationships of the form $e_{\mathcal{K}} = (\langle k_1, n_{\mathcal{K}_1}(n_1, g_1, K_1)\rangle, \langle k_2, n_{\mathcal{K}_2}(n_2, g_2, K_2)\rangle)$ are retrieved. Joining this with the next inputs retrieved for (k_2, k_3) for instance, ensures that n_2 is connected with both n_1 and n_3.
- Processing the entire join sequence of keyword pairs yields a set of graphs. Depending on the underlying summary model, these graphs capture Steiner graphs at the source, schema or element level.
- Some resulting graphs might be indistinguishable in terms of the sources and connections between sources they represent. Keep only one of those because the other does not contain additional information.
- Extract sources associated with elements of the graphs to obtain combination of sources, i.e., the routing plans $\mathcal{RP}$.

This procedure is the same for all KERGs. Given that the underlying data contain results, we provide proofs in the report [8] to show that applying this procedure on the S-KERG summary will yield routing plans, i.e., when Steiner graphs can be found for $\mathcal{K}$ in the data, then there will be corresponding graphs that can be found in the summary. Thus, given $\mathcal{K}$, the procedure will output a non-empty set of $\mathcal{RP}$ if $\mathcal{W}$ contains a result for $\mathcal{K}$. In the same manner, it is straightforward to show that E-KERG and D-KERG can provide this guarantee. However, we show formally in [8] that the other way around is not true, i.e., the graphs derived from the summary are not necessarily valid such that there might be no corresponding Steiner graph in the data. Thus, the fact that a routing plan can be derived from the summaries does not guarantee there exists a result for $\mathcal{K}$. This formal result is interesting because it makes clear that while the

[2] For this work, we omit the aspect of join order optimization and simply generate a random order for joining keyword pairs.

use of summaries might be required to obtain the desired performance, it has consequences on the result validity. In particular, it implies that the more compact the summary, the more likely that plans computed from it are not valid. We will now discuss the intuition behind this formal result.

4.3 Result Validity

A graph derived from a summary does not always have a corresponding Steiner graph in the data, unless we use E-KERG. This model makes a difference because it in fact captures all nodes and paths in the Web data graph. In particular, when a d_{max}^{sum}-E-KERG is used, d_{max}^{data}-Steiner graphs are keyword query answers, and $d_{max}^{sum} = d_{max}^{data}$, then E-KERG captures all the paths in the data that are relevant for Steiner graph computation, i.e., all paths up to length d_{max}^{sum}. For every path $n_i \rightsquigarrow^{d_{max}} n_j$ in the data, there is a one-to-one corresponding relationship $e_{\mathcal{K}} = (\langle k_i, n_{\mathcal{K}_i}(n_i, g_i, K_i)\rangle, \langle k_j, n_{\mathcal{K}_j}(n_j, g_j, K_j)\rangle)$ in E-KERG. This one-to-one correspondence of paths constitutes the base argument, which can be extended inductively to show that there is also a one-to-one correspondence of graphs such that a graph derived from E-KERG always has a corresponding Steiner graph in the data, and vice versa.

A S-KERG however, combines two edges $e(n_{i_1}, n_{j_1})$ and $e(n_{i_2}, n_{j_2})$ (paths) in the data graph to one single relationship $e'_{\mathcal{K}} = (\langle k_i, n'_{\mathcal{K}_i}(n'_i, g_i, K_i)\rangle, \langle k_j, n'_{\mathcal{K}_j}(n'_j, g_j, K_j)\rangle)$ iff $n_{i_1}, n_{i_2} \in n'_i$, $n_{j_1}, n_{j_2} \in n'_j$, n_{i_1}, n_{i_2} mention k_i, and n_{j_1}, n_{j_2} mention k_j. Thus, for an element in the data, there is always a counterpart in S-KERG, which is however a grouping of elements, constituting a one-to-many correspondence. Through this grouping, we loose detailed information about elements in the group. That is, for the pair of keyword (k_i, k_j), S-KERG captures the corresponding connection from the element group n'_i to n'_j but can no longer tell for instance, whether this represents a connection between n_{i_1} to n_{j_1} or n_{i_1} to n_{j_2}. In other words, it can be inferred from S-KERG that n_{i_1} is connected with n_{j_2} even though such a connection does not exist in the data. With respect to our example, a graph can be derived from S-KERG that covers the keywords *Stanford*, *John* and *Music*. It is clear from Fig. 3a that there is no Steiner graph corresponding to this. The problem here is that S-KERG does not distinguish the John McCarthy connected with "Stanford" from the John Smith connected with "Music". Thus, it incorrectly infers the connection "Stanford" and "Music".

The same arguments can be applied to D-KERG. The difference is that the grouping in D-KERG is even more coarse-grained. Two edges $e(n_{i_1}, n_{j_1})$ $e(n_{i_2}, n_{j_2})$ are aggregated to one single relationship in D-KERG, when n_{i_1} and n_{i_2} mention the same keyword k_i, n_{j_1} and n_{j_2} mention k_j, and they belong to the same data source, i.e., $n_{i_1}, n_{i_2} \in n''_i$ and $n_{j_1}, n_{j_2} \in n''_j$. Note that with S-KERG, the incorrect inference mentioned before would not occur when John Smith is in a different class than John McCarthy. They would not have been aggregated to one single node in S-KERG. This however happens with D-KERG. No matter the classes they belong to, these elements would be aggregated to one single node when they are in the same data source. With respect to our example, D-KERG makes one additional false inference: it does not distinguish the person "John"

connected with "Stanford" from yet another "John" connected with "Music", which is an article (see that John as an article and John as a person in Fig. 3b is aggregated to one element in Fig. 3c).

Compared to the KERG models, KS does not capture relationships between keywords at all. Given two keywords k_i, k_j, the sources which cover these keywords can be derived from KS, e.g. the graphs n''_i, n''_j. However, this does not imply there exist two elements $n_i \in n''_i$ and $n_j \in n''_j$, and $n_i \rightsquigarrow n_j$. More generally, a combination of sources derived from KS covers all keywords but does not ensure that elements matching these keywords are connected, and thus, does not necessarily correspond to a Steiner graph.

In summary, the percentage of valid plans for D-KERG is less or equal that for S-KERG, which in turn is less or equal that for E-KERG. When d^{sum}_{max} value of E-KERG is sufficiently large to cover all paths relevant for Steiner graph computation, i.e., $d^{sum}_{max} = d^{data}_{max}$, this percentage is 100 for E-KERG. By chance, the percentage of valid plans for KS might be higher than that for the summary models but in general, is expected to be less (because relationships between elements are not considered).

4.4 Complexity

Using KS, complexity is $O(input_{max}^{|K|-1})$, where $input_{max}$ denotes the largest number of elements that can be obtained for a keyword k_i. This is because for computing the combination of sources for a 2-keyword query $K = \{k_i, k_j\}$, we have to union every element retrieved for k_i with every other retrieved for k_j (Cartesian product), thus requiring $|input_i| \times |input_j|$ time and space. For queries with $|K|$ keywords, we have to combine elements retrieved for one keyword k_i with elements retrieved for every other keyword $k_j \in K, k_j \neq k_i$. Thus, $|K| - 1$ combinations of input sets of maximum size $input_{max}$ have to be performed.

With KERG models, retrieved elements have to be joined. While in practice, this operation can be performed more efficiently using special indexes and join implementation, this operation in worst case, also requires $|input_i| \times |input_j|$ time and space. Inputs are retrieved not for every k_i but for all possible pair of keywords. This results in complexity $O(input_{max}^{C(\mathcal{K},2)-1})$, where $input_{max}$ here refers to the largest number of relationships that can be obtained for a keyword pair, and $C(\mathcal{K}, 2)$ is the number of 2-combinations of the set $\mathcal{K}$, denoting the number of joins that have to be processed.

While the number of operations are same for all KERG models, the size of $input_{max}$ varies. Clearly, the more coarse-grained the grouping, i.e., the higher the number of elements aggregated to one group at the summary level, the smaller will be $input_{max}$. In particular, we have $input_{max}(D\text{-}KERG) \leq input_{max}(S\text{-}KERG) \leq input_{max}(E\text{-}KERG)$. How much smaller a KERG summary is compared to one other depends on the data. In the extreme case where every data element mentions only distinct terms, i.e., does not share terms with one other, all KERG models are actually equal in size.

While KERG models require joins on input sets to be performed $C(\mathcal{K}, 2) - 1$ times, KS only needs $|K| - 1$ combinations of input sets. However, the advantage of using KERG is that the size of the input sets that have to be processed is

expected to be smaller. This is not only due to the effect of summarization. For all KERG models, inputs are retrieved for keyword pair while for KS, inputs are retrieved using single keywords. Two keywords are more selective than one keyword, thus more likely result in smaller input.

5 Evaluation

We implemented our approach for keyword query routing in Java using JDK 1.6 on top of MySQL 5.1. The experiments were conducted on a commodity PC with 2.5GHz Intel Core, 4GB of RAM and 500GB HDD SATA II 7200rpm, running on Windows 7. As discussed, while KRG [10] is limited to the problem of database selection, E-KERG can be seen as an extension that captures the ideas behind KRG. KS represents a naive baseline. The goal was to assess the performance of routing plan computation and the validity of results that can be achieved with S-KERG and D-KERG, compared to the baselines E-KERG and KS.

5.1 Data Preprocessing

We employed a chunk of RDF data part of the Billion Triple Challenge dataset[3]. It contains about 10M RDF triples that are from 154 different data sources, linked via 500K mappings.

In total, the number of distinct terms extracted from all sources was 121,434. We measured the number of elements in KS and the number of KERG relationships. This was done for different settings of d_{max} to investigate the changes in the number of relationships as longer distances are considered. KS contains 804,528 elements. For $d_{max} = 0, 1, 2, 3, 4$, E-KERG contains 2.4M, 7.7M, 364M, 616M and 889M relationships, S-KERG contains 1.8M, 5.1M, 144M, 215M and 312M relationships and D-KERG contains 1.7M, 4.7M, 141M, 203M and 279M relationships. Clearly, there were more relationships in KERG models than elements in KS. The number of relationships increases with d_{max}. The increase was particularly sharp (one order of magnitude) when changing d_{max} from 1 to 2.

Similar results were obtained for index size. The E-KERG index was the largest. As an average over different settings for d_{max}, S-KERG was about 36%, D-KERG was about 32% and KS was less than 1% the size of E-KERG. For $d_{max} = 2$ for instance, the sizes for E-KERG, S-KERG, D-KERG and KS were 8694MB, 3438MB, 3279MB and 22 MB, respectively.

Larger indexes required more building times. The times for building the S-KERG indexes for $d_{max} = 4, 3, 2, 1$ for instance, were 846 Min, 583 Min, 339 Min and 27 Min, respectively.

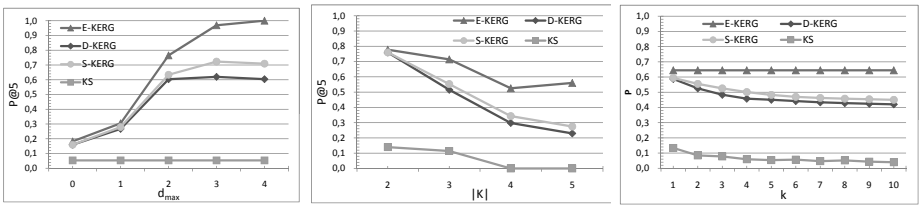
Our report [8] provides a breakdown of the results into 6 categories of datasets that vary in size. According to these results, both index size and building time increased with the size of the dataset. However, there is no strict correlation because there are cases where relatively small datasets resulted in large indexes. Rather, structural density was the dominant factor. Large index and high building costs were obtained for datasets which exhibit large number of links to other datasets, and contain nodes with large in- and outdegree.

[3] http://vmlion25.deri.ie/index.html

5.2 Query processing

For the experiment, we used a set of 30 keyword queries. All queries are valid, i.e., they produce non-empty keyword answers (4-Steiner graphs to be precise). For each query, at least two data sources contribute to the answers. One example submitted by participants is "Rudi AIFB ISWC2008". The sources containing partial answers to this are *uni-karlsruhe.de* and *semanticweb.org*. Other examples are "Town River America", "Markus Denny Semantic Wikis" and "Beijing Conference Database 2007". All queries can be found in our report [8].

Validity of Routing Plans To investigate the validity, we use precision at k ($P@k$) to measure the percentage of plans that are valid out of the top-k plans returned by the system. For instance, $P@10$ is 1 when every plan in the top-10 list returned by the system, produces at least one keyword query result.



(a) P@k at various d_{max} (b) P@k at various $|K|$ (c) P at various k

Fig. 4: Validity of the plans measured using $P@k$.

Fig. 4a shows $P@5$ for the settings $d_{max} = 0, 1, 2, 3, 4$. These values represent the average computed for all 30 queries. Using E-KERG, precision was up to 100 percent, i.e., for $d_{max}^{sum} = d_{max}^{data} = 4$. With $P@5$ being always above 0.6 when $d_{max} > 1$, S-KERG and D-KERG also achieved relatively good results. $P@5$ for KS was only 6%. Clearly, d_{max} had a positive effect. More valid plans were computed when a higher value was used for d_{max}. However, using $d_{max} = 4$ instead of 3 did not yield clear improvement.

Fig. 4b shows the effect of query length $|K|$. Quite clear, queries with larger number of keywords resulted in lower precision. It dropped as low as 0.23 when using D-KERG for queries with 5 keywords.

Fig. 4c shows that as more results from the system were taken into account (larger k), precision decreased. The decrease is small for k values larger than 3.

Experimental results thus correspond to the analysis we presented before: KS is the model that produces only very few valid plans. This result was improved by one order of magnitude when relationships between keywords were used. The more fine-grained a model captures the relationships, the larger was the percentage of valid plans. Even a summary at the level of sources produced reasonably high quality results, i.e., every second plan was a valid one.

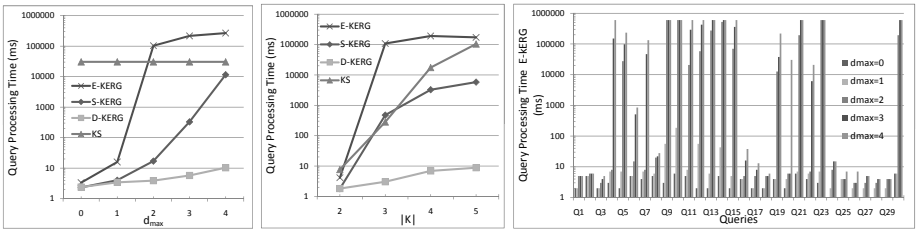
Performance Performance is measured as the average response time for computing routing plans. Fig. 5a shows the performance for queries at various settings using different values for d_{max}. This parameter had no effect on the KS's results but clearly influenced the performance achieved with KERG summaries. Times

increased with higher values for d_{max}. While this increase was sharp for E-KERG and S-KERG, time performance of D-KERG was relatively stable. In particular, time required by D-KERG was no more than 10ms on average.

Expectedly, more time was needed when the number of query keywords increases, as illustrated in Fig. 5b. It seems that all the other models had poor performance w.r.t complex queries but D-KERG. In particular, E-KERG is no longer affordable for queries with more than 2 keywords because it needed more than 100s to produce results. While the times shown are the actual times obtained for the other models, only the lower bound was shown for E-KERG. This is because we applied a timeout of 6min. Fig. 5c shows the exact times obtained for E-KERG and the queries that had to be aborted due to timeout. For $d_{max} = 4$ for instance, 1 out of every three queries was aborted.

Less expected, Fig. 5a+ 5b show that KS did not achieve good performance. It needed more than 30s on average, up to 100s for queries with 5 keywords.

This can be explained using the theoretical result achieved in the previous section. Namely, the poor performance of KS indicates that the number of elements (see $input_{max}$ in Section 4.4) retrieved for single keywords must have been much larger than for two keywords. In other words, keyword pairs proved to be the much more selective queries. Considering relationships between keywords thus did not only improve result validity but also performance.



(a) Times at var. d_{max} (b) Times at var. $|K|$ (c) Times for E-KERG at var. d_{max}

Fig. 5: Processing times.

6 Conclusion

We presented a solution to the novel problem of keyword query routing. It helps users without knowledge of the evolving linked data and schema to find combination of sources that contain answers corresponding to their needs. This solution also partially addresses the aspect of efficiency as queries can be then evaluated against the relevant sources identified by the user, instead of using the entire Web of linked data.

We have proposed a family of summary models. Through theoretical and experimental analysis, we showed that it is important to capture keyword relationships. Compared to the KS model representing the naive baseline that stores only single keywords, the KERG models relying on relationships could produce a much

larger number of valid results, i.e., improved precision by more than one order of magnitude when compared to the naive baseline represented by KS. Further, finding out which *relationships* are covered as opposed to single keywords resulted in less intermediate results to be processed. Thus, using relationships also has a positive effect on performance.

We could also show that *summarizing relationships* is essential for dealing with the large-scale linked data Web. Using a fine-grained E-KERG model representing an extension of work in database selection that captures all relationships in the data, precision was up to 100%, but response time was too high. While specific requirements shall determine what is the "best" model, it seems that D-KERG which summarizes at the level of sources represents the most practical trade-off. It produced results in less than 10ms out of which every second one was valid.

As future work, we will combine the proposed work on query routing with query processing to obtain a scalable procedure for computing relevant sources as well as retrieving the final answers from them.

Acknowledgements Research reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF) under the iGreen (grant 01A08005) and CollabCloud project (grant 01IS0937A-E).

References

1. A. Harth, A. Hogan, R. Delbru, J. Umbrich, S. O'Riain, and S. Decker. Swse: Answers before links! In *Semantic Web Challenge*, 2007.
2. A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich. Data summaries for on-demand queries over linked data. In *WWW*, pages 411–420, 2010.
3. A. Harth, J. Umbrich, A. Hogan, and S. Decker. Yars2: A federated repository for querying graph structured data from the web. In *ISWC/ASWC*, pages 211–224, 2007.
4. F. Liu, C. T. Yu, W. Meng, and A. Chowdhury. Effective keyword search in relational databases. In *SIGMOD Conference*, pages 563–574, 2006.
5. T. Neumann and G. Weikum. The rdf-3x engine for scalable management of rdf data. *VLDB J.*, 19(1):91–113, 2010.
6. M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano. Efficient keyword search across heterogeneous relational databases. In *ICDE*, pages 346–355, 2007.
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 *ICDE*, pages 405–416, 2009.
8. T. Tran and L. Zhang. Keyword query routing. Technical report, Karlsruhe Institute of Technology, 2010. `http://www.aifb.uni-karlsruhe.de/WBS/dtr/papers/kqueryrouting.pdf`.
9. G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig.ma: live views on the web of data. In *WWW*, pages 1301–1304, 2010.
10. Q. H. Vu, B. C. Ooi, D. Papadias, and A. K. H. Tung. A graph method for keyword-based selection of the top-k databases. In *SIGMOD Conference*, pages 915–926, 2008.
11. B. Yu, G. Li, K. R. Sollins, and A. K. H. Tung. Effective keyword-based selection of relational databases. In *SIGMOD Conference*, pages 139–150, 2007.