Demo: Swip, a Semantic Web Interface using Patterns

Camille Pradel, Ollivier Haemmerlé, and Nathalie Hernandez

IRIT, Université de Toulouse le Mirail, Département de Mathématiques-Informatique, 5 allées Antonio Machado, F-31058 Toulouse Cedex
{camille.pradel,ollivier.haemmerle,nathalie.hernandez}@univ-tlse2.fr

Abstract. Our purpose is to provide end-users with a means to query ontology based knowledge bases using natural language queries and thus hide the complexity of formulating a query expressed in a graph query language such as SPARQL. The main originality of our approach lies in the use of query patterns. Our contribution is materialized in a system named *SWIP*, standing for *Semantic Web Interface Using Patterns*. The demo will present use cases of this system.

1 Introduction

End-users need to access the huge amount of data available through the Internet. With the development of RDF triplestores and OWL ontologies, a need for interfacing SPARQL engines emerged, since it is impossible for an end-user to handle the complexity of the "schemata" of these pieces of knowledge. We think that the availability of voice recognition software which are becoming more and more popular, especially on smartphones, implies that we now have to work on the translation of NL queries into formal queries. The main hypothesis behind our work states that, in real applications, the submitted queries are variations of a few typical query families. We propose to guide the interpretation process by using predefined query patterns which represent these query families. The process benefits from the pre-established families of frequently expressed queries for which we know that real information needs exist.

In [1], we proposed a way of building queries expressed in terms of conceptual graphs from user queries composed of keywords. In [2] we extended the system in order to take into account relations expressed by the user between the keywords he/she used in his/her query and we introduced the pivot language allowing these relations to be expressed in a way inspired by keyword queries. In [3], we adapted our system to the Semantic Web languages instead of Conceptual Graphs. Such an adaptation was important for us in order to evaluate the interest of our approach on large and actual knowledge bases. The Swip system also participated in the first and third editions of the *Question Answering over Linked Data* (QALD) challenge. The results of this participation are detailed in [4].

This article gives a brief overview of our approach and presents its implementation which will be demonstrated.

2 Swip system overview

In the SWIP system, the query interpretation process is made of two main steps: the translation of the NL user query into a pivot query, and the formalization of this pivot query, respectively described in subsections 2.1 and 2.2.

2.1 From natural language to pivot query

The whole process of interpreting a natural language query is divided into two main steps, with an intermediate result, which is the user query translated into a new structure called the *pivot query*. This structure is half way between the NL query and the targeted formal query, and can be expressed through a language, called pivot language, which is formally defined in [3]. Briefly, this structure represents a query made of keywords and also expresses relations between those keywords. We use this pivot language in order to facilitate the implementation of multilingualism by means of a common intermediate format: a specific module of translation of NL to pivot has to be written for each different language, but the pivot query formalization step remains unchanged. This translation step is detailed in [4]. It consists of four stages.

The first stage aims at identifying in the NL query named entities corresponding to knowledge base resources; this allows these entities to be considered as a whole and prevents the parser from separating them in the next stage. Then, in the second stage, a dependency tree of the user NL query is processed by a dependency parser, taking into account the previously identified named entities. The third stage aims at identifying the query focus, i.e. the element of the query for which the user wants results (the element corresponding to the variable which will be attached to the SPARQL `SELECT` clause); SWIP is also able to detect *count queries* which ask for the number of resources fulfilling certain conditions and correspond in SPARQL to a `SELECT` query using a `COUNT` aggregate as a projection attribute, and *dichotomous (or boolean) queries* which allow only two answers (True or False / Yes or No), and are expressed in SPARQL with an `ASK` query. Finally, a set of predefined rules are applied to the dependency graph in order to obtain the elements of the pivot query and their relations.

2.2 From pivot to formal query

Formalizing pivot queries using query patterns was the first task we tackled and is extensively described in [2] and [3]. We briefly describe the structure of a query pattern and the process of this formalization.

A pattern is composed of an RDF graph which is the prototype of a relevant family of queries. Such a pattern is characterized by a subset of its elements – either class, property or literal type –, called the qualifying elements, which can be modified during the construction of the final query graph. It is also described by a sentence in natural language in which a distinct substring must be associated with each qualifying element. For now, the patterns are designed by experts who

know the application domain. The designer of a pattern builds its RDF graph manually, selects its qualifying elements and also gives the describing sentence.

The process of this step is as follows. Each element of the user query expressed in the pivot language is matched to an element of the knowledge base. Elements of the knowledge base can either be a class, a property, an instance or a literal type (which can be any type supported by SPARQL, i.e. any type defined in RDF Schema). Then we map query patterns to these elements. The different mappings are presented to the user by means of natural language sentences. The selected sentence allows the final SPARQL query to be built.

A recent evolution of the pattern structure makes the patterns more modular and the query generation more dynamic. We can now assign values of minimal and maximal cardinalities to subgraphs of the patterns, making these subgraphs optional or repeatable when generating the formal query. The descriptive sentence presented to the user also benefits from this novelty and no longer contains non relevant parts (parts of the pattern which were not addressed by the user query), thus making our system more ergonomic.

3 Implementation and evaluation

A prototype of our approach was implemented in order to evaluate its effectiveness. It is available at http://swip.univ-tlse2.fr/SwipWebClient. It was implemented in Java and uses the MaltParser[1] for the dependency analysis of English user queries. The system performs the second main process step (translating from pivot to formal query) by exploiting a SPARQL server based on the ARQ[2] query processor, here configured to exploit LARQ[3], allowing the use of Apache Lucene[4] features, such as indexation and Lucene score (used to obtain the similarity score between strings).

Experiments were carried out on the evaluation framework proposed in task 1 of the QALD-3 challenge[5]. A detailed analysis of the results is available in [4]. The evaluation method was defined by the challenge organizers. It consists in calculating, for each test query, the precision, the recall and the F-measure of the SPARQL translation returned by the system, compared with handmade queries of a gold standard document. We participated in both subtasks proposed by the challenge organizers, one targeting the DBpedia[6] knowledge base and the other targeting an RDF export of Musicbrainz[7] based on the music ontology[8]. The quality of the results varies with the target KB.

[1] http://www.maltparser.org/
[2] http://openjena.org/ARQ/
[3] LARQ = Lucene + ARQ, see http://jena.sourceforge.net/ARQ/lucene-arq.html
[4] http://lucene.apache.org/
[5] http://greententacle.techfak.uni-bielefeld.de/~{}cunger/qald/index.php?x=task1&q=3
[6] http://dbpedia.org
[7] http://musicbrainz.org/
[8] http://musicontology.com/

On the Musicbrainz test dataset, we processed 33 of the 50 test queries. 24 were correctly interpreted, 2 were partially answered and the others failed. The average precision, recall and F-measure, calculated by the challenge organizers, are all equal to 0.51. We consider these results as quite good and very encouraging. However, results on the DBpedia dataset are more disappointing. We processed 21 of the 100 test queries, of which 14 were successful, 2 were partially answered and 5 were not correct. The average precision, recall and F-measure are all equal to 0.16.

The proposed demo will showcase a didactic user interface which displays results of intermediate steps. The target dataset will be Musicbrainz and the target language English. It will also be possible to visualize and edit query patterns through a control panel in order to influence the interpretation process.

4 Conclusion and future work

In this paper, we presented the approach we are designing to allow end users to query graph-based knowledge bases. This approach is implemented in the SWIP system and is mainly characterized by the use of query patterns in the interpretation of the user NL query. The setting up of the two main parts of the system process is nearly done and the first results are very encouraging.

We plan to extend our work in several directions: experimenting the ease of adaptation to different user languages, by participating to the multilingual task of the QALD challenge, and collaborating with IRSTEA (the French institute of ecology and agriculture) in order to build a real application framework concerning French queries on organic farming; experimenting methods to automate or assist the conception of query patterns; extending the query that can be processed by our system, for example by taking into account extensions of SPARQL 1.1, such as aggregates; exploring new leads allowing the approach to evolve and stick more to the data itself than to the ontology, in order to obtain better results on datasets from the Web of linked data, such as DBpedia.

References

1. Comparot, C., Haemmerlé, O., Hernandez, N.: An easy way of expressing conceptual graph queries from keywords and query patterns. In: ICCS. pp. 84–96 (2010)
2. Pradel, C., Haemmerlé, O., Hernandez, N.: Expressing conceptual graph queries from patterns: how to take into account the relations. In: Proceedings of the 19th International Conference on Conceptual Structures, ICCS'11, Lecture Notes in Artificial Intelligence # 6828. pp. 234–247. Springer, Derby, GB (July 2011)
3. Pradel, C., Haemmerlé, O., Hernandez, N.: A semantic web interface using patterns: The swip system (regular paper). In: Croitoru, M., Rudolph, S., Wilson, N., Howse, J., Corby, O. (eds.) IJCAI-GKR Workshop, Barcelona, Spain, 16/07/2011-16/07/2011. pp. 172–187. No. 7205 in LNAI, Springer, http://www.springerlink.com (mai 2012)
4. Pradel, C., Peyet, G., Haemmerlé, O., Hernandez, N.: Swip at qald-3: results, criticisms and lesson learned (working notes). In: CLEF 2013, Valencia, Spain, 23/09/2013-26/09/2013 (2013)