AGGREGO SEARCH: Interactive Keyword Query Construction

Grégory Smits[1], Olivier Pivert[1], Hélène Jaudoin[1] and François Paulus[2]
[1]ENSSAT-IRISA, University of Rennes 1
Lannion, France
{gregory.smits, olivier.pivert, helene.jaudoin}@irisa.fr
[2]SEMSOFT
Rennes, France
francois.paulus@semsoft-corp.com

ABSTRACT

AGGREGO SEARCH offers a novel keyword-based query solution for end users in order to retrieve precise answers from semantic data sources. Contrary to existing approaches, AGGREGO SEARCH suggests grammatical connectors from natural languages during the query formulation step in order to specify the meaning of each keyword, thus leading to a complete and explicit definition of the intent of the search. An example of such a query is *name of person at the head of company and author of article about "business intelligence"*. In order to help users formulate such connected keywords queries, a specific autocompletion strategy has been developed. A translation of the user keyword query into SPARQL is performed on-the-fly during the interactive query construction process. For this demonstration, we show how AGGREGO SEARCH has been integrated on top of a mediation system to let users intuitively define explicit and precise keyword queries in order to extract knowledge distributed in heterogeneous large semantic data sources.

1. INTRODUCTION

Social networks have boosted the need for efficient and intuitive query interfaces to access large scale knowledge graphs whose semantics is defined by means of ontologies. The buzz around the launches of Google Knowledge Graph[1] and Facebook Graph Search[2] clearly illustrates how crucial that issue is. Before getting a great deal of media attention, this issue has been largely addressed by the scientific community where interesting approaches have been developed to propose keyword-based access to structured data stored in XML documents [1], relational databases [7] or ontologies [10, 3]. In opposition to previously cited approaches where queries are just flat enumerations of keywords, some works have been dedicated to the enrichment of the expressivity of query interfaces especially using natural language processing techniques [2]. In the spirit of the approaches proposed in [6, 4], AGGREGO SEARCH is half-way between natural language queries and classical keyword queries. An AGGREGO SEARCH query is composed of keywords referencing classes and instances of an ontology but also includes connectors referencing properties that clearly express how these concepts have to be linked. Connectors are very useful to express and determine the meaning of a keyword query and thus the way it has to be translated into a SPARQL query. An example of such a connected keyword query is: *name* **of** *person* **at the head of** *company* **and author of** *article* **about** *"business intelligence"* where the grammatical connectors in bold clearly disambiguate the keywords they link. Contributions of this demonstration prototype that implements the approach introduced in [8] on top of an integration architecture are twofold: i) the definition of a template-based formalism to connected keyword queries and ii) the development of an efficient and suitable autocompletion system to help users make the most of this novel keyword query system and makes it possible to translate a keyword query into a SPARQL query. The rest of the article is organized as follows. Section 2 briefly presents how connected keyword queries are internally formalized, whereas Section 3 points out the crucial role of the autocompletion strategy in this work. Finally, Section 4 describes the context of the scenario proposed for the demonstration, which is characterized by the fact that the translated SPARQL query is submitted to a mediation system connected to distributed heterogeneous data sources.

[1]http://www.google.com/insidesearch/features/search/knowledge.html
[2]https://www.facebook.com/about/graphsearch

EDBT '14 Athens, Greece

2. CONNECTED KEYWORDS QUERIES

2.1 Data Structure and Query Vocabulary

AGGREGO SEARCH offers a query interface to a triplestore whose semantics is defined by means of an RDFS ontology. The structure of the searchable data is defined by an RDFS ontology that is composed of *rdf:classes* used to divide *resources*, *rdfs:properties* that link a subject resource to an object resource, and instances of classes. Figure 1 gives an example of an ontology about business organizations, employees and news articles.

Searchable classes and instances of the ontology are linked by an *rdfs:label* property to at least one instance of *rdfs:Literal* that gives a human readable description of the concerned resource, possibly with synonyms. Each property linked to a searchable class of the ontology is associated with three *rdfs:label* properties, one attached to the property itself to describe its meaning, and two others attached respectively to its domain and range, so as to be able to explain this property from both sides. Textual literals linked by *rdfs:label* properties to searchable elements of the ontology form the query vocabulary that can be used inside AGGREGO SEAR-

10.5441/002/edbt.2014.62

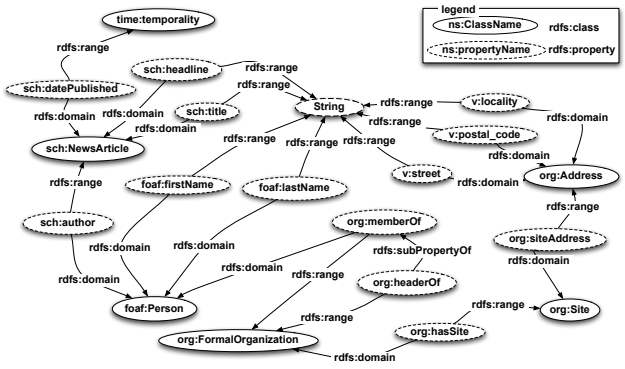

Figure 1: Extract of an ontology

CH keyword queries. Whereas classical approaches interpret keywords corresponding to classes and instances only, AGGREGO SEARCH also handles keywords associated with properties as an expression of the way classes and instances have to linked in the target SPARQL query. A valid connected keyword query corresponds to a connected subgraph of the ontology. Considering the query introduced in Section 1, Figure 2 illustrates the matching performed between the keywords (corresponding to *rdfs:label* properties) appearing in the query and searchable elements of the ontology. For the sake of clarity, only *rdfs:label* properties concerned by the query are informally specified into rectangles.

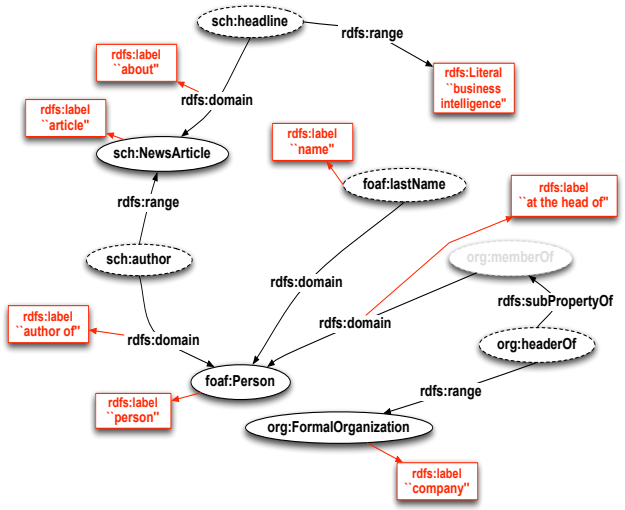

Figure 2: Graph view of the query *name of person at the head of company and author of article about "business intelligence"*

The vocabulary that can be used inside an AGGREGO SEARCH query is thus composed of:

- labels of classes hereafter called *CLASSNAME*,
- labels of properties hereafter called *PROPNAME*,
- labels of property ranges hereafter called *RANCON*,
- labels of property domains hereafter called *DOMCON*,
- *rdfs:Literal* instances hereafter called *VALUES*,
- and additional logical connectors like *and* and some tool words like *of* that make the query more natural and explicit (see Section 2.2).

2.2 Structured Query

The goal of the AGGREGO SEARCH system is to translate a user-defined keyword query into a SPARQL query. A SPARQL query is composed of a projection part introduced by the keyword SELECT and a selection part introduced by the keyword WHERE. The projection clause is used to declare the variables on which matching patterns defined in the selection clause are applied when querying the graph. For example, the keyword query illustrated in Figure 2 is translated into SPARQL as follows:

```
SELECT DISTINCT ?name
WHERE {
  ?name rdf:type foaf:lastName.
  ?person rdf:type foaf:Person. ?person foaf:name ?name.
  ?person org:headerOf ?comp. ?person sch:author ?art.
  ?art rdf:type sch:NewsArticle. ?art sch:headline ?head.
  FILTER regex(?head, "business intelligence") }
```

In AGGREGO SEARCH, it is assumed that keyword queries are composed of a first projection part where the expected information is specified, and a second optional selection part where filtering criteria are defined. A context free grammar has been defined to determine the patterns that may composed a valid AGGREGO SEARCH query. This grammar $G = (terminals, nonTerminals, startSymbol, rules)$ is defined as follows:

- $terminals$ uses the searchable vocabulary as the set of terminal symbols,
- $nonTerminals$ is composed of the symbols $\{query, select, where, selectElmt, whereElmt\}$,
- the $startSymbol$ is $query$,
- the production $rules$ are given hereinbelow in a Backus Naur Form.

query :: = select where | select
select ::= selectElmt 'and' select | selectElmt select | selectElmt
where ::= whereElmt 'and' where | whereElmt where | whereElmt
selectElmt ::= propNameList 'of' CLASSNAME | CLASSNAME
propNameList ::= PROPNAME 'and' propNameList | PROPNAME
whereElmt ::= RANCON CLASSNAME | DOMCON CLASSNAME | DOMCON VALUE | DOMCON RANCON VALUE

According to this grammar, the structure of the query introduced in Figure 2 may be represented by the derivation tree illustrated in Figure 3.

Notice that some labels of properties attached to a domain or a range involving a datatype (string, integer, real, etc.) are tagged in order to indicate that they have to be interpreted as a SPARQL filter. Such filters may also be explicitly used inside a selection statement of the type *DOMCON RANCON VALUE* as in: *has title contains "SPARQL"*, where *has title* may be a label attached to the domain of property *sch:title* and *contains* is attached to its range.
The current grammar covers a limited number of query patterns only. As explained in Section 3, this is not a problem in practice as this grammar is used to guide an autocompletion system only and not to syntactically validate the structure of freely-typed user queries.

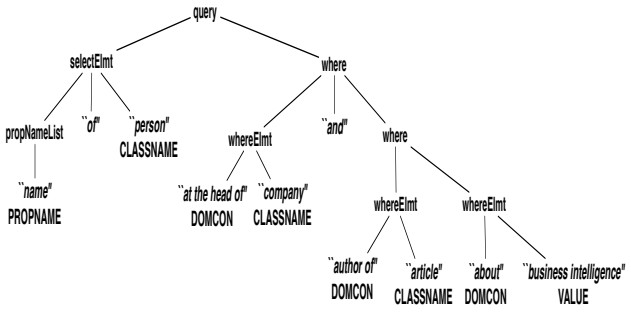

Figure 3: Derivation tree of the query from Fig. 2

3. AUTOCOMPLETION SYSTEM

As illustrated in Figure 5, the AGGREGO SEARCH query interface is composed of a single field. Currently, this field is not completely freely editable by the user, but may only be used with an autocompletion mechanism that plays a crucial role as it:

- helps users define connected keyword queries whose structure is covered by the grammar,
- builds its SPARQL translation on-the-fly.

Autocompletion of Structured Queries

Suggestions made by the autocompletion system are guided by the grammar detailed in Section 2.2. Compared with classical autocompletion systems embedded in search engines, AGGREGO SEARCH suggests completions that are relevant regarding not only the first letters typed by the user but also regarding the structure of the query whose construction is in progress. The algorithm of the autocompletion system simply follows the graph representation of the grammar (Figure 4). Starting with the initial node *query*, the autocompletion system suggests all the vocabulary elements corresponding to the category of the nodes directly connected to the current node. Thus, if the query field is empty (which means that the current node is *query*), then names of searchable properties (*PROPNAME*) and classes (*CLASSNAME*) are suggested, which are the two categories of elements that introduce a projection statement and thus may start a query. When a suggested element is selected or discriminated by its first letters typed by the user, then one progresses in the graph to the neighbors of the current node.

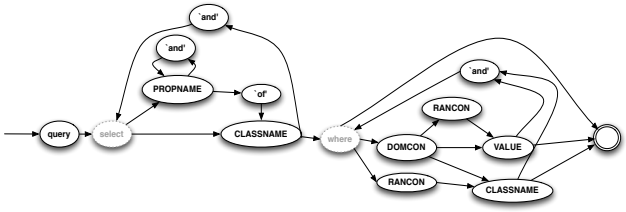

Figure 4: Graph representation of the grammar

As explained in Section 4, this keyword query approach is implemented in a particular context of a mediated-access to distributed and heterogeneous sources. This context implies that the data stored in the different sources is not available for local indexing, this is why the autocompletion system suggests elements of the vocabulary that correspond to labels of the reference domain ontology. Thus, if the structure being suggested by the autocompletion system involves a value, the user is invited to type this value as a quoted string. In the example illustrated in Figure 2, the label *about* associated with the domain of the property *sch:headline*, which is itself linked with the class *sch:article*, introduces a value. This is why a quoted string is opened directly after the selection by the user of the keyword *about* to let him type a description of the headline he/she is interested in.

It is worth noticing that, in case of ambiguities between suggested completions, addition information is displayed to explain which element the ambiguous suggestion is related to. If one considers two ambiguous labels, e.g. *name*, attached to two different classes, e.g. *foaf:Person* and *org:FormalOrganization*, then instead of suggesting the property label *name* twice, complete projection statements are displayed as *name <of Person>* and *name <of Company>*. This last point of disambiguation is all the more important when it concerns selection statements starting with ambiguous labels of domain properties whose range is linked to a string element. In order to precisely transcribe the sense of the keyword query into SPARQL, it is mandatory to attach each selection statement to its right variable introduced in the projection clause or a previous selection statement.

Finally, to obtain a reduced and relevant list of suggestions only, the autocompletion system makes the most of the fact that a SPARQL query is a connected subgraph of the ontology. Thus, when the system suggests the construction of a new selection statement, only labels attached to the domain or range of already activated classes are proposed, where suggestions about the last activated class are proposed first. For example, if one considers *headline of article written by person* as the current state of the query, then only labels of domains of properties linked to the classes *sch:NewsArticle* and *foaf:Person* are suggested as illustrated in Figure 5.

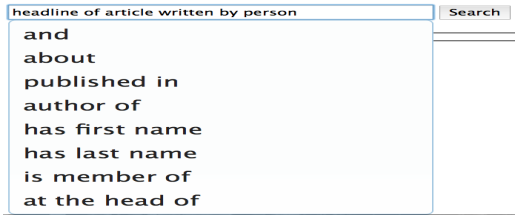

Figure 5: Suggestion of relevant completions

On-the-Fly Translation to SPARQL

During query construction using the autocompletion system, a translation into SPARQL is performed on-the-fly. As soon as a projection statement is completed by the user, the *SELECT* clause of the target SPARQL query is completed. A translation rule is indeed associated with each of the two covered projection statements. For the first one (*propNameList PROJCON CLASSNAME*), a variable is created for each property enumerated in the *propNameList* as well as a link with a variable representing the class explicitly defined as the *CLASSNAME* element of the rule. Variables representing classes (resp. properties) involved in the query are stored in an array of activated classes (reps. properties). These arrays are used by the autocompletion system to suggest the construction of selection statements on these activated classes and properties only. For the second type of covered projection statement that contains the name of a class only, one completes this statement with all the properties linked to this class. For example, the projection statement described only by *article* is completed with *title and published date*

and headline and author and mentioned persons of article that is translated into:

```
SELECT DISTINCT ?title ?published_data ?headline ?author
WHERE {
  ?title rdf:type sch:title. ?headline rdf:type sch:headline.
  ?published rdf:type sch:datePublished. ?author rdf:type sch:author.
  ?article rdf:type sch:NewsArticle.
  ?article sch:title ?title. ?article sch:datePublished ?published.
  ?article sch:headline ?headline. ?article sch:author ?author.
```

As selection statements concern classes and properties already mentioned in the query and referenced in the arrays of activated classes and properties, their translation into SPARQL is straightforward. For example, if a class say Person is associated with the variable ?person in the array of activated classes, then the selection statement *author of article* corresponding to the pattern *DOMCON CLASSNAME* is translated into:

```
?article ref:type sch:NewsArticle. ?person sch:author ?article.
```

Thus, using such an autocompletion strategy, one has the guarantee that all the submitted queries are semantically valid wrt. the ontology and return the exact answers to the user.

4. DEMONSTRATION

The keyword query approach presented in this article has been integrated on top of an integration system called AGGREGO server developed by the company SEMSOFT[3]. As shown in Figure 6, AGGREGO server relies on a mediation layer associated with a domain ontology corresponding to the searchable ontology and adaptors to make the integration of distributed and heterogeneous data sources easier [9]. SPARQL queries submitted to this mediation layer are rewritten in terms of views [5] describing the schemas of the searchable data sources.

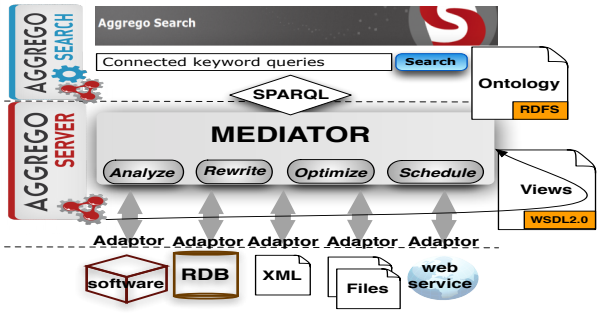

Figure 6: Architecture of AGGREGO toolbox

To illustrate the intuitiveness and relevance of this autocompletion system, an instance of the AGGREGO server will be made available for the demonstration to query data coming from different sources like Twitter, Linkedin, BFM, Coface Services, Infolegale, etc. An example of query submitted in this context as well as the returned results are illustrated in Figure 7. A video is available online to illustrate the use of AGGREGO Search in such a mediation context: `http://www.youtube.com/watch?v=7LzFC_7tuf0&feature=youtu.be`.

5. REFERENCES

[3] `http://semsoft-corp.com`

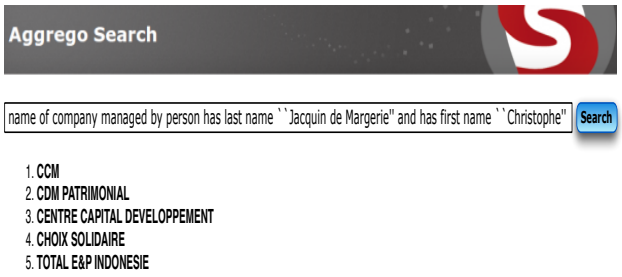

Figure 7: AGGREGO SEARCH on top of AGGREGO SERVER

[1] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: ranked keyword search over xml documents. In *Proc. of the 2003 ACM SIGMOD international conference on Management of data*, pages 16–27, 2003.

[2] J. Heinecke and F. Toumani. A natural language mediation system for e-commerce applications: an ontology-based approach. In *Proc. of 2nd International Semantic Web Conference*, 2003.

[3] Y. Lei, V. Uren, and E. Motta. Semsearch: A search engine for the semantic web. In S. Staab and V. Svátek, editors, *Managing Knowledge in a World of Networks*, volume 4248 of *Lecture Notes in Computer Science*, pages 238–245. Springer Berlin / Heidelberg, 2006.

[4] R. Patil and Z. Chen. Struct: Incorporating contextual information for english query search on relational databases. In *Proc. of the 3rd workshop KEYS*, 2012.

[5] R. Pottinger and A. Halevy. Minicon: A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3):182–198, Sept. 2001.

[6] J. Pound, I. F. Ilyas, and G. Weddell. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *Proceedings of the 2010 ACM SIGMOD international conference on Management of data*, pages 423–434, 2010.

[7] A. Simitsis, G. Koutrika, and Y. Ioannidis. Précis: from unstructured keywords as queries to structured databases as answers. *The VLDB Journal*, 17:117–149, 2008.

[8] G. Smits, O. Pivert, H. Jaudoin, and F. Paulus. An autocompletion mechanism for enriched keyword queries to rdf data sources. In *Proc. of the 10th international conference on Flexible Query Answering Systems*, pages 601–612. Springer, 2013.

[9] G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992.

[10] Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu. SPARK: Adapting keyword query to semantic search. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 694–707. 2007.