sciendo International Conference KNOWLEDGE-BASED ORGANIZATION
Vol. XXVI No 3 2020

IQ SEMANTIC: SEMANTIC WEB INTELLIGENT QUERYING FRAMEWORK

Ilie Cristian DOROBĂȚ, Vlad POSEA

"Politehnica" University of Bucharest, Bucharest, Romania
ilie.dorobat@stud.acs.upb.ro, vlad.posea@cs.pub.ro

Abstract: *The continuous expansion of the semantic web and of the linked open data cloud meant more semantic data are available for querying from endpoints all over the web. We propose extending a standard SPARQL interface with UI and Natural Language Processing features to allow easier and more intelligent querying. The paper describes some usage scenarios for easy querying and launches a discussion on the advantages of such an implementation.*

Keywords: Linked Data, SPARQL, Natural Language Processing

1. Motivation

In a period when the quality level of the data alongside the rendering of operations more flexible and efficient play an important role in the society's development, the data cleaning process and their publication as Linked Data represents only a first stage which public institutions must follow in order to improve the method of digital representation of the cultural heritage [1]. Another direction towards which institutions must focus their attention is represented by the ease of the access the users have to data. For this, "intelligent" search engines can be implemented, which would allow users to use the natural language to query the datastore. Moreover, in order to allow specialists and researchers an advanced search in the entire datastore, it can be considered the development of an Integrated Development Environment (IDE) which, on the one hand would assist users in the creation of queries through autocompletion feature, and, on the other hand, would validate the syntax of these queries.

Even though some cultural institutions such as British National Library [2], Library of Congress [3] or German National Library [4], have available the human and financial resources necessary for the research and development of new solutions of digital representation of data, most of the public institutions in this area do not have the necessary resources for such efforts [5][6]. Starting from the users' need to easily access the published datasets, and from the cultural institutions' need to offer users a browsing environment as flexible as possible, the Semantic Web Intelligent Querying Framework (semIQ) [7] is developed to simplify and render more efficient the way of querying the datastores by providing an endpoint capable of receiving a query in natural language and translating it into a SPARQL query, and also by designing of an additional support to the manual creation of the SPARQL queries.

2. The Framework Workflow

The current section is dedicated to the presentation of the two features provided by semIQ, the interface for triple store querying in natural language, respectively, the IDE through which professionals can

DOI: 10.2478/kbo-2020-0122

build SPARQL queries in an easier way.

2.1. NLP Querying Interface

Natural Language Processing (NLP) Querying Interface (NQI) is a component of the semIQ Framework which aims is to allow users to query triple stores using the natural language, that is, by formulating the requests as if behind the interface there is another person and not a machine. The workflow of translation of the request sent by the user in a SPARQL query includes several stages, starting with providing the query written in natural language and ending with generating the SPARQL query.

The language based on which the user made the request is identified through the means of the *langdetect* library [8], so that the user can be informed if the language used is not supported. After that, using the *spaCy* library [9], the query provided by the user is transformed into a *Doc* object, which allows accessing linguistic annotations. Because the named entities (natural and legal persons, places, etc.) are chunks that don't make sense if they are split into words, we will proceed to identify them before tokenization begins.

After the identification of named entities follows the tokenization processes, through which the *Doc* object is divided into *Token* objects. Although tokens can usually be seen as words, there are two advantages to representing them through the *Token* data type, namely: a) the existence of an additional layer for the storage of cross-grained part-of-speech tags; b) the tokens can reflect not only words, but also compound chunks as named entities, which make no sense to split them [10].

After this stage, it is considered to extract all the properties and namespaces defined in the analysed graph, and then we proceed to the actual construction of the SPARQL query. Basically, for the extracted tokens, the best matches with the properties found in the graph are searched.

2.2. SPARQL Querying IDE

SPARQL Querying IDE (SQI) is a concept which will be developed as part of the semIQ Framework. The aim of SQI is to offer to users, particularly to analysts and programmers, an IDE useful for the building as efficient as possible of SPARQL queries. Alongside the classic syntax highlighting feature through which certain elements are shown in different colours (keywords, the name of variables and methods, the type of data, etc.) [11], SQI will also integrate an intelligent code completion feature, which provides a set of suggestions related to the namespaces and properties which the user may not be aware of, but which could be helpful in obtaining the desired result.

Table 1 describes the way the autocompletion feature works by presenting the actions which will trigger the prediction (the "Action" column), the locations where is triggered (the "Where" column) and the results which are displayed to the user (the "What" column). Basically, the prediction could be realized both according to namespaces and to the defined variables, but also according to the used ontologies.

Table 1 SQI autocomplete feature clauses

Action	Where	What
typing the subject variable name	in the body of the WHERE clause	namespaces
typing the ";" sign		namespaces
inserting a comparison sign	in the body of the FILTER clause	namespaces
inserting the ":" sign	after the ":" sign that was added to a namespace name	the closest ontology elements of the context
keypress	after every occurrence of the "?" sign which appears after defining the variables name from the SELECT clause	defined variables

3. Preliminary Results

NLP is the branch of Artificial Intelligence which offers a number of techniques and procedures for manipulating the natural language by computers [12]. Although these techniques provide a reliable result in the analysis of large texts than in analysis of small texts, they are a good tool to use even for small texts such as the user queries.

In order to evaluate the NQI component, the current section will present a practical example of building a SPARQL query starting from the following request provided by the user using the natural language: *"Display the artefacts hosted in Romania"*. The first step in this direction is to extract the sentences that contain a noun, to identify the main sentence, and to identify the verbs that are related to them based on the syntactic dependencies.

As can be seen in Figure 1, in which the syntactic relations between tokens are presented, two elements are identified that fulfil the syntactic functions of the noun: "the artefacts" and "Romania". Based on these two items, the verbs that are related to them are identified, and the main sentence is marked through the *is_main_statement* flag as following:

```
statement[0]: {
    'is_main_statement': True,
    'verb': Display,
    'predicate': artifacts,
    'complement': []
}

statement[1]: {
    'is_main_statement': False,
    'verb': hosted,
    'predicate': Romania,
    'complement': [
        'city',
        'county',
        'country',
        'locality',
        'location',
        'state',
        'town',
        'village'
    ]
}
```

After the syntactic analysis, we need to query the datastore to get the properties used in the graph, then we can proceed to build the actual SPARQL query. To build the query we need to add one by one, the following three elements:

- list of prefixes;
- interrogation statements;
- filtering statements.

The result is the SPARQL query presented in Figure 2.

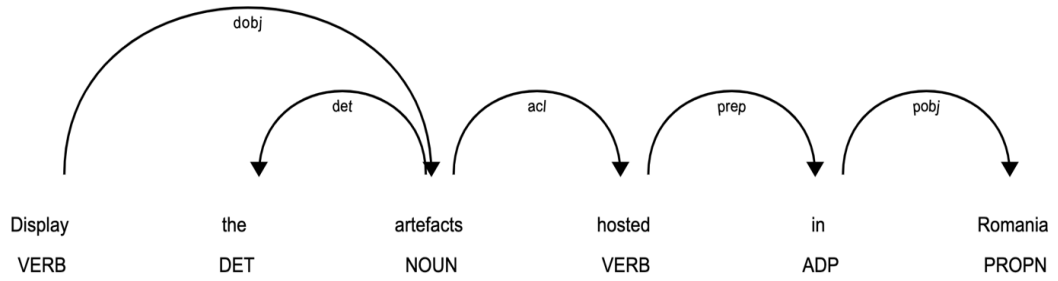

Figure 1: Syntactic dependencies

4. Conclusions and Further Development

The positioning of public institutions towards the digitalization of cultural heritage under triggered new challenges which must be dealt with [13]. From the identification of the best methods of data representation and up to the development of new solutions which would facilitate users' access to the presented data, all these improvements require time and resources in order to be accomplished.

Considering that not all public institutions have available the necessary resources for such efforts, developing a software solution

that might be integrated in the already developed applications in order to facilitate the extraction of data from semantic data stores can only relieve institutions from bearing the costs of research and development of such solutions.

semIQ is a framework developed to answer the need of improving the way in which people can query semantic data stores. One of the features that the framework provides is the translation of the query from the natural language into SPARQL query, so that it allows users to obtain the desired information easier. Also, it has been discussed the extending of semIQ with an IDE which specialist can use to generate more complex queries, with the benefit of autocompletion and syntax highlighting features.

Regarding the further development direction, we aim to improve the NQI feature by identifying the synonyms of both the tokens related to the formal vocabulary, and the informal one, used in common speech. We also intend to build a new graph based on the original one, but which uses lemmatized values, so that we can generate queries as accurate and as wide as possible.

```
PREFIX opendata: <http://opendata.cs.pub.ro/property/ro/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE {
        OPTIONAL { ?s opendata:age ?opendata_age } .
        OPTIONAL { ?s opendata:colour ?opendata_colour } .
        OPTIONAL { ?s dc:description ?dc_description } .
        OPTIONAL { ?s dc:subject ?dc_subject } .
        OPTIONAL { ?s dc:title ?dc_title } .
        OPTIONAL { ?s dcterms:alternative ?dcterms_alternative } .
        OPTIONAL { ?s edm:aggregatedCHO ?edm_aggregatedCHO } .
        OPTIONAL { ?s edm:country ?edm_country } .
        OPTIONAL { ?s edm:wasPresentAt ?edm_wasPresentAt } .
        OPTIONAL { ?s rdf:subject ?rdf_subject } .
        OPTIONAL { ?s skos:note ?skos_note } .
        OPTIONAL { ?s skos:prefLabel ?skos_prefLabel } .
        OPTIONAL { ?s foaf:name ?foaf_name } .
        [................................................]
        FILTER (
                contains(edm:country, "Romania")
        )
}
```

Figure 2: SPARQL query

References

[1] Dorobăț I. C., Rinciog O., Muraru G. C. and Posea V.: *Improving the Quality of Linked Data Using String Suggestions*. In: eLearning & Software for Education, eLSE 2020, vol. 2, pp. 375-389, 2020, doi:10.12753/2066-026X-20-133;

[2] https://bnb.data.bl.uk/;

[3] https://www.loc.gov/;

[4] https://www.dnb.de/EN/Home/home_node.html;

[5] Dietrich D., Peke J.: *Open Data in Cultural Heritage Institutions*. European Public Sector Information Platform. Topic Report no. 2 012/04. 2012. https://www.europeandataportal.eu/sites/default/files/report/2012_open_data_in_cultural_heritage_institutions.pdf, last accessed 2020/05/15;
[6] de Boer V. et al.: *Supporting Linked Data Production for Cultural heritage Institutes: The Amsterdam Museum Case Study*. In: Simperl E., Cimiano P., Polleres A., Corcho O., Presutti V. (eds) The Semantic Web: Research and Applications - ESWC 2012, LNCS, vol. 7295, pp. 733-747, Springer, Berlin, Heidelberg (2012), doi:10.1007/978-3-642-30284- 8_56;
[7] https://github.com/iliedorobat/semIQ;
[8] https://pypi.org/project/langdetect/;
[9] https://spacy.io/;
[10] Webster, J. J., Kit, C.: *Tokenization as the initial phase in NLP*. In: Proceedings of the 14th conference on computational linguistics, pp. 1106–1110, COLING (1992), doi:10.3115/992424.992434;
[11] Rietveld L., Hoekstra R.: *The YASGUI family of SPARQL clients*. In: Semantic Web, vol. 8, no. 3, pp. 373-383, IOS Press, doi:10.3233/SW-150197;
[12] Bird S., Klein E. and Loper E., *Natural Language Processing with Python*, O'Reilly Media, 2009;
[13] Ross S.: *Digital Preservation, Archival Science and Methodological Foundations for Digital Libraries*. In: New Review of Information Networking, vol. 17, no. 1, pp. 43-68, Routledge (2015), doi:10.1080/13614576.2012.679446.