RESEARCH ARTICLE OPEN ACCESS

Approaches for Keyword Query Routing

Mrs. Suwarna Gothane, Srujana.P
M.E , Assitant professor, CMRTC
, M.Tech, CMRTC

Abstract-
The growing number of datasets published on the Web as linked data brings both opportunities for high data availability of data. As the data increases challenges for querying also increases. It is very difficult to search linked data using structured languages. Hence, we use Keyword Query searching for linked data. In this paper, we propose different approaches for keyword query routing through which the efficiency of keyword search can be improved greatly. By routing the keywords to the relevant data sources the processing cost of keyword search queries can be greatly reduced. In this paper, we contrast and compare four models – Keyword level, Element level, Set level and query expansion using semantic and linguistic analysis. These models are used for keyword query routing in keyword search.
Index terms: Keyword search, Keyword query routing, Graph-structured data, linguistic and semantic analysis

I. Introduction

The web is no longer a collection of textual data but also a web of interlinked data sources. One project that largely contributes to this develop- ment is Linking Open Data. Through this, a vast amount of structured information was made publicly available. Querying that huge amount of data in an intuitive way is challenging.

Collectively, Linked Data comprise hundreds of sources containing billions of RDF triples, which are connected by millions of links. While different kinds of links can be established, the ones frequently published are *sameAs* links, which denote that two RDF resources represent the same real-world object. The representation of the linked data on the web is shown in figure 1.

The linked data Web already contains valuable data in diverse areas, such as e-government, e-commerce, and the biosciences. Additionally, the number of available datasets has grown solidly since its inception. [1]

In order to search such data we use keyword search techniques which employ keyword query routing. To decrease the high cost incurred in searching structured results that span multiple sources, we propose routing of the keywords to the relevant databases. As opposed to the source selection problem [2], which is focusing on computing *the most relevant sources*, the problem here is to compute the *most relevant combinations of sources*. The goal is to produce routing plans, which can be used to compute results from multiple sources.

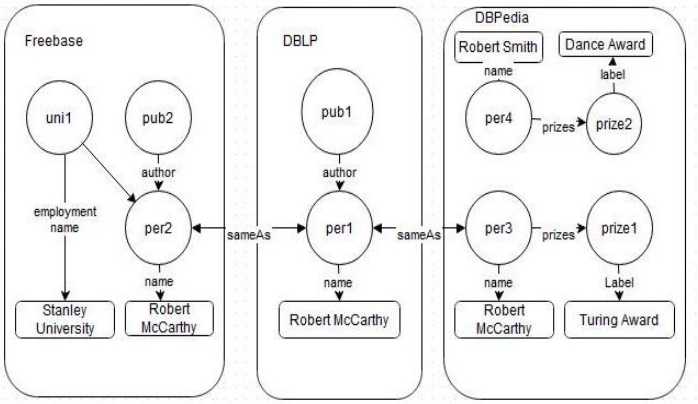

Figure 1: Example of Linked data on web

For selecting the correct routing plan, we use graphs that are developed based on the relationships between the keywords present in the keyword query. This relationship is considered at the various levels such as keyword level, element level, set level e.t.c.,

The rest of paper is organized as follows. Section 2 provides the brief outline on the existing work. The different approaches are listed along with the some examples explaining how the routing is considered in the section 3 before we conclude in the section 4.

II. Related work:

Keyword Query Search can be divided into two directions of work. They are: 1) keyword search approaches compute the most relevant structured results and 2) Solutions for source selection compute the most relevant sources.

2.1. Keyword search

In the keyword searching, we mainly follow two approaches. They are *schema-based approaches* and *schema-agnostic approaches*.

Schema-based approaches are implemented on top of off-the-shelf databases. A keyword is processed by mapping keywords to the elements of the databases, called *keyword elements*. Then, using the schema, valid join sequences are derived and are employed to join the computed keyword elements to form the candidate-networks that represent the possible results to the keyword query.

Schema-agnostic approaches operate directly on the data. By exploring the underlying graphs the structured results are computed in these approaches. Keywords and elements which are connected are represented using Steiner trees/graphs. The goal of this approach is to find structures in the Steiner trees.

For the query "Stanley Robert Award" for instance, a Steiner graph is the path between uni1 and prize1 in Fig. 1. Various kinds of algorithms have been proposed for the efficient exploration of keyword search results over data graphs, which might be very large. Examples are bidirectional search [3] and dynamic programming [4].

Recently, a system called Kite extends schema-based techniques to find candidate networks in the multi source setting [5]. It employs schema matching techniques to discover links between sources and uses structure discovery techniques to find foreign-key joins across sources. Also based on pre computed links, Hermes [6] translates keywords to structured queries.

2.2 Database Selection

In order to get the efficient results for keyword search, the selection of the relevant data sources plays a major role. The main idea is based on modeling databases using keyword relationships. A keyword relationship is a pair of keywords that can be connected via a sequence of join operations. For instance, (Stanley, Award) is a keyword relationship as there is a path between uni1 and prize1 in Fig. 1. A database is considered relevant if its keyword relationship model covers all pairs of query keywords.

M-KS considers only binary relationships between keywords. It incurs a large number of false positives for queries with more than two keywords. This is the case when all query keywords are pair wise related but there is no combined join sequence which connects all of them.

G-KS [7] addresses this problem by considering more complex relationships between keywords using a Keyword Relationship Graph (KRG). Each node in the graph corresponds to a keyword. Each edge between two nodes corresponding to the keywords (k_i, k_j) indicates that there exists at least two connected tuples $t_i \leftrightarrow t_j$ that match k_i and k_j. Moreover, the distance between t_i and t_j are marked on the edges.

III. Approaches

For routing the keywords to the relevant data sources and searching the given keyword query, we propose four different approaches. They are:

1) Keyword level model 2) Element level model, 3) Set level model, and 5) Query expansion using linguistic and semantic features.

We compute the *keyword query result* and *keyword routing plan* [11] which are the two important factors of keyword routing.

3.1 Keyword level model

In keyword level, we mainly consider the relationship between the keywords in the keyword query. This relationship can be represented using *Keyword Relationship Graph* (KRG) [7]. It captures relationships at the keyword level. As opposed to keyword search solutions, relationships captured by a KRG are not direct edges between tuples but stand for paths between keywords.

For database selection, KRG relationships are retrieved for all pairs of query keywords to construct a sub graph. Based on these keyword relationships alone, it is not possible to guarantee that such a sub graph is also a Steiner graph (i.e., to guarantee that the database is relevant). To address this, sub graphs are validated by finding those that contain Steiner graphs. This is a filtering step, which makes use of information in the KRG as well as additional information about which keywords are contained in which tuples in the database. It is similar to the exploration of Steiner graph in keyword search, where the goal is to ensure that not only keywords but also tuples mentioning them are connected.

However, since KRG focuses on database selection, it only needs to know whether two keywords are connected by some join sequences or not. This information is stored as relationships in the KRG and can be retrieved directly. For keyword search, paths between data elements have to be retrieved and explored. Retrieving and exploring paths that might be composed of several edges are clearly more expensive than retrieving relationships between keywords.

Keyword search over relational databases finds the answers of tuples in the databases which are connected through primary/foreign keys and contain query keywords. As there are usually large numbers of tuples in the databases, these methods are rather expensive to find answers by on-the-fly enumerating the connections.

To address this problem, proposed *tuple units* [8] to efficiently answer keyword queries. A *tuple unit* is

a set of highly relevant tuples which contain query keywords.

Definition-1 (**Tuple Units**): Given a database *D* with *m* connected tables, $R_1, R_2, \ldots, R_m$, for each tuple t_i in table R_i, let R_{ti} denote the table with the same primary/foreign keys as R_i, having a single tuple t_i. The joined result of table R_{ti} and other tables $R_j (j \neq i)$ based on foreign keys, denoted by $R = \bowtie_{j \neq i} R_j \bowtie R_{ti}$, is called a tuple set. Given two tuple sets t_1 and t_2, if any tuple in t_2 is contained in t_1, we call that t_1 covers t_2 (t_2 is covered by t_1). A tuple set is called a tuple unit if it is not covered by any tuple set.

To better understand the above definition, consider the following example.

Authors

AID	Name
a1	B.Ding
a2	B.Yu
a3	S.Pandit
a4	A.Doan
a5	LeKhac.H
a6	H.Den
a7	V.Hristidis
a8	P.Haase
a9	A.K.H.Tung

Author-paper

AID	PID
a1	p1
a2	p2
a3	p3
a4	p4
a5	p5
a6	p6
a7	p7
a8	p8
a9	p9

papers

PID	Title	Cont	Year
p1	Finding top-K min-cost connected trees in data base	ICDE	2007
p2	Effective keyword based Selection of relational databases	SIGMOD	2007
p3	Bidirectional expansion for keyword search on graph	VLDB	2005
p4	Efficient keyword search across relational databases	ICDE	2007
p5	Discoverer: keyword search in relational databases	VDLB	2002
p6	hermes: data web search	JWS	2009

Table 1: An example database

Example 1: Consider a publication database in Table 1. For each tuple in a table, we join the three tables and get the tuple sets as shown in Table 2. Tuple set Ta1 is not a tuple unit as it is covered by T_{p1}. T_{a2} is a tuple unit as any tuple set does not cover it. In this way, we can find all tuple units as shown in Table 2. Each tuple unit can represent a meaningful and integral information unit, and can be taken as an answer of a keyword query. Considering a keyword query {relational, database, keyword, search, Hristidis}, the underlined tuple unit T_{p5} (Table 2) contains all the input keywords. We can take this tuple unit as an answer. Note that we do not need to on-the-fly identify structural relationships between tuples in different tables, and the tuple-unit-based method can improve search performance.

3.2 Element level model

Keyword search [9] relies on an element-level model (i.e., data graphs) to compute keyword query results. Elements mentioning keywords are retrieved from this model and paths between them are explored to compute Steiner graphs. To deal with the keyword routing problem, elements can be stored along with the sources they are contained in so that this information can be retrieved to derive the routing plans from the computed keyword query results.

In this model, we mainly concentrate on IR technique of data retrieval. This technique allow users to search unstructured information using keyword based on scoring and ranking, and do not need users to understand any database schemas.

We use graph-based data models to characterize individual data models.

Definition 1 (Element-level Data Graph): An element-level data graph g (N, ε) consists of

- The set of nodes N, which is the disjoint union of N_ε N_V, where the nodes N_ε represent entities and the nodes N_V capture entities' attribute values, and
- The set of edges ε, subdivided by $\varepsilon = \varepsilon_R \rceil \varepsilon_A$, where ε_R represents inter entity relations, ε_A stands for entity-attribute assignments. We have e (n_1, n_2) 2 ε_R iff n_1; n_2 2 N_ε and e (n_1, n_2) 2 ε_A iff n_1 2 N_ε and n_2 2 N_V. The set of attribute edges ε_A (n) = {e (n, m) 2 ε_A} is referred to as the description of the entity n.

Note that this model resembles RDF data where entities stand for some RDF resources, data values stand for RDF literals, and relations and attribute

Table 2: Tuple sets and Tuple units

(a) For tuples in *Authors*

TID	AID	PID	Name	Title	···	
Ta_1	a1	p1	···	···	···	×
Ta_2	a2	p1	···	···	···	√
	a2	p2	···	···	···	
Ta_3	a3	p2	···	···	···	×
Ta_4	a4	p3	···	···	···	√
	a4	p4	···	···	···	
Ta_5	a5	p3	···	···	···	×
Ta_6	a6	p4	···	···	···	×
Ta_7	a7	p5	···	···	···	√
	a7	p6	···	···	···	
Ta_8	a8	p5	···	···	···	×
Ta_9	a9	p6	···	···	···	×

(b) For tuples in *Papers*

TID	AID	PID	Name	Title	···	
Tp_1	a1	p1	···	···	···	√
	a2	p1	···	···	···	
Tp_2	a2	p2	···	···	···	√
	a3	p2	···	···	···	
Tp_3	a4	p3	···	···	···	√
	a5	p3	···	···	···	
Tp_4	a4	p4	···	···	···	√
	a6	p4	···	···	···	
Tp_5	a7	p5	···	···	···	√
	a8	p5	···	···	···	
Tp_6	a7	p6	···	···	···	√
	a9	p6	···	···	···	

(c) For tuples in *Author-Paper*

TID	AID	PID	Name	Title	···	
Tap_1	a1	p1	···	···	···	×
Tap_2	a2	p1	···	···	···	×
Tap_3	a2	p2	···	···	···	×
Tap_4	a3	p2	···	···	···	×
Tap_5	a4	p3	···	···	···	×
Tap_6	a4	p4	···	···	···	×
Tap_7	a5	p3	···	···	···	×
Tap_8	a6	p4	···	···	···	×
Tap_9	a7	p5	···	···	···	×
Tap_{10}	a7	p6	···	···	···	×
Tap_{11}	a8	p5	···	···	···	×
Tap_{12}	a9	p6	···	···	···	×

correspond to RDF triples. While it is primarily used to model RDF Linked Data on the web, such a graph model is sufficiently general to capture XML and relational data. For instance, a tuple in a relational database can be modeled as an entity, and foreign key relationships can be represented as inter entity relations.

Existing keyword search solutions naturally apply to this problem. However, the data graph and the number of keyword elements are possibly very large in our scenario, and thus, exploring all paths between them in the data graphs is expensive. This is the main drawback of this model.

3.3 Set level model

In this model we derive the summary at the level of set of elements.

Definition 2 (Set-level Data Graph): A set-level data graph of an element-level graph g ($N_\varepsilon \uplus N_V$; $\varepsilon_R \uplus \varepsilon_A$) is a tuple $g' = (N', \varepsilon')$. Every node $n' \in N'$ stands for a set of element level entities $N_{n'} \subseteq N_\varepsilon$, i.e., there is mapping type: $N\varepsilon \rightarrow N'$ that associates every element-level entity $n \in N_\varepsilon$ with a set-level element $n' \in N'$. Every edge e' (n'_i, n'_j) $\in \varepsilon'$ represents a relation between the two sets of element-level entities n'_i and n'_j. We have $\varepsilon' = \{e' (n'_i, n'_j) \mid e (n_i, n_j) \in \varepsilon_R$, type $(n_i) = n'_i$; type $(n_j) = n'_j\}$.

This set-level graph essentially captures a part of the Linked Data schema on the web that is represented in RDFS, i.e., relations between classes. Often, a schema might be incomplete or simply does not exist for RDF data on the web. In such a case, a pseudo schema can be obtained by computing a structural summary such as a data guide [10].

A set-level data graph can be derived from a given schema or a generated pseudo schema. Thus, we assume a membership mapping type: $N_\varepsilon \rightarrow N'$ exists and use $n \in n'$ to denote that n belongs to the set n'. An example of the set level graph is given in Fig. 2. We consider the search space as a set of Linked Data sources, forming a web of data.

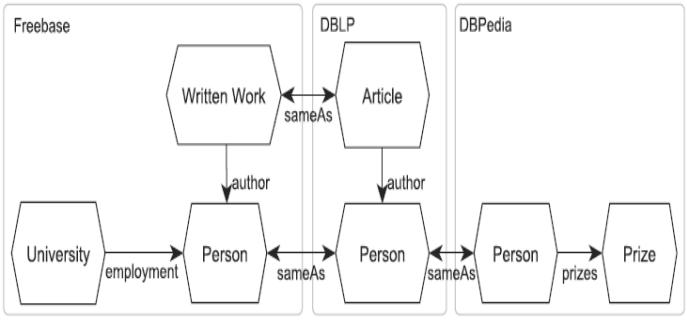

Fig 2: set-level web data graph

We develop a set level Keyword-Element Relationship Graph (KERG) [11].

Intuitively, a d_{max}-KERG represents all paths between keywords that are connected over a maximum distance dmax. This is to capture all d_{max}-Steiner graphs that exist in the data. The fig 3 below shows the set-level KERG with $d_{max} = 1$.

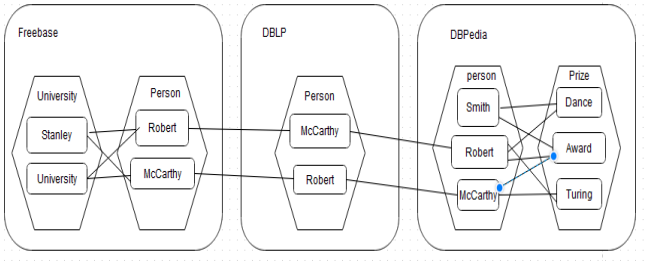

Fig 3: set-level KERG with $d_{max} = 1$

Example 1: A KERG for our running example with $d_{max} = 1$ is illustrated in Fig. 3. For instance, there is a keyword-element node (Robert, Person, DBPedia). Note that the relationship {(Robert, Person, DBPedia), (Award, Prize, DBPedia)} actually stands for the element-level connections {(Robert, per3, DBPedia), (Award, prize1, DBPedia)}, and {(Robert, per4, DBPedia), (Award, prize2, DBPedia)} because per3 and per4 mention Robert, prize1 and prize2 mention Award, per3, per4 $\in$ Person, prize1, prize2 $\in$ Prize, there is a path between per3 and prize1, and a path between per4 and prize2 (see web data graph in Fig. 1). This example illustrates that element-level relationships, which share the same pair of terms (Robert and Award), classes (Person and Prize), and sources (DBPedia and DBPedia) can be summarized to one single set-level relationship.

In order to compute the routing plan we use the following algorithm.

Algorithm1: PPRJ ComputeRoutingPlan(K, W'_K)

Input: The query K, the summary W'_K (N'_K, ε'_K)
Output: The set of routing plans [RP]
$JP \leftarrow$ a join plan that contains all $\{k_i, k_j\} \in 2^K$
$T \leftarrow$ a table where every tuple captures a join sequence of KERG relationships $e'_K \in \varepsilon'_K$, the score of each e'_K, and the combined score of the join sequence; it is initially empty;
While JP .empty() **do**
  $\{k_i, k_j\} \leftarrow JP$ .pop();
  $\varepsilon'_{\{ki, kj\}} \leftarrow$ retrieve(ε'_K, $\{k_i, k_j\}$);
  if T.empty() **then**
    $T \leftarrow \varepsilon'_{\{ki, kj\}}$;
  else
    $T \leftarrow \varepsilon'_{\{ki, kj\}} \bowtie T$;
Compute score of tuples in T via SCORE (K, W^S_K);

[*RP*] ← Group *T* by sources to identify unique combinations of sources;
Compute scores of routing plans in [RP] via SCORE (*K*, *RP*);
Sort [*RP*] by score;

3.4 Query expansion using linguistic and semantic features

In document retrieval, many query expansion techniques are based on information contained in the top-ranked retrieved documents in response to the original user query, e.g. [12], [15]. Similarly, our approach is based on performing an initial retrieval of resources according to the original keyword query. Thereafter, further resources are derived by leveraging the initially retrieved ones.

Overall, the proposed process depicted in Figure 4 is divided into three main steps. In the first step, all words closely related to the original keyword are extracted based on two types of features – linguistic and semantic. In the second step, various introduced linguistic and semantic features are weighted using learning approaches. In the third step, we assign a relevance score to the set of the related words. Using this score we prune the related word set to achieve a balance between *precision* and *recall*.

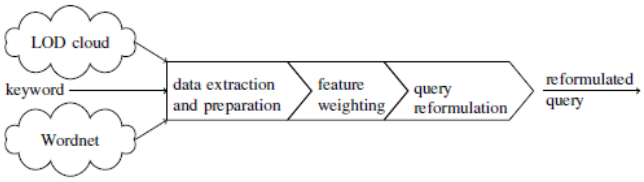

Fig 4: AQE pipeline

A. Extracting and Preprocessing of Data using Semantic and Linguistic Features:

For the given input keyword *k*, we define the set of all words related to the keyword *k* as $X_k = \{x_1, x_2, \ldots, x_n\}$. The set X_k is defined as the union of the two sets, LE_k and SE_k. LE_k is constructed as the collection of all words obtained through linguistic features and in the same manner the semantic features also.

Linguistic features extracted from WordNet are:

• *Synonyms:* words having similar meanings to the input keyword *k*.
• *Hyponyms:* words representing a specialization of the input keyword *k*.
• *Hyponyms:* words representing a generalization of the input keyword *k*.

The set *SE* comprises all words semantically derived from the input keyword *k* using Linked Data. These semantic features are defined as the following semantic relations:

• *sameAs:* deriving resources having the same identity as the input resource using owl:sameAs.
• *seeAlso:* deriving resources that provide more information about the input resource using rdfs:seeAlso.
• *class/property equivalence:* deriving classes or properties providing related descriptions for the input resource using owl:equivalentClass and owl:equivalentProperty.
• *superclass/-property:* deriving all super classes/properties of the input resource by following the rdfs:subClassOf or rdfs:subPropertyOf property paths originating from the input resource.
• *subclass/-property:* deriving all sub resources of the input resource r_i by following the rdfs:subClassOf or rdfs:subPropertyOf property paths ending with the input resource.
• *broader concepts:* deriving broader concepts related to the input resource r_i using the SKOS vocabulary properties skos:broader and skos:broadMatch.
• *narrower concepts:* deriving narrower concepts related to the input resource r_i using skos:narrower and skos:narrowMatch.
• *related concepts:* deriving related concepts to the input resource r_i using skos:closeMatch, skos:mappingRelation and skos:exactMatch.

For each $r_i \in AP_k$, we derive all the related resources employing the above semantic features. Then, for each derived resource r', we add all the English labels of that resource to the the set SE_k. Therefore, SE_k contains the labels of all semantically derived resource.

The set of all related words of the input keyword *k* is defined as $X_k = LE_k \bigcup SE_k$. After extracting the set X_k of related words, we run the following preprocessing methods for each $x_i \in X_k$:

1) *Tokenization:* extraction of individual words, ignoring punctuation and case.
2) *Stop word removal:* removal of common words such as articles and prepositions.
3) *Word lemmatisation:* determining the lemma of the word.

For example, as can be observed in Figure 5, the word "thinking machine" and "electronic brain" is derived by *synonym*, *sameAs* and *equivalent* Relations

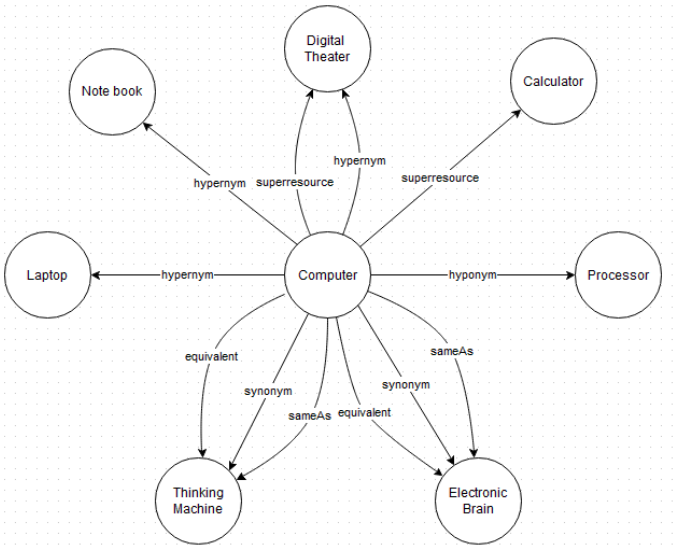

Fig 5: Exemplary expansion graph of the word computer using semantic features.

B. Feature Selection and Feature Weighting

In order to distinguish how effective each feature is and to remove ineffective features, we employ a weighting schema *ws* for computing the weights of the features as $ws : f_i \in F \rightarrow w_i$. Note that F is the set of all features taken into account. There are numerous feature weighting methods to assign weight to features like information gain [13], weights from a linear classifier [14], odds ratio, etc. Herein, we consider two well-known weighting schemas.

1) Information Gain (IG): Information gain is often used to decide which of the features are the most relevant. We define the information gain (IG) of a feature as:

$$\mathrm{IG}(f_i) = \sum_{\substack{c\in\{+,-\} \\ f_i\in\{present,absent\}}} \Pr(c, f_i) \ln \frac{\Pr(c, f_i)}{\Pr(f_i)\Pr(c)}$$

2) Feature weights from linear classifiers: Linear classifiers, such as for example SVMs, calculate predictions by associating the weight w_i to the feature f_i. Features whose w_i is close to 0 have a small influence on the predictions. Therefore, we can assume that they are not very important for query expansion.

C. Setting the Classifier Threshold

As a last step, we set the threshold for the classifiers above. To do this, we compute the relevance score value score (x_i) for each word $x_i \in X_k$. Naturally, this is done by combining the feature vector $V_{xi} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ and the feature weight vector $W = [w_1, w_2, \ldots, w_n]$ as follows:

$$\mathrm{score}(x_i) = \sum_{i=1:n} \alpha_i w_i$$

IV. Conclusion and Future Scope

Keyword query search is a widely used approach for retrieving linked data in an efficient manner. In order to reduce the high cost of searching, we redirect the keywords to the relevant data sources. Here we use keyword routing to redirect the keywords. This is done using different types of approaches. Here, we discussed the four approaches of keyword query evaluation to get the desired results. We use graph based methods to compute the routing plans. Further, we show that when routing is applied to an existing keyword search system to prune sources, substantial performance gain can be achieved.

References

[1] T. Berners-Lee, "Linked Data Design Issues," 2009;www.w3.org/DesignIssues/LinkedData.html

[2] B. Yu, G. Li, K.R. Sollins, and A.K.H. Tung, "Effective Keyword-Based Selection of Relational Databases," Proc. ACM SIGMOD Conf., pp. 139-150, 2007.

[3] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional Expansion for Keyword Search on Graph Databases," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), pp. 505-516, 2005.

[4] B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Top-K Min-Cost Connected Trees in Databases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 836-845, 2007.

[5] M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano, "Efficient Keyword Search Across Heterogeneous Relational Databases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 346-355, 2007.

[6] T. Tran, H. Wang, and P. Haase, "Hermes: Data Web Search on a Pay-as-You-Go Integration Infrastructure," J. Web Semantics, vol. 7, no. 3, pp. 189-203, 2009.

[7] 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," Proc. ACM SIGMOD Conf., pp. 915-926, 2008.

[8] Jianhua Feng, Guoliang Li and Jianyong Wang, "Finding Top-k answers in keyword search over relational databases using tuple units" IEEE transactions, VOL. 23 NO. 12, December 2011.

[9] G. Li, B.C. Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data," Proc. ACM SIGMOD Conf., pp. 903-914, 2008.
[10] R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases," Proc. 23rd Int'l Conf. Very Large Data Bases (VLDB), pp. 436-445, 1997.
[11] Thanh Tran and Lei Zhang, "Keyword Query Routing" IEEE Transactions, VOL.26, NO.2, February 2014.
[12] K. Collins- Thompson, Reducing the risk of query expansion via robust constrained optimization. In *CIKM*. ACM, 2009.
[13] H. Deng, G. C. Runger, and E. Tuv. Bias of importance measures for multi-valued attributes and solutions. In *ICANN (2)*, volume 6792, pages 293–300. Springer, 2011.
[14] D. Mladenic, J. Brank, M. Grobelnik, and N. Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. *In Proceedings of the 27th Annual International ACM SIGIR Conference SIGIR2004*. ACM, 2004.
[15] Saeedeh Shekarpour, Jens Lehmann, and Sören Auer, "Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features" IEEE Seventh International Conference on Semantic Computing, 2013.