User-friendly Query Interfaces for the HOPE Project

Discussion Paper

Diego Marcia, Manuela Sanguinetti and Maurizio Atzori

Dipartimento di Matematica e Informatica (DMI), Università di Cagliari, Via Ospedale 72, 09124, Cagliari (CA), Italy

Abstract

The paper introduces the current state of an ongoing research project devoted to the study and development of a user-friendly query interface to access open data. More specifically, the interface has been conceived so to enable simplified queries over heterogeneous data from different sources, such as public governmental sources or private structured repositories. This has been done by expanding the functionalities of an already existing interface for querying knowledge bases. The aim of this paper is showing the main interface features, highlighting the research questions they respond to and the open issues we intend to explore as next steps of the project.

Keywords

Question Answering, linked data, by-example, learning to rank

1. Introduction

In the recent years, we have seen a steady growth of structured data being made available to the public, e.g. with governments contributing to this trend by publishing their data through the so-called public open data.

The work introduced in this paper forms part of a wider research project called High-quality Open data Publishing and Enrichment[1] (HOPE), whose main goal is the development of a web-based open data management system addressed to public and private organizations and lay users. The former can use the system to release semantically-annotated and high-quality open data, while the latter can access such data in a user-friendly fashion.

Despite the increased availability of open data, various aspects still hinder its access to the general public, such as the need for several pre-processing steps before data can be of any use, or the availability of querying tools, once such data is properly encoded. Formal query languages, with SPARQL being the most representative example, are typically used to access and query the linked open data repositories available nowadays. However, even users with advanced skills will experience difficulties querying new data through standard SPARQL endpoints for at least two reasons: (i) schema behind data may not be known in advance, (ii) values may be stored with a syntax that may not be obvious in advance. These two issues, together with the complex

IIR 2021 – 11th Italian Information Retrieval Workshop, September 13–15, 2021, Bari, Italy

diego.marcia@unica.it (D. Marcia); manuela.sanguinetti@unica.it (M. Sanguinetti); atzori@unica.it (M. Atzori)
https://msang.github.io/ (M. Sanguinetti); http://atzori.webofcode.org/ (M. Atzori)
0000-0003-2576-8836 (D. Marcia); 0000-0002-0147-2208 (M. Sanguinetti); 0000-0001-6112-7310 (M. Atzori)

[1]hope-prin.org

syntax of query languages like SPARQL for lay users, hugely limit the full power of the available open data.

Several attempts have been made over the years to automatically generate structured queries from natural language questions [1, 2, 3, 4, 5] (or conversely, to translate structured queries into natural language [6]).

In this project, we address these issues with different approaches that may fit different applications and kind of users. One of these in particular is based on the *By-Example Structured Queries* (*BEStQ*), that is a user-friendly visual interface in which some example data from the dataset is shown to the users before they start their query.

The present paper describes the current state of development of the BEStQ interface, together with some still open issues regarding data presentation and input interpretation.

2. By-Example Structured Queries

As described above, BEStQ is a visual approach to query databases and in particular knowledge graphs, in a user-friendly fashion [7, 8]. BEStQ, in turn, builds upon and generalizes a previous tool for Wikipedia called SWiPE [9], introduced as a way to query Wikipedia in a structured way [7], therefore relying on DBpedia (its structured counterpart at that time) without the need to write complex SPARQL queries, and then extended to support temporal queries [8] and mobile screens [10].

As an example of interaction in SWiPE, the user looking for “politicians born in Palermo”, would first find an example Wikipedia page for a public figure, then input the constraint in the proper fields: “Palermo” in the infobox row *Birth place* and “politician” in the row *Occupations*. Then by clicking a button they would find the results.

Although this Query-By-Example-like approach from SWiPE is user friendly and simple to learn, a number of problems had to be addressed in the transition to BEStQ, which is not limited to Wikipedia and aims at adapting the approach to generic SPARQL endpoints. One important limitation was the need of an infobox-like synoptic table to be used as an input form for the user to insert their constraints. SWiPE used infoboxes from Wikipedia, while in the development of BEStQ we have studied a general way to generate such form, and created a prototype[2] that generates it automatically out of any endpoint by querying it for its attributes.

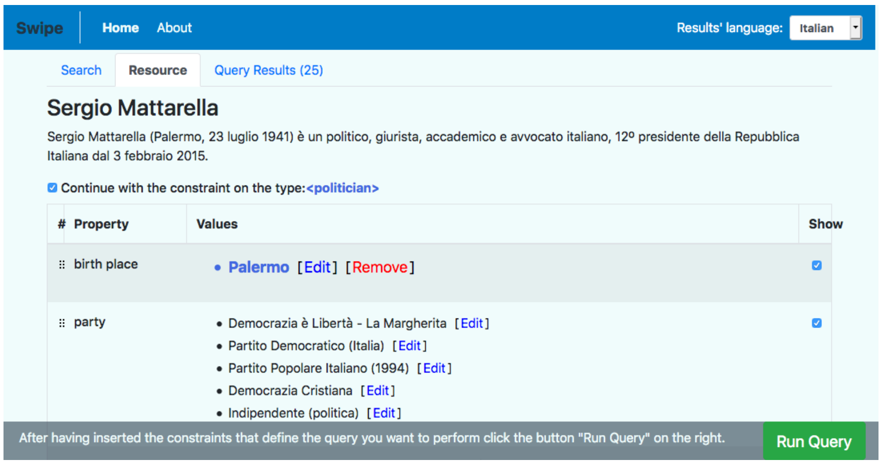
The high-level use flow is similar to the one in SWiPE. After specifying the desired SPARQL enpoint, the user first searches by name for a starting resource they deem representative of the desired result set. In this phase, they can specify two constraints: the locale language they would prefer the results in, and a specific type to retrieve the entity from. Regarding the first option, English is used as fallback language, while the default behaviour for the type filter, in case the user doesn’t specify any type, is a search based on name only. The goal of this type filter is to speed up the search for the example entity, and can be dropped in the subsequent phase. Once the example instance is retrieved, the user can apply the desired constraints on its properties. The allowed constraints depend on the range of values declared by the knowledge graph for the specific properties: the current prototype allows equality checks over properties of all types, and substring matching on string literals only. For other types, it also provides

[2]github.com/atzori/hope-bestq

regular expression matching as a fallback, but we plan on supporting more types.
Users can also apply a *presence* constraint on properties, without specifying the desired values. Finally, in case a specific type has been selected while looking up the example resource, the user can choose to keep this constraint and limit the final search to the same explicit type of the example resource, or remove it.

Fig. 1 shows a screenshot from the current prototype for this step in the search process.

A SPARQL query will be performed on the endpoint according to these user-generated constraints, and the results will be presented to the user with a preview comprised of all properties on which a constraint has been specified.



Sergio Mattarella

Sergio Mattarella (Palermo, 23 luglio 1941) è un politico, giurista, accademico e avvocato italiano, 12º presidente della Repubblica Italiana dal 3 febbraio 2015.

☑ Continue with the constraint on the type:<politician>

# Property	Values	Show
birth place	• Palermo [Edit] [Remove]	☑
party	• Democrazia è Libertà - La Margherita [Edit] • Partito Democratico (Italia) [Edit] • Partito Popolare Italiano (1994) [Edit] • Democrazia Cristiana [Edit] • Indipendente (politica) [Edit]	☑

After having inserted the constraints that define the query you want to perform click the button "Run Query" on the right.

Figure 1: Example of a query for politicians born in Palermo: starting from an example resource of a politician ("Sergio Mattarella"), the user inserts a constraint for *birth place* ("Palermo"). Unbound properties like *party* can be selected for showing up in the search results preview.

Although this implementative effort is an important step to apply BEStQ searches to any structured sources (not only DBpedia), some other research problems arise in order to make this interface really effective.

Finding a good example to start with. Our current research agenda is focusing on the design of a hybrid approach in which the user would first write their search goal as in a question answering system, and then the system finds a good example autonomously. This requires to understand the output type for the user question, and match it with those available from the data source (e.g., `dbo:Politician` in DBpedia). Once the type is clear, still a good instance must be selected. One simple approach would be to find the politician with the largest number of attributes, so that the user may have the highest chances to find the fields they need to insert their constraints on.

Another way may be to find a best-fit for the given natural language query from the user. For instance, since they are looking for the name "Alfonso", then politician *Alfonso Bonafede* may be a better example than *Sergio Mattarella*, despite Bonafede's infobox may have less attributes than Mattarella's infobox.

Simplifying the input of user constraints. Once an example page is given, depending on the level of automatic interpretation of the user's natural language question mentioned above, the system may try to fill in the constraint fields for the user. This task would require to find phrases in the question that semantically match some legal attribute-value pairs in the database.

Ranking of infobox attributes. A relevant usability issue that must be faced within the HOPE project is the ranking of attributes. In fact, many data sources contain several attributes, but only a reasonable subset may be deemed relevant and shown to the user. In order to filter out some attributes or to show the most relevant ones on top, we need proper ranking strategies for attributes. We already developed learning-to-rank approaches to this problem [11], and we are now studying the problem by defining different metrics. According to our ongoing experiments, a good candidate statistics is tf-idf, where we substitute the term-document relatedness score typically used in document retrieval, with property-type relatedness score. This seems to work very well to identify attributes more related with a given type (e.g. `dbp:party` is more specific/relevant than `foaf:givenName` to the type `dbo:Politician`).

Furthermore, should the user find the proposed ranking to suit their needs, our interface already allows them to re-arrange the attribute order. This change in ordering is recorded together with previous ones, and could be fed to the ranking algorithm. In the current prototype, the latest user-defined order is used for all subsequent searches.

Disambiguating types. In our prototype, the query is performed on the basis of the constraints specified by the user for the desired attributes. Keeping in mind the example from previous sections, the query would retrieve all the resources in the knowledge base matching the `birth place=Palermo` constraint and for which the property `party` exists, regardless of their explicit type being different from `Politician`. This allows us to overcome wrong types being assigned to resources, which is the case in those knowledge bases generated by automated extractors, as could likely be the case in the data resulting from the HOPE project's extraction and integration activities carried on heterogeneous and not necessarily certified sources. We leave to the user the option to specify a type for the results to be retrieved.

In future implementations, the use of tf-idf metrics mentioned in the previous section could provide a valuable aid in disambiguating the types for entities in a spurious or incomplete type system, as our ongoing experiments suggest (e.g. the presence of `dbp:party` relates more strongly with type `dbo:Politician` than with type `dbo:HorseTrainer`).

3. Conclusions

The existence of multiple proposals for the question answering task over knowledge graphs suggests the mismatch between the ever-growing relevance and size of open access linked data on one side, and availability to the general public on the other. We realized a prototype of a query-by-example platform that totally decouples user interaction and low-level data structures and interfaces. In the immediate future, our objective is bringing such prototype to stability both in terms of property constraint expressiveness and final query reliability. We have already identified a simple yet satisfactory metric for attribute relevance (namely, attribute-class relatedness). As a next step, we also plan to work on the development of our second approach to user-friendly queries, i.e., the question answering module.

Acknowledgments

This work is funded by PRIN 2017 (2019-2022) project *HOPE - High quality Open data Publishing and Enrichment*.

References

[1] K. B. Cohen, J.-D. Kim, Evaluation of SPARQL query generation from natural language questions, in: Proceedings of the Joint Workshop on NLP&LOD and SWAIE: Semantic Web, Linked Open Data and Information Extraction, INCOMA Ltd. Shoumen, BULGARIA, Hissar, Bulgaria, 2013, pp. 3–7. URL: https://www.aclweb.org/anthology/W13-5202.

[2] M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, J. Lehmann, Asknow: A framework for natural language query formalization in sparql, in: Proceedings of the 13th International Conference on The Semantic Web. Vol. 9678, Springer-Verlag, 2016, p. 300–316.

[3] D. Diefenbach, V. Lopez, K. Singh, P. Maret, Core techniques of question answering systems over knowledge bases: a survey, Knowledge and Information Systems 55 (2018) 529–569.

[4] K. Affolter, K. Stockinger, A. Bernstein, A comparative survey of recent natural language interfaces for databases, VLDB Journal 28 (2019) 793–819.

[5] H. Jung, W. Kim, Automated conversion from natural language query to sparql query, Journal of Intelligent Information Systems 55 (2020) 501–520.

[6] B. Ell, D. Vrandečić, E. Simperl, Spartiqulation: Verbalizing sparql queries, in: E. Simperl, B. Norton, D. Mladenic, E. Della Valle, I. Fundulaki, A. Passant, R. Troncy (Eds.), The Semantic Web: ESWC 2012 Satellite Events, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 117–131.

[7] H. Mousavi, M. Atzori, S. Gao, C. Zaniolo, Text-mining, structured queries, and knowledge management on web document corpora, SIGMOD Rec. 43 (2014) 48–54. doi:10.1145/2694428.2694437.

[8] C. Zaniolo, S. Gao, M. Atzori, M. Chen, J. Gu, User-friendly temporal queries on historical knowledge bases, Inf. Comput. 259 (2018) 444–459. doi:10.1016/j.ic.2017.08.012.

[9] M. Atzori, C. Zaniolo, Swipe: searching wikipedia by example, in: A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, S. Staab (Eds.), Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume), ACM, 2012, pp. 309–312. doi:10.1145/2187980.2188036.

[10] A. Dessi, A. Maxia, M. Atzori, C. Zaniolo, Supporting semantic web search and structured queries on mobile devices, in: R. D. Virgilio, J. Geller, P. Cappellari, M. Roantree (Eds.), 3RD International Workshop on Semantic Search over the Web, SSW '13, Riva del Garda, Italy, August 30, 2013, ACM, 2013, pp. 5:1–5:4. doi:10.1145/2509908.2509910.

[11] A. Dessi, M. Atzori, A machine-learning approach to ranking RDF properties, Future Gener. Comput. Syst. 54 (2016) 366–377. doi:10.1016/j.future.2015.04.018.