Keyword-driven SPARQL Query Generation Leveraging Background Knowledge

Saeedeh Shekarpour*, Sören Auer*, Axel-Cyrille Ngonga Ngomo*,Daniel Gerber*,
Sebastian Hellmann*, Claus Stadler*
* Universität Leipzig, Institut für Informatik, D-04103 Leipzig, Germany,
{lastname}@informatik.uni-leipzig.de

***Abstract*—The search for information on the Web of Data is becoming increasingly difficult due to its dramatic growth. Especially novice users need to acquire both knowledge about the underlying ontology structure and proficiency in formulating formal queries (e. g. SPARQL queries) to retrieve information from Linked Data sources. So as to simplify and automate the querying and retrieval of information from such sources, we present in this paper a novel approach for constructing SPARQL queries based on user-supplied keywords. Our approach utilizes a set of predefined basic graph pattern templates for generating adequate interpretations of user queries. This is achieved by obtaining ranked lists of candidate resource identifiers for the supplied keywords and then injecting these identifiers into suitable positions in the graph pattern templates. The main advantages of our approach are that it is completely agnostic of the underlying knowledge base and ontology schema, that it scales to large knowledge bases and is simple to use. We evaluate 17 possible valid graph pattern templates by measuring their precision and recall on 53 queries against DBpedia. Our results show that 8 of these basic graph pattern templates return results with a precision above 70%. Our approach is implemented as a Web search interface and performs sufficiently fast to return instant answers to the user even with large knowledge bases.**

***Keywords*-keyword search; graph pattern; SPARQL query**

I. Introduction

With the dramatic growth of the Linked Data Web (currently amounting to 27 Billion triples[1]) it is increasingly difficult for end users to find the information they are looking for. Services such as *Sindice* [23], *Sig.ma* [22], *Swoogle* [5] or *Watson* [4] offer simple search services[2], but are either restricted to the retrieval of single RDF documents or in the case of Sig.ma to the retrieval of information about a single entity from different sources. Some services (such as e. g. http://lod.openlinksw.com) on the other hand load the complete Data Web into a large triple store cluster and enable issuing SPARQL queries on top of it. However, in order to express their information needs in terms of SPARQL queries, users have to (a) understand the SPARQL concepts, (b) understand the SPARQL syntax (in absence of a a visual query builder) and (c) know what information structures are actually available in order to formulate queries that also return results. To enable lay users to access the Data Web, it becomes necessary to *simplify the access to the Data Web* by providing search interfaces that resemble the search interfaces commonly used on the document-oriented Web. However, because queries based on natural language (NL) are inherently ambiguous, their precise interpretation is extremely challenging. While SPARQL queries permit to express unambiguously which entities and relations are relevant for the query, keyword-based search as implemented in current web search engines does not permit the explicit expression of relations. Services such as *Poweraqua*[3][16] demand from the user to enter a question in natural language, but this is often inconvenient because most users prefer to obtain information by the lowest number of keywords[4]. Another obstacle to the realization of this approach lies in the sheer size of the Data Web, which requires very efficient and scalable query processing algorithms.

In this paper, we propose a novel approach for generating SPARQL queries based on user-supplied keywords. Our approach presupposes the availability of background knowledge in the form of a set of Linked Data sources upon which the user wants her search to be carried out. Based on a set of user-supplied keywords, we first compute a list of candidate IRIs (Internationalized Resource Identifier) for each of the keywords issued by the user. In a second step, we restrict the set of valid IRIs to those which are related to each other via a link in the background knowledge. Finally, we use the filtered set of IRIs to generate SPARQL queries that aim to encompass the semantics of the query supplied by the user. We currently use DBpedia as background knowledge, but the approach is easily transferable to the whole Data Web. Since the approach is based on simple operations, it can generate and execute SPARQL queries very efficiently. Another advantage of this approach is that it is completely agnostic of the underlying knowledge base as well as its ontology schema.

This paper is organized as follows: In Section II we present the background definitions and an overview of the approach along with our method for choosing candidate IRIs. The subsequent Section III introduces all possible graph pattern templates for pairs of IRIs. In Section IV we describe our approach to the construction of SPARQL queries based on graph pattern templates. We elaborate on our experimental setup and the selection of graph patterns, as well as analyze our results in Section V. The related work is reviewed in Section VI. We close with concluding remarks and an outlook

[1] http://www4.wiwiss.fu-berlin.de/lodcloud/state/ (March 4th, 2011)

[2] These systems are available at: http://sindice.com, http://sig.ma, http://swoogle.umbc.edu, http://kmi-web05.open.ac.uk/WatsonWUI

[3] http://poweraqua.open.ac.uk:8080/poweraqua2

[4] http://www.keyworddiscovery.com/keyword-stats.html

on future work in the last section.

II. Approach

A. Preliminaries

The basic assumption underlying our approach is that natural language queries cannot always be converted into formal queries automatically. This is due to the meaning of some of the query elements being either unknown, ambiguous, or implicit. For example, in the query 'Which are the islands in Germany?', the relation between *Germany* and *islands* is indicated by *are*, but the precise relationship is *are located in*. The problem of mapping a user query to a formal query gets even more complex when the user uses keywords instead complete natural language queries, because even more information is omitted. In addition, experience with classical search engines shows that users prefer to enter the lowest possible number of keywords in order to retrieve information related to their query. For example, the query mentioned above would be naturally expressed with the keywords *Germany* and *islands*.

In the context of the Semantic Web, the expected answer to a query is usually a set of RDF resources linked by certain relations (representing a connected graph). Consequently, the second assumption underlying our approach is that user-supplied keywords must play the role of anchors points (i.e., nodes or edges of the RDF graph) that are to be used to retrieve knowledge from the background knowledge via some form of bootstrapping. We illustrate the difficulties of the bootstrapping processing with the following example:

Example 1. *Consider two keywords "Germany" and "island" used with the intention to search for the list of Germany's islands. The suitable SPARQL query is:*

```
SELECT * WHERE {
  ?island  a   dbo:Island .
  ?island  ?p  dbp:Germany .
}
```

Some desired answers to be retrieved are:

```
1: db:Rettbergsaue a           dbo:Island .
   db:Rettbergsaue dbp:country dbr:Germany .
2: db:Sylt         a           dbo:Island .
   db:Sylt         dbp:country dbr:Germany .
3: db:Vilm         a           dbo:Island .
   db:Vilm         dbp:country dbr:Germany .
4: db:Mainau       a           dbo:Island .
   db:Mainau       dbp:country dbr:Germany .
```

Here, we encounter two issues. First, we need to find a set of IRIs corresponding to each keyword. Second, we have to construct suitable triple patterns based on the anchor points extracted previously so as to retrieve appropriate data. These goals are achieved by the approach presented in the following.

B. Terminology and Definitions

We call an IRI matching to a keyword an *anchor point*. The process of selecting a relevant neighborhood for each of the anchor points is called *induction* (also known as *relation discovery*). Note, that most semantic search approaches (e.g. [16, 11, 18, 24, 17]) perform induction first on the ontology level to extract appropriate graph pattern templates, and then apply those templates to the instance level. We, however, do not separate *induction* in the ontology level from the instance level since ontology statements are usually available either in the knowledge base or via Linked Data de-referencing as RDF triples. Consequently, instances and ontology statements are connected based on `rdf:type` properties, allowing our *induction* not to have to separate between ontology and instance knowledge.

Formally, we base our approach on the following definitions:

Definition 1 (Keyword set)**.** *We define the set of user-supplied keywords as* $K = \{K_1, K_2, ..., K_n\}$.

Definition 2 (Knowledge base signature)**.** *The knowledge base signature* KBS *is represented by* $KBS = (C, I, P)$ *, where* C *denotes the set of classes,* I *denotes the instances of these classes and* P *denotes the set of properties used in the relationships between classes or instances (also including datatype properties).*

Definition 3 (Connected query result)**.** *A single connected query result denoted* $R = \{(s, p, o) | (s, p, o)$ *a triple*$\}$*, consists of a set of triples which are connected through common subjects or objects, i.e.:*

$$(|R| \leq 1) \vee (\forall (s_1, p_1, o_1) \in R : \exists (s_2, p_2, o_2) |$$

$$(s_2 = s_1 \vee s_2 = o_1 \vee o_2 = o_1 \vee o_2 = s_1))$$

These sets of triples express sentences which represent a sort of integrated information around the user keywords.

C. Overview

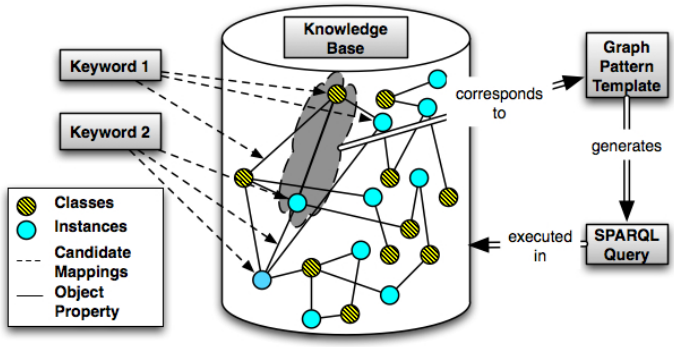

Fig. 1. Overview of the proposed method.

Figure 1 shows an overview of our approach. Our approach firstly retrieves relevant IRIs related to each user-supplied keyword from the underlying knowledge base and secondly injects them to a series of graph pattern templates for constructing formal queries. So as to find these relevant IRIs, the following two steps are carried out:

1) Mapping keywords to IRIs: The goal of this function is the retrieval of entities that match with the user-supplied keywords. Matching entities and keywords is carried out by applying a string similarity function on the keywords and the

label properties of all entities in the knowledge base. This similarity evaluation is carried out on all types of entities (i.e., classes, properties and instances). As a result, for each keyword, we retrieve a list of IRI candidates, i.e. *anchor points*.

Definition 4 (Mapping function)**.** *Let K be the set of user-supplied keywords. The mapping function $M : K \rightarrow 2^{C \cup I \cup P}$ applies the sub-string similarity measure on each $K_i \in K$ and on the `rdfs:label` of all IRIs in our underlying knowledge base and returns the set $AP_{K_i} \subseteq C \cup I \cup P$ (where C, I and P are the sets of classes, instances and properties contained in the knowledge base respectively), whose labels contain K_i as a sub-string or are equivalent to K_i.*

2) Ranking and Selecting Anchor Points: This step aims at excluding anchor points which are probably unrelated to any interpretation of the user keyword; thereby reducing the potentially high number of anchor points to a minimum. This reduction is carried out by applying a ranking method over the string similarity score and the connectivity degree of the previously detected IRIs in each AP_{K_i}. For each $u \in AP_{K_i}$ a *specificity score*, denoted by S, is defined based on two parameters, i. e. a *string similarity score* and a *connectivity degree*.

The string similarity score σ calculates the similarity of the `rdfs:label` of $u \in AP_{K_i}$ and of the keyword K_i by measuring the normalized edit distance between $u_{rdfs:label}$ and K_i. As the query we use for retrieving IRIs guarantees that K_i is a substring of u_{label}, computing the edit distance between these two strings is equivalent to computing the difference in their length. We normalize the *string similarity score* of each label by using the *max-min normalization method* to compute similarity values between 0 and 1. Consequently, $\sigma(u_{label}, K_i) = 1$ means that the two strings are equal. Formally,

$$\sigma(u_{label}, K_i) = 1 - \frac{|u_{label}| - \min\limits_{v \in AP(K_i)} |v_{label}|}{\max\limits_{v \in AP(K_i)} |v_{label}| - |K_i|} \quad (1)$$

We also compute a simplified approximation of the *connectivity degree* $CD(u)$ for each $u \in AP_{K_i}$ by counting how often u occurs in the triples of the knowledge base. It is important to note that IRIs with type `class` and `property` have higher CD values. In DBpedia, for example, classes have an average connectivity degree of 14,022, while properties have in average 1,243 and instances 37.

The specificity S for each $u \in AP(K_i)$ is finally calculated as follows:

$$S(u) = \sigma(u_{label}, K_i) \times \log(CD(u)) \quad (2)$$

Definition 5 (Ranking and selection function)**.** *The ranking and selection function RS maps AP_{K_i} to the set U_{K_i} as top-10 of the IRIs contained in AP_{K_i} sorted in descending order based on $S(u)$ where $u \in AP_{K_i}$.*

III. Graph Pattern Templates

Throughout the paper, we use the standard notions of the RDF [5] and SPARQL [6] specifications, such as *graph pattern*, *triple pattern* and *RDF graph*. The SPARQL queries generated with our approach are a restricted kind of SPARQL queries, since they use only *basic graph patterns* without blank nodes. We analysed 1,000 distinct queries from the query log of the public DBpedia endpoint[7] and learned that the number of IRIs is usually larger than the number of triple patterns occurring in the query. As a consequence of this finding we decided to assume graph patterns for generating SPARQL queries for two user-supplied keywords to consist of either one or two triple patterns.

Definition 6 (Graph pattern template)**.** *Let H be a set of placeholders and V be a set of variable identifiers being disjoint from each other and from $C \cup I \cup P$. A graph pattern template is defined as $GPT = \{(s, p, o) | (s \in V \cup H) \wedge (p \in V \cup H) \wedge (o \in V \cup H)\}$ that contains exactly two placeholders. Two triple patterns being part of the same graph pattern template have to share a common subject or object. In our triple pattern templates, a placeholder can stand either for a property (when occurring in the predicate position), an instance (when occurring at subject or object position) or a class (when occurring at the object position) depending on its position. After replacing the placeholders in a graph pattern template with the detected IRIs, a graph pattern with triple patterns of the form $(V \cup I) \times (V \cup P) \times (V \cup C \cup I)$ is obtained.*

Note that our notion of graph pattern templates is a slight restriction of the SPARQL basic graph patterns in the general case, since our definition does not consider blank nodes and restricts the set of possible IRIs at a certain position in the triple pattern. Definition 6 leads to the 17 possible graph pattern templates as shown in Table I. In this table, we subdivided the patterns in different categories, depending on whether they map instances to instances, classes to instances etc. Symbols preceded by question marks denote variables while symbols without question marks are placeholders which will be replaced by IRIs referring to the identified anchor points.

Example 2. *After applying mapping and ranking functions on the user keywords (from Example 1), we obtain two identified IRIs, i.e. http://dbpedia.org/ontology/Island with the type class and http://dbpedia.org/resource/Germany with the type instance. The possible graph pattern templates for these two IRIs are:*

1) *(?island, a, dbo:Island), (?island, ?p, dbr:Germany)*
2) *(?island, a, dbo:Island), (dbr:Germany, ?p, ?island)*

As detailed in Section V, we performed an accuracy study on all combinatorial possible graph pattern templates. This study showed that the patterns contained in Table II limit the

[5] http://www.w3.org/TR/rdf-schema/

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

[7] The DBpedia SPARQL endpoint is available at: http://dbpedia.org/sparql/ and the query log excerpt at: ftp://download.openlinksw.com/support/dbpedia/.

Category	Possible Patterns	Pattern Schema
Instance-Property (IP)	**IP.P1**	$(s, p, ?o)$
	IP.P2	$(?s, p, o)$
	IP.P3	$(?s_1, ?p_1, o_1)(?s_1, p_2, ?o_2)$
	IP.P4	$(?s_1, ?p_1, o_1)(?o_2, p_2, ?s_1)$
	IP.P5	$(s_1, ?p_1, ?o_1)(?s_2, p_2, ?o_1)$
	IP.P6	$(s_1, ?p_1, ?o_1)(?o_1, p_2, ?o_2)$
Class-Instance (CI)	**CI.P7**	$(?s_1, a, c)(?s_1, ?p_1, o_1)$
	CI.P8	$(?s_1, a, c)(s_2, ?p_1, ?s_1)$
Instance-Instance (II)	**II.P9**	$(s, ?p, o)$
	II.P10	$(s, ?p_1, ?x)(?x, ?p_2, o)$
	II.P11	$(s_1, ?p_1, ?x)(s_2, ?p_2, ?x)$
	II.P12	$(?s, ?p_1, o_1)(?s, ?p_2, o_2)$
Class-Property (CP)	CP.P13	$(?s, a, c)(?s, p, ?o)$
	CP.P14	$(?s, a, c)(?x, p, ?s)$
Property-Property (PP)	PP.P15	$(?s, p_1, ?x)(?x, p_2, ?o)$
	PP.P16	$(?s_1, p_1, ?o)(?s_2, p_2, ?o)$
	PP.P17	$(?s, p_1, ?o_1)(?s, p_2, ?o_2)$

TABLE I
CATEGORIZATION OF ALL POSSIBLE GRAPH PATTERN TEMPLATES FOR EACH TYPED PAIR OF PLACEHOLDERS.

Category	Possible Patterns	Pattern Schema
Instance-Property(IP)	IP.P1	$(s, p, ?o)$
	IP.P4	$(?s_1, ?p_1, o_1)(?o_2, p_2, ?s_1)$
	IP.P6	$(s_1, ?p_1, ?o_1)(?o_1, p_2, ?o_2)$
Class-Instance(CI)	CI.P7	$(?s_1, a, c)(?s_1, ?p_1, o_1)$
	CI.P8	$(?s_1, a, c)(s_2, ?p_1, ?s_1)$
Instance-Instance(II)	II.P9	$(s, ?p, o)$
	II.P10	$(s, ?p_1, ?x)(?x, ?p_2, o)$
Class-Property(CP)	CP.P14	$(?s, a, c)(?x, p, ?s)$
Property-Property(PP)	-	-

TABLE II
APPROPRIATE IDENTIFIED GRAPH PATTERN TEMPLATES.

search space (thus leading to more efficiency) without reducing the accuracy of our approach significantly. Consequently, we only considered these patterns during the SPARQL-query generation process described below.

IV. SPARQL QUERY GENERATION

Algorithm 1 outlines the procedure for generating SPARQL queries based on the graph pattern templates shown in Table II. After selecting the top ranked IRIs based on Definition 5, according to the type of each pair of IRIs issued from the cross-product of U_{K_i}, a set of suitable graph pattern templates is selected from Table II for generating SPARQL queries.

Example 3. *For the pair of IRIs http://dbpedia.org/resource/Germany and http://dbpedia.org/ontology/Island, our algorithm would generate the following two queries:*

```
1) SELECT * WHERE {
     ?island  a   dbo:Island .
     ?island  ?p  dbr:Germany . }
2) SELECT * WHERE {
     ?island     a   dbo:Island .
     dbr:Germany ?p  ?island . }
```

The results of this algorithm are the output of our approach. To validate the approach, we implemented it as a Java Web application which is publicly available at: http://lod-query.aksw.org. A screenshot of the search results is shown in Figure 2. The whole query interpretation and processing is performed typically on average in 15 seconds (while first results are already obtained after one second) when using DBpedia as knowledge base.

Data: K Keyword Set, knowledge base KB
Result: A set of connected query results
foreach *keyword* K_i **do**
  retrieve AP_{K_i};
  sort AP_{K_i};
  $RS(AP_{K_i})$ = top-10 ranked IRIs from AP_{K_i};
end
foreach $u \in RS(AP_{K_i})$ & $u' \in RS(AP_{K_j})$ **do**
  switch *Category of* u, u' **do**
    case *Instance-Property*
      query(IP.P1,u, u');
      query(IP.P4,u, u');
      query(IP.P6,u, u');
    case *Class-Instance*
      query(CI.P7,u, u');
      query(CI.P8,u, u');
    case *Class-Property*
      query(CP.P14,u, u');
    case *Instance-Instance*
      query(II.P9,u, u');
      query(II.P10,u, u');
  endsw
end

Algorithm 1: Query generation algorithm. The function `query` constructs the query based on the query pattern given as first argument and the entity identifier to placeholder mapping supplied as 2nd and 3rd argument.

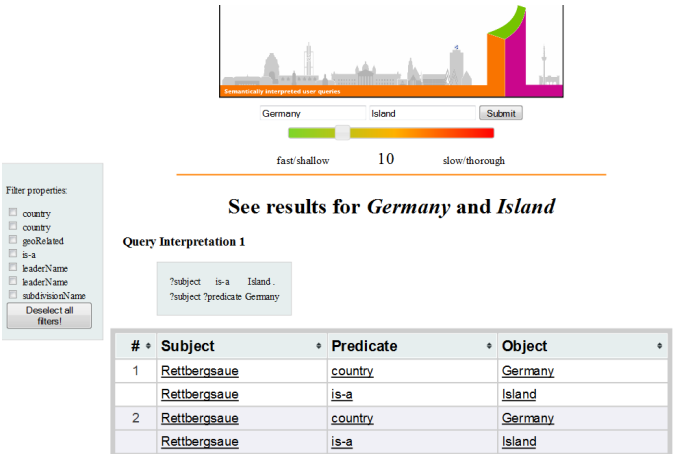

Fig. 2. Example query in GUI available at lod-query.aksw.org for the search keywords *germany* and *island*.

Table III shows some samples of keywords for which the application is capable to retrieve suitable results. These keywords were categorized based on the type of queries which they can answer in three categories, i.e. similar instances, characteristics of an instance and associations between instances.

V. EVALUATION

This section is divided to three parts. First, we introduce the accuracy metrics used for evaluation. Second, we outline the results of an accuracy study on all valid graph pattern

Keywords	Answers
Instance characteristics.	
Kidman spouse	`d:Kidman dp:spouse Keith Urban .`
Iran language	`d:Iran dp:Language d:Persian_language .`
Titanic length	`d:RMS_Titanic dp:Length 268.8336 .`
Capital China	`d:Republic_of_China dp:capital Beijing .`
Michelangelo death	`1. d:Michelangelo dp:deathDate "1564-02-18" .` `2. d:Michelangelo dp:deathPlace "Rome, Italy" .`
Associations between instances.	
Obama Clinton	`d:Obama dp:predecessor d:Bush . d:Bush dp:predecessor d:Clinton .`
Volkswagen Porsche	`d:Volkswagen_Group dp:subsidiary d:Volkswagen .`
Similar instances.	
Facebook Person	`d:Facebook dp:keyperson d:Sheryl Sandberg . d:Sheryl Sandberg a d:Person .`

Germany Island	`1. d:Germany dp:Islands d:Rgen . d:Rgen a do:Island .`
	`2. d:Germany dp:Islands d:Fhr . d:Fhr a do:Island .`
	`3. d:Germany dp:Islands d:Sylt . d:Sylt a do:Island .`

Lost Episode	`1. d:Raised_by_Another dp:series dbp:Lost . d:Raised_by_Another a do:TVEpisode .`
	`2. d:Homecoming dp:series dbp:Lost . d:Homecoming a do:TVEpisode .`
	`3. d:Outlaws dp:series dbp:Lost . d:Outlaws a do:TVEpisode .`

English Country	`1. d:Ghana dp:officialLang d:English_language . d:Ghana a do:Country .`
	`2. d:Cameroon dp:officialLang d:English_language . d:Cameroon a do:Country .`
	`3. d:UK dp:officialLang d:English_language . d:UK a do:Country .`

TABLE III
SAMPLES OF KEYWORDS AND RESULTS.

templates introduced in Table I with the aim of selecting those templates that lead to a high accuracy. Finally, we evaluate our whole application by using the metrics presented in the following subsection.

A. Accuracy Metrics

In this section we first motivate and explain the metrics that are fundamental to our evaluation. Since the user's intention in keyword-based search is ambiguous, judging the correctness of the retrieved answers is a challenging task. Let us consider the following example:

Example 4. *Given the keywords France and President the following RDF graphs (i.e.* answers*) are presented to the user:*

```
1. Nicolas_Sarkozy    nationality  France .
   Nicolas_Sarkozy    a            President .

2. Felix_Faure        birthplace   France .
   Felix_Faure        a            President .

3. Yasser_Arafat      deathplace   France .
   Yasser_Arafat      a            President .
...
```

The input of the user can be interpreted in at least two ways: 1. Who is the current president of France? 2. Who are the people that have ever been presidents of France? Depending on the meaning intended by the users, these patterns can be considered as being accurate or not. If the second interpretation is correct, then *Felix Faure*, who was the president of France from 1895 to 1899, is a correct answer, else it is not. We only consider those answers correct that meet our original intention whereas all other ones are considered incorrect. According to this criterion the correct answers are (1) and (2). However, among the correct answers note the difference in the involved predicates, namely *birthplace* and *nationality*. An observation is that we can draw a distinction between whether an answer contains statements relevant to our search intention and whether these statements are the preferred ones. We will measure the preference of an answer based on the occurring RDF terms. RDF terms (short *terms*) comprise each individual subject, object or predicate in the triples of the answer. In our example, we prefer *nationality* over *birthplace* because theoretically a person born in one country may be president in a different one, however, it is very unlikely that a president has a different nationality than the country he is president in.

Therefore, besides distinguishing between answers related to different interpretations, we should also differentiate between pure answers (just containing preferred terms) and those which contain some impurity. In fact, the correctness of an answer is not a bivalent value but based on the user's perception. As such it may vary between completely irrelevant and exactly correct. In essence for evaluation, we investigate two questions: 1) For how many of the keyword queries do the templates yield answers at all with respect to the original intention? 2) If answers are returned, how correct are they?

Therefore, we introduce the *Correctness Rate (CR)* as a measure for the preference of certain RDF terms. This metric allows a user to rate the correctness of each individual answer based on its own perception.

Definition 7 (Correctness rate)**.** *For an individual answer a for a query q, we define $CR_q(a)$ as the fraction of correct (preferred) RDF terms occurring in it.*

$$CR_q(a) = \frac{|correct\ terms|}{|total\ terms|}$$

Based on the CR for individual answers, we can derive the average CR (ACR) for a set of answers:

Definition 8 (Average CR)**.** *For a given set of answers A of a query q, we define $ACR_q(A)$ as the arithmetic mean of the CRs of its individual answers.*

$$ACR_q(A) = \frac{1}{|A|} * \sum_{a \in A} CR_q(a)$$

The ACR is the basis for the *fuzzy precision* metric (FP), which measures the overall correctness of a template's corresponding answers A_q with respect to a set of keyword queries

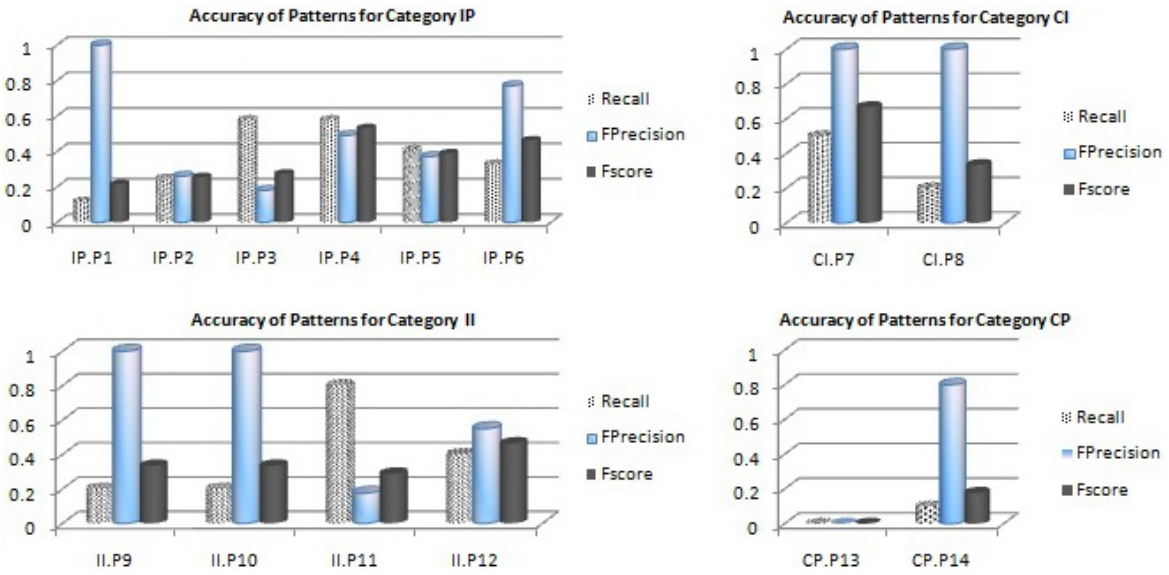

Fig. 3. Accuracy of each categorized graph pattern.

Q.

$$FP = \frac{\sum_{q \in Q} ACR_q(A_q)}{|queries\ with\ answers|}$$

By using the fuzzy precision, we can now measure the quality of the results returned by each individual graph pattern template. The rationale between our measurements is that a template is not required to contribute answers to the set of all answers (as other templates of the same corresponding category may compensate for that. However, if answers are provided, they are subject to correctness evaluation.

We also measured the recall as the fraction of keyword queries for which answers were found:

$$Recall = \frac{|queries\ with\ answers|}{|total\ queries|}$$

Finally, we use the following definition of the F-Score:

$$F = 2 * \frac{FP * R}{FP + R}$$

B. Accuracy Evaluation of Possible Graph Pattern Templates

Since we are interested in using those graph pattern templates which typically result in precise answers with respect to the user intention of keywords, we evaluated the accuracy of each graph pattern template by running a SPARQL query containing each individual graph pattern template introduced in Table I by injecting a series of IRI pairs. We selected 53 natural language queries of *TREC 9* from which we extracted the two main keywords conveying the general meaning. For example, the query *'How many people live in Chile?'* can be expressed by the keywords *Chile* and *population*. Thereafter, the mapping function was applied to these keywords and from the retrieved IRIs, the most suitable ones were *manually* selected and assigned to the related dataset with regard to their type. We used DBpedia 3.5.1 [12] as the underlying knowledge base. After preparing the datasets, we performed a series of SPARQL queries for each single graph pattern template over the corresponding dataset. The results of the SPARQL queries along with the keywords were shown to two evaluators to score the CR metric for each individual answer. After rating CR for all retrieved answers related to a graph pattern template, $fuzzy\ precision$, $recall$ and $F-score$ were computed. Figure 3 shows the accuracy of each graph pattern template based on these three metrics. In the category Property-Property, the number of retrieved answers for all graph pattern templates was zero.

Our results show that some pattern templates such as P1 in the Instance-Property category as well as P7 and P8 in the Instance-Class category have a high fuzzy precision while their recall is low. In the case of P11 from the Instance-Instance category we have a high recall while the fuzzy precision is low. Hence, this graph pattern template generates a large number of irrelevant answers.

We discarded all templates with a fuzzy precision of less than 0.5, resulting in an increase of the overall precision and only a small loss in recall. We monitored the ACR for a set of queries before and after the reduction of graph pattern templates in the category IP and II, because most reductions occurred there. In the category IP, all queries with ACR higher than 0.4 and in the category II with ACR higher than 0.6 were properly answered with the same accuracy. So, this reduction maintained precise results (i.e. high ACR value).

As an interpretation of graph pattern templates, we present different scenarios in which a user is interested in retrieving different kinds of information. This categorization is based on the matter of information which is retrieved from the knowledge base.

Finding special characteristics of an instance: Datatype properties which emanate from instances/classes to literals or simple types and also some kinds of object properties state characteristics of an entity and information around them. So, in the simplest case of a query, a user intends to retrieve specific information of an entity such as *"Population of Canada"* or

"Language of Malaysia". Since this information is explicit, the simple graph patterns IP.P1, IP.P4 and IP.P6 can be used for retrieving this kind of information.

Finding similar instances: In this case, the user asks for a list of instances which have a specific characteristic in common. Examples for these type of queries are: *"Germany Island"* or *"Countries with English as official language"*. A possible graph structure capturing potential answers for this query type is depicted in Figure 4. It shows a set of instances from the same class which have a certain property in common. Graph pattern templates CI.P7, CI.P8, and CP.P14 retrieve this kind of information.

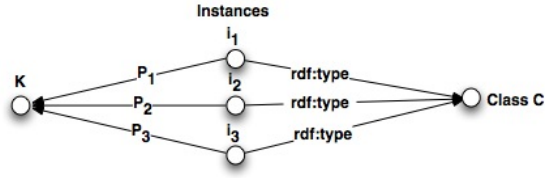

Fig. 4. Similar instances with an instance in common.

Finding associations between instances: Associations between instances in knowledge bases are defined as a sequence of properties and instances connecting two given instances (cf. Figure 5). Therefore, each association contains a set of instances and object properties connecting them which is the purpose of the user query. As an example, the query *Volkswagen Porsche* can be used to find associations between the two car makers. The graph pattern templates II.P9 and II.P10 extract these associations.

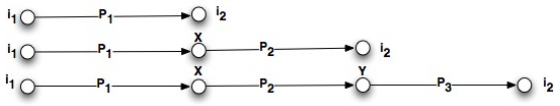

Fig. 5. Associations between two instances.

C. Application Evaluation

In this step, we evaluated the approach based on the three previously defined metrics, i.e. fuzzy precision, recall and f-score. The experimental setup consisted of giving a novice user 40 queries from TREC 9, and asking him to run each of the queries against DBpedia using our prototype implementation. Then, for each single answer of a query, he assigned CR according to his own intention. Subsequently, fuzzy precision and recall were computed based on the user's ratings. Note, that since hyperlinks among pages are inserted as `wikilink` in DBpedia and they do not convey special meaning between resources, we rempved all triples containing the IRIs http://dbpedia.org/property/wikilink. Table IV shows the evaluation results after running 40 queries against DBpedia. The overall precision of our system is 0.72.

Essentially, the accuracy of this method, specifically *recall*, does not depend on using suitable graph pattern templates, because on the one hand, the mapping approach for choosing relevant IRIs significantly influences the results, and on the other hand the quality of the data in DBpedia severely affects the accuracy. For example, the query *"Greece population"* returns the correct answer while the similar query *"Canada population"* led to no results.

Category	**Recall**	**Fuzzy precision**	**F-score**
General accuracy	0.625	0.724	0.670
Similar instances	0.700	0.735	0.717
Characteristics of an instance	0.625	0.700	0.660
Associations between instances	0.500	0.710	0.580

TABLE IV
ACCURACY RESULTS.

In addition to the overall evaluation, in order to make a comparison between functionality of the approach for different types of queries (i.e. finding special characteristics of an instance, finding similar instances and finding associations between instances) the employed queries were categorized based on their type and a separate evaluation was computed for each type. Our evaluation in Table IV shows the precision does not differ significantly for different types of queries, while the recall is type dependent. For instance, in the category *"similar instances"* the recall is significantly higher rather than in the category *"association between instances"*.

VI. Related Work

With the advent of the Semantic Web, information retrieval and question answering approaches were adapted for making use of ontologies. We can roughly divide related work into ontology-based information retrieval, ontology-based question answering and keyword search on structured data.

Ontology-based information retrieval: Approaches falling into this category annotate and index documents using a background ontology. The retrieval process is subsequently carried out by mapping user query terms onto these semantic document annotations. The approaches described in [3, 8, 19] are examples of this paradigm. All these approaches use background knowledge to enhance the retrieval accuracy, however, they do not utilize the background knowledge for semantically answering user queries.

Ontology-based question answering: Approaches falling into this category take a natural language question or a keyword-based query and return matching knowledge fragments drawn from the knowledge base as the answer. There are two different methods: (1) Using linguistic approaches for extracting complete triple-based patterns (including relations) from the user query and matching these triples to the underlying ontology (e.g. AquaLog [16] and OntoNL [11]). (2) Detecting just entities in the user query and discovering relations between these entities by analysing the knowledge base. Examples for this second group are KIM [18] and OntoLook [13] and [20, 6, 24, 17]. In these two approaches the RDF data is considered to be a directed graph and relations among entities are found through sequences of links (e.g. using graph traversal). Sheth [21] introduced the term *semantic association* for describing meaningful and complex relations between entities. Our work differs from these approaches, since it is completely independent of the underlying schema.

Furthermore, schema information is in our approach just *implicitly* taken into account, so a complex induction procedure is not required.

Keyword search on relational and XML data: With the beginning of the millennium, research on keyword search on relational and XML data attracted research interest. Meanwhile there exist many approaches such as [1], [10], [9], [14] for the relational domain and [7], [15], [2] for the XML domain. Especially the relational domain is relevant to our work due to the similarities to the RDF datamodel. All these approaches are based on *schema graphs* (i.e. a graph where tables and their primary-foreign key relations are represented as nodes and edges, respectively). In our work, we do not rely on an explicitly given schema, which is often missing for datasets on the Web of Data. However, achieving sufficient performance for instant query answering is more an issue in the RDF case, which is why our approach is currently limited to two keywords.

VII. Conclusion and Future Work

We see this work as a first step towards the user-friendly querying of the Data Web using rich semantic structures. By tightly intertwining the keyword interpretation and query generation with the available background knowledge we are able to obtain results of relatively good quality. We applied a number of techniques such as an thorough analysis of potential graph pattern so as to limit the search space and enable instant question answering. A problem, however, beyond our control is the data quality and coverage. Currently our evaluation is still limited to 150M facts comprised by DBpedia, but due to the generic nature and efficiency of the approach we will be able extend it quickly to the whole Data Web. For doing so, we aim to apply some optimizations original to our approach, since we currently use just a plain SPARQL interface. These optimizations will, for example, comprise the pre-computation of views and statistical summaries for each of our graph pattern templates. A current limitation is the restriction to two keywords. The rational behind this was to restrict the search space of possible query interpretations, in order to return the most meaningful results to the user quickly in a compact form. There are a number of possible advancements: (a) to allow a larger number of keywords or (b) to enable users to refine obtained queries and to add additional constraints and (c) to make more extensive use of linguistic features and techniques.

Acknowledgments

We would like to thank our colleagues from AKSW research group for their helpful comments and inspiring discussions during the development of this approach. This work was supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943) and by the Eurostars Project SCMS (E!4604).

VIII. References

*Bibliography

[1] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16. IEEE Computer Society, 2002.

[2] Liang Jeff Chen and Yannis Papakonstantinou. Supporting top-k keyword search in xml databases. In Feifei Li, Mirella M. Moro, Shahram Ghandeharizadeh, Jayant R. Haritsa, Gerhard Weikum, Michael J. Carey, Fabio Casati, Edward Y. Chang, Ioana Manolescu, Sharad Mehrotra, Umeshwar Dayal, and Vassilis J. Tsotras, editors, *ICDE*, pages 689–700. IEEE, 2010.

[3] Fabio Crestani. Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev.*, 11(6):453–482, 1997.

[4] M. D'aquin, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez, and D. Guidi. Toward a new generation of semantic web applications. *Intelligent Systems, IEEE*, 23(3):20–28, 2008.

[5] Li Ding, Timothy W. Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In David A. Grossman, Luis Gravano, ChengXiang Zhai, Otthein Herzog, and David A. Evans, editors, *CIKM*, pages 652–659. ACM, 2004.

[6] Ramanathan V. Guha, Rob McCool, and Eric Miller. Semantic search. In *WWW*, pages 700–709, 2003.

[7] Lin Guo, Feng Shao, Chavdar Botev, and Jayavel Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In Alon Y. Halevy, Zachary G. Ives, and AnHai Doan, editors, *SIGMOD Conference*, pages 16–27. ACM, 2003.

[8] Markus Holi and Eero Hyvnen. Fuzzy view-based semantic search. In *ASWC*, volume 4185 of *LNCS*, pages 351–365. Springer, 2006.

[9] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. Efficient ir-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.

[10] Vagelis Hristidis and Yannis Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, pages 670–681. Morgan Kaufmann, 2002.

[11] Anastasia Karanastasi, Alexandros Zotos, and Stavros Christodoulakis. The OntoNL framework for natural language interface generation and a domain-specific application. In *Digital Libraries: Research and Development, First International DELOS Conference, Pisa, Italy*, pages 228–237. 2007.

[12] Jens Lehmann, Chris Bizer, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[13] Yufei Li, Yuan Wang, and Xiaotao Huang. A relation-based search engine in semantic web. *IEEE Trans. Knowl. Data Eng.*, 19(2):273–282, 2007.

[14] Fang Liu, Clement T. Yu, Weiyi Meng, and Abdur Chowdhury. Effective keyword search in relational databases. In Surajit Chaudhuri, Vagelis Hristidis, and Neoklis Polyzotis, editors, *SIGMOD Conference*, pages 563–574. ACM, 2006.

[15] Ziyang Liu and Yi Chen. Reasoning and identifying relevant matches for xml keyword search. *PVLDB*, 1(1):921–932, 2008.

[16] Vanessa Lopez, Victoria S. Uren, Enrico Motta, and Michele Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *J. Web Sem.*, 5(2):72–105, 2007.

[17] Xiaomin Ning, Hai Jin, Weijia Jia, and Pingpeng Yuan. Practical and effective ir-style keyword search over semantic web. *Inf. Process. Manage.*, 45(2):263–271, 2009.

[18] Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, and Miroslav Goranov. Kim - semantic annotation platform. *Journal of Natural Language Engineering*, 10(3-4):375–392, September 2004.

[19] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi de Aragão. A hybrid approach for searching in the semantic web. In *WWW*, pages 374–383. ACM, 2004.

[20] Guus Schreiber, Alia Amin, Lora Aroyo, Mark van Assem, Victor de Boer, Lynda Hardman, Michiel Hildebrand, Borys Omelayenko, Jacco van Osenbruggen, Anna Tordai, Jan Wielemaker, and Bob Wielinga. Semantic annotation and search of cultural-heritage collections: The MultimediaN E-Culture demonstrator. *Journal of Web Semantics*, 6(4):243–249, November 2008.

[21] Amit Sheth, Boanerges Aleman-Meza, I. Budak Arpinar, Christian Halaschek-Wiener, Cartic Ramakrishnan, Yashodhan Warke Clemens Bertram, David Avant, F. Sena Arpinar, Kemafor Anyanwu, and Krys Kochut. Semantic association identi?cation and knowledge discovery for national security applications. *Journal of Database Management*, 16(1):33–53, 2005.

[22] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. Sig.ma: Live views on the web of data. *J. Web Sem.*, 8(4):355–364, 2010.

[23] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: weaving the open linked data. pages 552–565, 2007.

[24] Gideon Zenz, Xuan Zhou, Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. From keywords to semantic queries – incremental query construction on the semantic web. *Web Semantics*, 7(3):166–176, 2009.