Natural Language Query Interpretation into SPARQL 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. In this article we justify the postulate supporting our work which claims that queries issued by real life end-users are variations of a few typical query families. We also explain how our approach is designed to be adaptable to different user languages. Evaluations on the QALD-3 data set have shown the relevancy of the approach.

1 Introduction

With the development of RDF triplestores and OWL ontologies, it became necessary to interface SPARQL engines, since it is impossible for an end-user to handle the complexity of the "schemata" of these pieces of knowledge: in order to express a valid query on the knowledge of Linked Data, the user needs to know the SPARQL query language as well as the ontologies used to express the triples he/she wants to query on. Several works have been done on the generation of graph queries from keyword queries. We think that the availability of voice recognition softwares which understand natural speech and become more and more popular, especially on smartphones, implies that we have now to work on the translation of NL queries into formal queries.

Our work takes place in that field of research: how could we interpret a natural language (NL) query and translate it in SPARQL. The main postulate leading our work states that, in real applications, the submitted queries are variations of a few typical query families. Our approach differs from existing ones in the way that we propose to guide the interpretation process by using predefined query patterns which represent these query families. The use of patterns avoids exploring the ontology to link the semantic entities identified from the keywords since potential query shapes are already expressed in the patterns. The process thus benefits from the pre-established families of frequently expressed queries for which we know that real information needs exist.

In [3], we proposed a way of building queries expressed in terms of conceptual graphs from user queries composed of keywords. In [12] 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 to express these relations in a way inspired by keyword queries. In [13], 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 (KBs).

Our approach takes a natural language query as input and proposes ranked SPARQL queries and their associated descriptive sentences as output. This paper recall this aproach while focusing on novelties: pattern justification from literature, translation from NL queries to pivot queries, and modular patterns containing optional and repeatable subpatterns. Section 2 summarizes works and systems sharing our objectives. Section 3 presents an overview of our approach and introduces the pivot language which is an intermediate format between the NL query and the SPARQL query, and which allows us to adapt easily our approach to different natural languages. Section 4 presents the notion of query patterns and their use. In Section 5, we present the implementation of our work in the *Semantic Web Interface using Patterns* (SWIP) system and the results obtain by this system in the third edition of the *Question Answering over Linked Data* (QALD) challenge.

2 Related work

In this section, we present approaches from the literature aiming at helping users to query graph based KBs. For this, we stick to the classification proposed in [8], and present them following a formality continuum, from most formal to most permissive, beginning with the query languages themselves.

On an extreme side of this continuum are the formal graph languages, such as SPARQL[1]. They are the targets of the systems we are presenting in this section. Such languages present obvious usability constraints, which make them unsuited to end-users. Expressing formal queries implies knowing and respecting the language syntax used, understanding a graph model and, most constraining, knowing the data schema of the queried KB. The work presented in [4] aims at extending the SPARQL language and its querying mechanism in order to take into account keywords and wildcards when the user does not know exactly the schema he/she wants to query on. Here again, such an approach requires that the user knows the SPARQL language.

Similar are approaches assisting the user during the formulation of queries in such languages. Very light interfaces such as *Flint*[2] and *SparQLed*[3] implement simple features such as syntactical coloration and autocompletion. Other approaches rely on graphical interfaces such as [16,2] for SPARQL queries. Even if these graphical interfaces are useful and make the query formulation work less

[1] http://www.w3.org/TR/rdf-sparql-query/
[2] http://openuplabs.tso.co.uk/demos/sparqleditor
[3] http://sindicetech.com/sindice-suite/sparqled/

tedious, we believe they are not well suited to end-users since they do not overcome the previously mentioned usability limits of formal graph query languages.

Sewelis [5] introduces *Query-based Faceted Search*, a new paradigm combining faceted search and querying, the most popular paradigm in semantic data search, while other approaches such as *squall2sparql* [6] define a controlled natural language whose translation into SPARQL is straightforward.

Other works aim at generating formal queries directly from user queries expressed in terms of keywords or NL. Our work is situated in this family of approaches. The user expresses his/her information need in an intuitive way, without having to know the query language or the knowledge representation formalism. Some works have already been proposed to express formal queries in different languages such as SeREQL [10] or SPARQL [22,19,1]. In these systems, the generation of the query requires the following steps: (i) matching the keywords to semantic entities defined in the KB, (ii) building query graphs linking the entities previously detected by exploring the KB, (iii) ranking the built queries, (iv) making the user select the right one. The existing approaches focus on several main issues: optimizing the first step by using external resources (such as WordNet or Wikipedia)[10,21], optimizing the knowledge exploration mechanism for building the query graphs [22,19], enhancing the query ranking score [21], and improving the identification of relations using textual patterns [1].

Autosparql [9] extends the previous category: after a basic interpretation of the user NL query, the system interacts with the user, asking for positive and negative examples (i.e. elements which are or are not in the list of expected answers), in order to refine the initial interpretation by performing a learning algorithm. In [20] the interpretation process of this same system is improved by determining a SPARQL template from the syntactic structure of the natural language question.

3 The pivot query

In the SWIP system, the query interpretation process consists of two main steps which are illustrated in Figure 1. The first step, the *natural language query interpretation* roughly consists in the identification of named entities, a dependency analysis and the translation of the obtained dependency graph into a new query, called *pivot query* and presented in this section. In the second step, *pivot query formalization*, predefined query patterns are mapped to the pivot query; we thus obtain a list of potential interpretations of the user query, which are then ranked according to their estimated relevance and proposed to the user in the form of reformulated NL queries. The displayed relevancy mark and the reformulation of the suggested query interpretation in NL are for us a way to overcome the habitability problem introduced in [8]. The habitability problem states that end users can be quite at a loss when facing too much freedom in query expression: he/she can express queries out of the system capabilities (with respect to the information need or to the query formulation) and, in worst cases, the system can misunderstand the query and the user not realize it. The relevancy mark

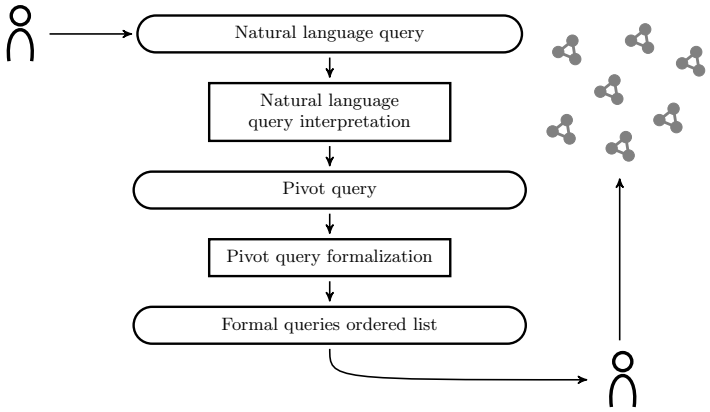


Fig. 1. Overview of the SWIP interpretation process.

and the NL query reformulation give to the user feedback in an understandable way and consequently increase the confidence in the system.

Pivot query is a structure half-way between the NL query and the targeted formal query, and aims at storing results from the first interpretation step. It explicitly represents extracted relations between keywords of the NL query sentence. 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. For the moment, we have adapted our system for English and French.

3.1 Definition and syntax of the pivot language

The detailed grammar of the pivot language is presented in [13]. A pivot query is composed of keywords connected with relationships which are more or less explicit. The optional "`?`" symbol before a keyword means that this keyword is the focus of the query: we want to obtain specific results corresponding to that keyword. A pivot query is composed of a conjunction of subqueries: unary subqueries, like `?"actor"` which asks for the list of actors in the KB; binary subqueries which qualify a keyword with another keyword: the query `?"actor": "married"` asks for the list of married actors; and ternary subqueries which qualify, by means of a keyword, the relationship between two other keywords: the query `?"actor": "married to"= "Penelope Cruz"` asks for the actor(s) that is/are/was/were married to Penelope Cruz.

3.2 From natural language to pivot query

The process of translating a natural language query into a pivot query is detailed in [14]. It consists of four stages. The first stage aims at identifying in the natural

language query named entities corresponding to KB 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 (alternatively 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 extract all the query elements and their relations.

4 Query patterns

The following sentences are NL queries a user could ask on the cinema domain: (i) "Which actors play in the movie Biutiful?" ; (ii) "Which thrillers were released in 2008?" ; (iii) "Which movies were made by a French director?" ; (iv) "Which movies were directed by the Coen brothers?". These queries look very familiar. Everybody once wondered who was the actor/actress playing in such or such a movie. This is a basic observation that naturally leads to the idea of patterns.

4.1 Justification

The main postulate directing our work states that, in real applications, the submitted queries are variations of a few typical query families. The authors from [18] analyse query logs from English-speaking users on a real Web search engine and discuss their results based on previous similar studies [7,17]. Although first observations tend to contradict our hypothesis, their conclusions reinforce our need to retain it. Authors firstly point out that the vocabulary (and so potentially, for what matters to us, the semantics) of queries is highly varied. On the query set they analysed, including 926,877 queries containing at least one keyword, of the 140,279 unique terms, some 57,1% where used only once, 14,5% twice, and 6,7% three times. This represents quite a higher rate of very rarely used terms, compared to "classical" text resources. These figures must be moderated, since this phenomenon is partially caused by a high number of spelling errors, terms in languages other than English, and Web specific terms, such as URLs.

On the other hand, a few unique terms are used very frequently. In the analysed queries, the 67 most frequent meaningful terms (i.e. terms such as "and", "of", "the" were not taken into account) represent only 0.04% of unique terms that account for 11.5% of all terms used in all queries.

Moreover, two further analyses where carried out to complete these results. The first one addresses co-occurrence of terms; by identifying the most frequently occurring pairs of terms, it is possible to highlight some popular and recurrent topics. The last presented analysis is qualitative and is the most relevant to the semantics of queries. It consists in the manual classification of a subset of the query set. 11 major categories where highlighted from this analysis, each one corresponding to a set of related topics. Such observations led us to propose a mechanism allowing a user query to be expressed in NL and then translated into a graph query built by adapting pre-defined query patterns chosen according to the keywords. The use of patterns was the main difference between our approach and other approaches developed simultaneously by other teams.

4.2 Definition

The patterns we propose are modular since it is possible to define "sub-pattern" which can be optional or repeatable in the generated SPARQL query. A sub-pattern is characterized by a minimal and a maximal cardinality. A minimal cardinality of 0 (resp. a maximal cardinality greater than 1) means that the sub-pattern is optional (resp. repeatable). Nested sub-patterns are allowed.

Let G be a graph and v a vertex belonging to this graph; we note $G \setminus v$ the subgraph of G obtained by removing the vertex v and all its incident edges.

Definition 1. *A pattern p is a 4-tuple $(G, \mathcal{Q}, \mathcal{SP}, \mathcal{S})$ such that:*

- *G is a connected RDF graph which describes the general structure of the pattern and represents a family of queries. Such a graph only contains triples according to the structure presented in Figure 2 ;*
- *$\mathcal{Q}$ is a subset of elements of G, called qualifying elements; these elements are considered to be characteristics of the pattern and will be taken into account during the mapping of the user query and the considered pattern. A qualifying element can either be a vertex (representing a class or a datatype) or an edge (representing an object property or a datatype property) of G ;*
- *$\mathcal{SP}$ is the set of sub-patterns sp of p such that,*
  $\forall sp = (SG, v, card_{min}, card_{max}) \in \mathcal{SP}$, we have:
  - *SG is a subgraph of G and v a vertex of SG (and then of G), such that $G \setminus v$ is non-connected (v is a cut vertex of G) and admits $SG \setminus v$ as a connected component (i.e. all the vertices belonging to that connected component belong to the graph of the sub-pattern) ;*
  - *at least one element (vertex or edge) of SG is qualifying;*
  - *$card_{min}, card_{max} \in \mathbb{N}$ such that $0 \leq card_{min} \leq card_{max}$ are respectively the minimal and maximal cardinalities of sp.*
- *$\mathcal{S} = (s, (sw_1, sw_2, ...sw_n), (w_1, w_2, ...w_m))$ is a template of a descriptive sentence in which n substrings sw_i correspond to the n sub-patterns, m distinct substrings w_j correspond to the m qualifying elements.*

Figure 3 shows an example of a generic query pattern. It is composed of four subpatterns named *genre*, *date*, *person* and *nationality*. All of them are optional.

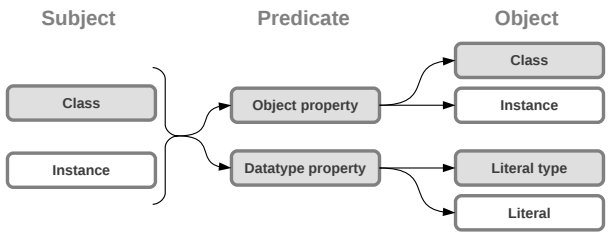
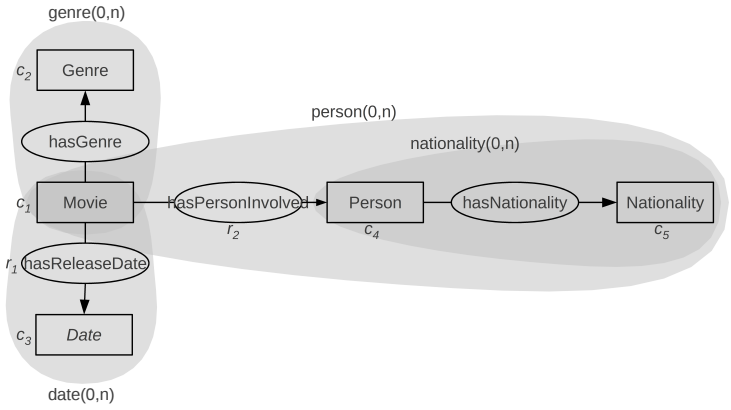


Fig. 2. Constraints on the triples constituting the graph patterns.

They are also repeatable, except for subpattern *date*: it is considered that a movie cannot have more than one release date. In the descriptive sentence template, parts corresponding to a subpattern are written between square brackets and with the subpattern identifier as an index, qualifying vertices are underlined, with an index referring to the graph element.



A movie$_{c_1}$ [of genre c_2]$_{genre}$ [that was released on c_3]$_{date}$ [has for person involved$_{r_2}$ a person$_{c_4}$ [which is c_5]$_{nationality}$]$_{person}$

Fig. 3. Example query pattern.

4.3 Pivot query interpretation through patterns

We now present how a query pattern is instantiated, i.e. how it is transformed into a SPARQL query. We present successively the instantiation of a pattern element, of a sub-pattern, then of a whole pattern.

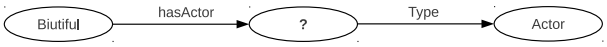
Instantiation of a pattern element We explain how a qualifying element of a pattern is instantiated, i.e. how the query graph is modified when one of its qualifying elements is matched with an element of the user query.

Let the pattern $p = (G, \mathcal{Q}, \mathcal{SP}, \mathcal{S})$. For all $q \in \mathcal{Q}$ qualifying elements of p and for all resources α extracted from the user query (which can be either a class, an instance, or a property), we denote by $I\big(p, (q \leftarrow \alpha)\big) = (G', \mathcal{Q}', \mathcal{SP}', \mathcal{S}')$ the pattern obtained after the instantiation of q by the resource α in the pattern p. This instantiation is only possible if q and α are compatible, as explained in [13]. Without describe formally the different cases of compatibility, we recall the cases and describing the instantiation of the corresponding qualifying element by means of an example built from the pattern of Figure 3.

1. q is a class and α an instance of q. Then the instantiation of the qualifying concept consists in replacing the URI of the class by the URI of the instance.
2. q is a datatype and α a value corresponding to the type q. Then the instantiation of the qualifying concept consists in replacing the URI of the class by the value α.
3. q is a property and α the same property or one of its sub-properties. Then the instantiation of the qualifying edge consists in replacing the URI of the edge by the URI of the property α.
4. q is a class and α the same class or one of its sub-classes. In this case, G' is graph G in which q has been replaced by a blank node characterized by a supplementary triple specifying that this blank node is of type α.
5. Finally, a pattern element can also be instantiated if it has been associated with no element of the KB ($\alpha = \varnothing$). The result of this instantiation is such that:
   - if q is a property, then $G' = G$;
   - if q is a class or a datatype, then G' is graph G in which q has been replaced by a blank node characterized by a supplementary triple specifying that this blank node is of type q.

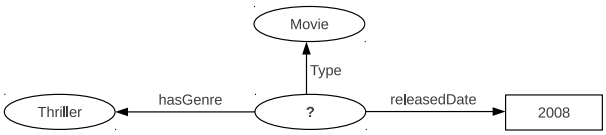
Instantiation of a sub-pattern Since the sub-patterns can be nested, we define their instantiation recursively. Let $p = (G, \mathcal{Q}, \mathcal{SP}, s)$ be a pattern and $sp = (SG, s, card_{min}, card_{max}) \in \mathcal{SP}$ a sub-pattern of p. Pattern p' is induced from the instantiation of sp in p if p' has its graph G' which is graph G in which SG' can appear n times ($card_{min} \leq n \leq card_{max}$) with, for each occurrence of SG', a different combination of instantiations of qualifying elements. SG' is obtained by applying to SG the instantiation of the set of its sub-patterns then by the instantiation of all the remaining qualifying elements.

Instantiation of a pattern A pattern can be considered as a sub-pattern which is not nested in another one, and of which the minimal and maximal cardinalities are equal to 1. The instantiation mechanism remains the same.



Descriptive sentence : Biutiful has for actor **some actor**.

Fig. 4. Instantiation of pattern 3 corresponding to the query (i).



Descriptive sentence : **A movie** of genre thriller was released in 2008.

Fig. 5. Instantiation of pattern 3 corresponding to the query (ii).

We present successively in Figures 4, 5, 6 and 7 the query graphs generated by the SWIP system for the queries given in the introduction of this section.

Note that the interpretation of the query (iv) is possible thanks to the repeatability of a sub-pattern which allows to match simultaneously "Joel Coen" and "Ethan Coen".

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.

5 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 [11] 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[4] query processor, here configured to exploit LARQ[5], allowing the use of Apache Lucene[6] features, such as indexation and Lucene score (used to obtain the similarity score between strings).

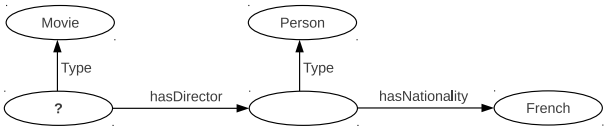
Experiments were carried out on the evaluation framework proposed in task 1 of the QALD-3 challenge[7]. A detailed analysis of the results is available in [14]. The evaluation method was defined by the challenge organizers. It consists in calculating, for each test query, the precision, the recall and 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

[4] `http://openjena.org/ARQ/`

[5] LARQ = Lucene + ARQ, see `http://jena.sourceforge.net/ARQ/lucene-arq.html`

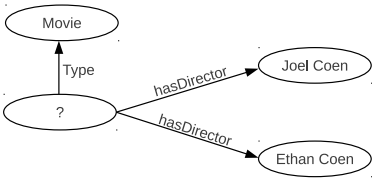
[6] `http://lucene.apache.org/`

[7] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=task1&q=3`



Descriptive sentence : **A movie** that was directed by some person which is French.

Fig. 6. Instantiation of pattern 3 corresponding to the query (iii).



Descriptive sentence : **A movie** that was directed by Joen Coen and was directed by Ethan Coen.

Fig. 7. Instantiation of pattern 3 corresponding to the query (iv).

challenge organizers, one targeting the DBpedia[8] KB and the other targeting an RDF export of Musicbrainz[9] based on the music ontology [15]. We took into consideration only questions in English and defined query patterns manually for each dataset. The quality of the results varies with the target KB.

The Musicbrainz test dataset was composed of 50 NL questions. We processed 33 of them. 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. It is difficult to evaluate these performances, as no other challenge participant tackled the Musicbrainz dataset. However, these results are quite good when compared to those obtained by participants on the DBpedia dataset. Indeed, among the other participants, the squall2sparql system showed the best performance by far, but this system does not accept full NL sentences as an input [6], and then, the second best F-measure is 0.36. Of course the relevance of such a direct comparison is very questionable, as both graph bases present significant differences and our system obtained poor results on DBpedia. Indeed, the results obtained by SWIP on the DBpedia dataset, composed of 100 test questions, are 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 main problem for us was the "tidiness" of the processed data. Indeed, most of the DBpedia KB is automatically generated from information written by thousands of contributors and not exported from a consistent database like Musicbrainz. Consequently, we had to deal with the heterogeneity and inconsistency

[8] `http://dbpedia.org`
[9] `http://musicbrainz.org/`

of the target KB that SWIP was not able to overcome. We can deduce that our pattern based approach is very well suited for "clean" KB, with consistent data, domain specific and respecting the schema it is based on. However, in its current implementation, it is not well suited to KBs containing many inconsistencies, which are the most popular KB on the Web of linked data.

6 Conclusion and future work

In this paper, we presented the approach we are designing to allow end-users to query graph-based KBs. This approach is mainly characterized by the use of query patterns leading the interpretation of the user NL query and its translation into a formal graph query. Although some arrangements still need to be carried out, the setting up of the two main parts of the system's process is almost done and the first results are very encouraging. We plan to extend in the following directions:

- experimenting the ease of adaptation to different user languages; we will participate to the *Multilingual question answering* task of the QALD challenge and we are developing a partnership with the IRSTEA (a French institute on ecology and agriculture) in order to build a real application framework concerning French queries on sustainable and organic farming;
- experimenting methods to automate or assist the conception of query patterns; we first want to automatically determine the pattern structures by analysing graph query examples, and then compare the developed method(s) to an approach based on NL query learning;
- exploring new leads allowing the approach to evolve and stick more to the data itself than to the ontology, in order to get better results on datasets from the Web of linked data, such as DBpedia.

References

1. Cabrio, E., Cojan, J., Aprosio, A., Magnini, B., Lavelli, A., Gandon, F.: Qakis: an open domain qa system based on relational patterns. In: International Semantic Web Conference (Posters & Demos) (2012)
2. Clemmer, A., Davies, S.: Smeagol: a "specific-to-general" semantic web query interface paradigm for novices. In: Proceedings of the 22nd international conference on Database and expert systems applications - Volume Part I. pp. 288–302. DEXA'11, Springer-Verlag, Berlin, Heidelberg (2011), `http://dl.acm.org/citation.cfm?id=2035368.2035396`
3. 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)
4. Elbassuoni, S., Ramanath, M., Schenkel, R., Weikum, G.: Searching rdf graphs with sparql and keywords. IEEE Data Eng. Bull. 33(1), 16–24 (2010)
5. Ferré, S., Hermann, A.: Reconciling faceted search and query languages for the semantic web. International Journal of Metadata, Semantics and Ontologies 7(1), 37–54 (2012)

6. Ferré, S.: Squall: A controlled natural language for querying and updating rdf graphs. In: Controlled Natural Language, pp. 11–25. Springer (2012)
7. Jansen, B., Spink, A., Saracevic, T.: Real life, real users, and real needs: a study and analysis of user queries on the web. Information processing & management 36(2), 207–227 (2000)
8. Kaufmann, E., Bernstein, A.: Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. Web Semantics: Science, Services and Agents on the World Wide Web 8(4), 377–393 (2010)
9. Lehmann, J., Bühmann, L.: Autosparql: let users query your knowledge base. In: Proceedings of the 8th extended semantic web conference on The semantic web: research and applications-Volume Part I. pp. 63–79. Springer-Verlag (2011)
10. Lei, Y., Uren, V.S., Motta, E.: Semsearch: A search engine for the semantic web. In: EKAW. pp. 238–245 (2006)
11. Nivre, J., Hall, J., Nilsson, J.: Maltparser: A data-driven parser-generator for dependency parsing. In: Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006), May 24-26, 2006, Genoa, Italy. pp. 2216–2219. European Language Resource Association, Paris (2006)
12. 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)
13. 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)
14. 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)
15. Raimond, Y., Abdallah, S., Sandler, M., Giasson, F.: The music ontology. (2007)
16. Russell, A., Smart, P.R.: Nitelight: A graphical editor for sparql queries. In: International Semantic Web Conference (Posters & Demos) (2008)
17. Silverstein, C., Marais, H., Henzinger, M., Moricz, M.: Analysis of a very large web search engine query log. In: ACm SIGIR Forum. vol. 33, pp. 6–12. ACM (1999)
18. Spink, A., Wolfram, D., Jansen, M., Saracevic, T.: Searching the web: The public and their queries. Journal of the American society for information science and technology 52(3), 226–234 (2001)
19. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: ICDE. pp. 405–416 (2009)
20. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A., Gerber, D., Cimiano, P.: Template-based question answering over rdf data. In: Proceedings of the 21st international conference on World Wide Web. pp. 639–648. ACM (2012)
21. Wang, H., Zhang, K., Liu, Q., Tran, T., Yu, Y.: Q2semantic: a lightweight keyword interface to semantic search. In: Proceedings of the 5th European semantic web conference on The semantic web: research and applications. pp. 584–598. Springer-Verlag (2008)
22. Zhou, Q., Wang, C., Xiong, M., Wang, H., Yu, Y.: Spark: Adapting keyword query to semantic search. In: ISWC/ASWC. pp. 694–707 (2007)