Ontology-Schema Based Query by Example

Lucas Peres, Ticiana L. Coelho da Silva(✉), Jose Macedo(✉), and David Araujo(✉)

Insight Data Science Lab, Fortaleza, CE, Brazil
{lucasperes,ticianalc,jose.macedo,david}@insightlab.ufc.br

Abstract. The Web has evolved from a network of linked documents to one where both documents and data are linked, resulting in what is commonly known as the Web of Linked Data, that includes a large variety of data usually published in RDF from multiple domains. Intuitive ways of accessing RDF data become increasingly important since the standard approach would be to run SPARQL queries. However, this can be extremely difficult for non-experts users. In this paper, we address the problem of question answering over RDF. Given a natural language question or a keyword search string, our goal is to translate it into a formal query as SPARQL that captures the information needed. We propose Von-QBE which is a schema-based approach to query over RDF data without any previous knowledge about the ontology entities and schema. This is different from the-state-of-art since the approaches are instance-based. However, it can be unfeasible using such approaches in big data scenarios where the ontology base is huge and demands a large amount of computational resource to keep the knowledge base in memory. Moreover, most of these solutions need the knowledge base triplified, which can be a hard task for legacy bases. Von-QBE results are promising for the two real benchmarks evaluated, considering that only the ontology schema is used to generate SPARQL queries.

1 Introduction

The Web has evolved from a network of linked documents to one where both documents and data are linked, resulting in what is commonly known as the Web of Linked Data, that includes a large variety of data usually published in RDF from multiple domains. Intuitive ways of accessing RDF data become increasingly important since the standard approach would be to run structured queries in triple-pattern-based languages like SPARQL [12]. This can be extremely difficult for non-experts users.

Consider the example question, such as "Find the title of action movies produced in Eastern Asia and the name of their company". A possible SPARQL formulation, assuming a user familiar with the schema of the underlying knowledge base, could consist of the following:

A. H. F. Laender et al. (Eds.): ER 2019, LNCS 11788, pp. 204–212, 2019.
https://doi.org/10.1007/978-3-030-33223-5_17

```
SELECT DISTINCT ?x ?title ?company_name WHERE {
?x a mo:Movie; mo:title ?title;
 mo:isProducedBy ?y; mo:belongsToGenre [ a mo:Brute_Action ] .
?y :companyName ?company_name .
?y :hasCompanyLocation [ a mo:Eastern_Asia ] . }
```

This complex query, which involves multiple joins, is difficult for the user to come up with specific relations, classes and entities. This would require familiarity with the knowledge base, which in general, no user should be expected to have. In this paper, we address the problem of question answering over RDF. Given a natural language question Q_N and an underlying ontology O, our goal is to translate Q_N into a formal query Q_S as SPARQL that captures the information need to be expressed by Q_N. We focus on queries that emphasize classes and relations between them. We do not consider aggregation, disjunctive and negation queries.

A considerable number of question answering approaches for RDF data has been proposed, to name a few [1,8,9,11,13] and [12]. They address the same problem of this paper. However, they present several limitations: [1,9,13] and [11] are instance-based approaches, which can be unfeasible in big data scenarios where the ontology base is huge and demands a large amount of computational resource to keep the knowledge base in memory. Moreover, most of those solutions need the knowledge base triplified, which can be a hard task for legacy bases. [12] requires a pre-processing phase to constructed a phase-concept dictionary as part of the knowledge base, and [8] is based on SPARQL templates. In this paper, we propose Von-QBE (stands for Virtual Ontology Query By Example) that overcome such limitations. Von-QBE derives from the term virtual ontology, since it is not instance-based and it can use an ontology virtualized by other tools like Ontop [3] instead of RDF stores.

Von-QBE is a schema-based approach to query over RDF data without any previous knowledge about the ontology entities and schema. Von-QBE lets the user queries using natural language questions or by using a keyword search and translates the query into SPARQL. Furthermore, Von-QBE assists the user to construct his/her query search interactively. The remainder of the paper is structured as follows: Sect. 2 introduces our proposal Von-QBE. Section 3 presents the experimental evaluation, and finally Sect. 4 draws the final conclusions.

2 Von-QBE Framework

In this section, we introduce our proposal. Given a natural language question Q_N (or a keyword search string) and an ontology base O, Von-QBE translates Q_N into a SPARQL query Q_S that can capture the same information expressed by Q_N. Beyond that, Von-QBE also helps the user to construct its keyword search interactively. For the sake of brevity, from now on we will use Q_N in place of the natural language question, and the keyword search string since each word in the natural language question is tackled as a keyword.

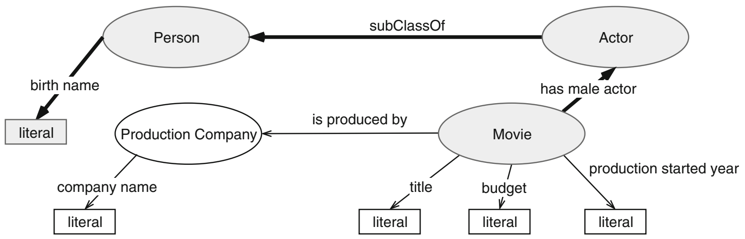

Fig. 1. Part of IMDB ontology schema with the highlighted fragment for the search: *movies and their actors birth name*

Suppose the ontology schema in Fig. 1[1] represented as an RDF graph based on [4], where the classes are graph nodes and the properties, edges. Imagine a user that starts Q_N with the keyword *movie*. Von-QBE suggests improving Q_N using concepts from the ontology schema and whenever Q_N represents what the user is searching for, the user can submit Q_N to Von-QBE. Then, Von-QBE transforms Q_N into a SPARQL query Q_S and returns the answers.

Von-QBE comprises three main components: (1) *Fragment Extractor* responsible to, from Q_N, identify the ontology subset involved in the query. Throughout this paper, we call such subset as *fragment*; (2) *Fragment Expansor* which expands the fragment with classes and properties, i.e. ontology concepts. Based on this expansion, *Fragment Expansor* suggests new ontology concepts to the user expands Q_N; and finally, (3) *Query Builder* which transforms the fragment into a SPARQL query Q_S. In what follows, we describe in details each Von-QBE component.

Fragment Extraction. As we mentioned before a *fragment* from the underlying RDF graph corresponds to the classes and properties of the ontology schema involved on Q_N. The *Fragment Extraction* is made up by two main components: (1) *Keyword Matcher* that identifies the ontology concepts mentioned on Q_N, and (2) *Fragment Constructor* that discovers how these concepts are related on the ontology schema. Consider the ontology schema presented in Fig. 1 and Q_N as "Give the movie actors". The *Keyword Matcher* would identify the classes *Movie* and *Actor* that yield the highest similarity value with the terms of Q_N. From these classes, the *Fragment Constructor* would extract the fragment *Movie has male actor Actor*, once the class *Movie* directly achieves *Actor* in the RDF graph.

Now consider Q_N as "Find the birth name of actors from movies". The *Keyword Matcher* would identify the classes *Actor* and *Movie*, and the property *birth name*. However, *birth name* is not a property of *Actor* neither a property of *Movie* in the RDF graph. So *Fragment Constructor* identifies the class *Person* and the property *subClassOf* to relate *birth name* with *Actor*. Finally, the fragment is built by means of the relation *has male actor* that relates *Movie* and *Actor*, the relation *subClassOf* that relates *Actor* and *Person*, and the relation

[1] https://sites.google.com/site/ontopiswc13/home/imdb-mo.

birth name which is a property of *Person*. The generated fragment is highlighted in Fig. 1.

Algorithm 1 performs the *Keyword Matcher*. First of all, we prefer to use Jaro-Winkler [10] as the similarity measure since it is widely used. Algorithm 1 receives as input a list of *words* from Q_N, the ontology schema (*RDFSchema*) and a similarity threshold ρ. It outputs the ontology concepts (classes and/or properties) that match with *words*. For each word (line 3) in Q_N, *testWord* appends such word with the previous words (*composedConcept*) in Q_N such that they together are similar to a concept in *RDFSchema* (Line 4). So, Line 5 checks if there is any concept on *RDFSchema* such that the similarity between *testWord* is greater than a threshold ρ. If so, the algorithm updates *composedConcept* (Line 6) in order to keep in such variable a sequence of words in Q_N that matches an ontology concept according to ρ. If the similarity is not greater than ρ, the algorithm adds into *elements* list the highest similar ontology concept with *composedConcept* (Lines 8 and 9), and updates the *composedConcept* variable to the current analyzed word. The intuition behind is from that word the algorithm might start a new sequence of words that match with any ontology concepts. Algorithm 1 needs to check if there is any ontology concept that is similar to the last value assigned to *composedConcept* at Line 13. If so, such concept is added into the *elements* list (Lines 14 and 15). Line 17 returns the output of the algorithm.

The output of Algorithm 1 is given as input *Fragment Constructor* module, which builds the fragment that relates the ontology concepts involved in Q_N according to the RDF Schema (also given as input). First, it computes the closure

Algorithm 1. Algorithm Keyword Matcher

```
   Data: words, RDFSchema, ρ
   Result: elements //ontology elements
1  elements := {}; n :=(words.length-1); composedConcept := ”” ;
2  for i := 0 to n-1 do
3      word = words[i] ;
4      testWord = composedConcept + word ;
5      if similarConcept(testWord, RDFSchema) > ρ then
6          composedConcept = testWord ;
7      else
8          concept := RDFSchema.getMostSimilarConcept(composedConcept);
9          elements.add(concept) ;
10         composedConcept = word ;
11     end
12 end
13 if similarConcept(composedConcept, RDFSchema) > ρ then
14     concept := RDFSchema.getMostSimilarConcept(composedConcept) ;
15     elements.add(concept) ;
16 end
17 return elements
```

graph [5] which is a subgraph constructed by using the shortest paths (obtained by running Dijkstra algorithm) among all the pair of ontology concepts returned by Algorithm 1. This graph might have cycles which are often found on RDF ontologies. Imagine two properties that are one inverse of another, like *has male actor* that connects *Movie* to *Actor* and *is male actor in*, connecting *Actor* to *Movie*. To remove these cycles, the module applies Prim's algorithm [6] to find the Minimum Spanning Tree(MST). Prim outputs a fragment smaller or equal than the closure graph. This means that such a fragment contains only the minimal number of paths to connect all the ontology concepts outputted by Algorithm 1. Of course, the closure graph might have multiple MSTs. However, Prim only takes one of them.

After the computation of the fragment, Von-QBE starts two other components: (1) *Fragment Expansor* which expands the fragment with ontology classes and properties. Based on this expansion, *Fragment Expansor* suggests new ontology concepts to the user expands Q_N; (2) *Query Builder* which transforms the fragment into a SPARQL query Q_S.

Fragment Expansor. Von-QBE suggests the user expands Q_N using the ontology classes and properties that are directly connected to the fragment. The *Fragment Expansor* expands the fragment with all edges (of course, the ones that are not already in the fragment) that come in (or out) from the fragment nodes. Remember that our ontology is represented as an RDF graph, and the fragment nodes are ontology classes. Let Q_N be "Find the movies and their actors" and the underlying ontology is the RDFSchema represented in Fig. 1. The fragment nodes derived from Q_N are *Movie* and *Actor*. From the node *Movie*, *Fragment Expansor* can find the following properties: *title*, *budget*, *production started year*, *is produced by* and *has male actor*. However, the *has male actor* is already in the fragment. So, only the other properties should be presented as a suggestion to the user expands Q_N. The node *Actor* contains a particular case. Consider the property *has male actor* is already in the fragment. In this case, *Fragment Expansor* can only suggest *subClassOf*. However, when a class is a subclass of another, it must inherit the properties from the parent class. So, instead of suggesting *subClassOf*, *Fragment Expansor* suggests to the user *birth name*.

Query Builder works as follows: each edge in the fragment (output of *Fragment Constructor* module or expanded with the suggestions outputted by *Fragment Expansor* module and accepted by the user) is added as a clause (or triple pattern), and the source and the target nodes are named as variables. Since the ontology schema might have properties that present multiple domains and ranges, *Query Builder* also adds a clause to inform the instance type (class) of each variable. All the clauses (or triple patterns) are given as input to Apache Jena library[2] which generates Q_S according to the SPARQL syntax. Let Q_N be: "Find the birth name of actors from movies" and the fragment with the following relations: *Movie has male actor Actor*, *Actor subClassOf Person* and *Person birth name*. The second edge relates to the property *subClassOf*. Whenever *Query*

[2] http://jena.apache.org.

Builder finds an edge with *subClassOf*, it replaces such edge by other new edges that have the parent class as a source. So considering the *RDFSchema* represented in Fig. 1, *Query Builder* generates the following edges: *E1: Movie has male actor Actor* and *E2: Actor birth name*.

The edge *E1* is a relation connecting two classes. Since this is the first edge to be processed, we have two new variables (one for *Movie* and one for *Actor*). *Query Builder* also adds three clauses: a definition for the *Movie* variable, a definition for the *Actor* variable and the connection of both variables using the property *has male actor*. So, the edge *E1* corresponds to the three following triple patterns: *TP1: ?Movie a imdb:Movie*, *TP2: ?Actor a imdb:Actor* and *TP3: ?Movie imdb:has_ male_ actor ?Actor*. The edge *E2* is a property that points to a literal value (also called attribute or, in RDF, *DataTypeProperty*). This kind of edge needs to use also two variables, but only the first one is a class instance, the other variable is a literal value (strings, numbers, dates, among others). Since the *Actor* variable is already defined, *Query Builder* only needs to add one more triple pattern: *TP4: ?Actor imdb:birth_ name ?Actor_ birth_ name*. Once all the triple patterns are generated from the fragment edges, Apache Jena generates the SPARQL query Q_S as follows:

```
SELECT ?Movie ?Actor ?Actor_birth_name WHERE{
?Movie a imdb:Movie .
?Actor a imdb:Actor .
?Movie imdb:has_male_actor ?Actor .
?Actor imdb:birth_name ?Actor_birth_name .}
```

3 Experiments

In this section, we provide the details about the experimentation performed with Von-QBE. No experiments were made to compare Von-QBE to other solutions previously mentioned since these works use the ontology data instances to improve their performance, which would not be a fair comparison. From the authors' knowledge, OptiqueVQS [7] and Von-QBE are the only solution schema-based, but OptiqueVQS does not accept natural language or keyword search then we can not compare with OptiqueVQS as well. Instead, we experimented Von-QBE with two real benchmarks. For each benchmark, we evaluate each question is evaluated according to well-established metrics, i.e., *recall* and *precision*.

3.1 Datasets

Our experiments are based on two collections of questions: IMDB Movie Ontology[3] virtualized using Ontop [3] with questions formulated in SPARQL query[4]. IMDB provides data about actors, movies, directors, and production company.

[3] https://sites.google.com/site/ontopiswc13/home/imdb-mo.

[4] https://raw.githubusercontent.com/wiki/ontop/ontop/attachments/Example_MovieOntology/movieontology.q.

Another collection of questions are the QALD[5] task for question answering over linked data. It comprises two sets of questions over DBpedia [2], annotated with SPARQL queries and answers. We used QALDs 5, 6, 7 and 9 training questions which are provided with a SPARQL benchmark. For both benchmarks, the questions out-of-scope for Von-QBE were not considered, like ASK type questions, aggregation, and counting. Moreover, the ones that no entities are available (empty results).

After removing these questions, our test set consists of 12 QALD-(5, 6, 7, 9) training questions and 29 Ontop questions out of 37. The number of evaluated questions in QALD-(5, 6, 7, 9) reduces by much, because these questions comprise information about instances, like people names, country names, aggregations, sorting, while Von-QBE works with conceptual questions, using classes and properties names only.

3.2 Evaluation Results

Table 1 lists the results of each experiment using IMDB and QALD datasets. To both benchmarks, we set 0,9 as the similarity threshold (ρ) in the *Keyword Matcher* algorithm. IMDB contains some questions with low results for precision, like the keyword search question 24: "title movies company name production company located East Asia", for example, demands the movie title and company names located in East Asia. Von-QBE generates a SPARQL query that projects all properties and entities used in the triple-patterns on the SELECT clause, then the movies and companies URIs, titles, and names are returned. However, there exist movies with the same title but different IDs, so Von-QBE retrieves more answers than the benchmark. This decreases precision.

Table 1. Experiment results for IMDB and QALD-(5, 6, 7, 9) datasets.

Dataset	Questions	Select-project questions	Answerable questions	Recall (R)	Precision(P)
IMDB	37	37	29	0.96	0.69
QALD-5	286	269	1	0.2	$1 \times 10e-4$
QALD-6	335	308	5	0.55	0.001
QALD-7	215	193	2	0.39	$3 \times 10e-5$
QALD-9	408	371	4	0.66	0.013
Weighted mean				0.53	0.0047

Another drawback is for questions that use entities to filter the results, like question 271 from QALD-9: "Which awards did Douglas Hofstadter win?". The SPARQL gold standard retrieves only the awards from *Douglas Hofstader* while the SPARQL generated from Von-QBE retrieves all the awards. This happens because VON-QBE is schema-based only, and *Douglas Hofstader* is an entity,

[5] http://qald.aksw.org.

not a concept. This has a major impact in the QALD results, since only a really small portion of the questions, Von-QBE can retrieve any result. Analyzing only the queries with answers, we still get some acceptable recall results using only the schema. We plan to enhance VON-QBE by using, for instance, Named Entity Recognition tools to detect for each entity described in the query (like *Douglas Hofstader*) its corresponding class. The evaluation data can be found at[6] and a demonstration of Von-QBE can be found at[7].

4 Conclusion and Future Work

In this paper, we propose Von-QBE to address the problem of translating a natural language question or a keyword search over RDF data into SPARQL query. From the authors' knowledge, Von-QBE is the first work to address such a problem using only the ontology schema. We believe our results are promising for the two real benchmarks evaluated, considering that only the ontology schema was used to generate SPARQL queries. As future work, we aim at using natural language processing tools to detect entities described in the query and find its corresponding concept over the ontology schema. Moreover, we aim at expanding Von-QBE to process different types of queries, like aggregation.

Acknowledgments. This work has been supported by FUNCAP SPU 8789771/2017 research project.

References

1. Arnaout, H., Elbassuoni, S.: Effective searching of RDF knowledge graphs. J. Web Semant. **48**, 66–84 (2018)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Calvanese, D., et al.: Ontop: answering SPARQL queries over relational databases. Semant. Web **8**(3), 471–487 (2017)
4. World Wide Web Consortium, et al.: RDF 1.1 concepts and abstract syntax
5. Kompella, V.P., Pasquale, J.C., Polyzos, G.C.: Multicast routing for multimedia communication. IEEE/ACM Trans. Netw. (TON) **1**(3), 286–292 (1993)
6. Prim, R.C.: Shortest connection networks and some generalizations. Bell Labs Tech. J. **36**(6), 1389–1401 (1957)
7. Soylu, A., Kharlamov, E., Zheleznyakov, D., Jimenez-Ruiz, E., Giese, M., Horrocks, I.: OptiqueVQS: ontology-based visual querying. In: VOILA@ ISWC, p. 91 (2015)
8. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: Proceedings of the 21st International Conference on World Wide Web (2012), pp. 639–648. ACM (2012)

[6] https://github.com/InsightLab/linked-graphast/tree/evaluation.
[7] https://github.com/InsightLab/von-qbe.

9. Usbeck, R., Ngomo, A.-C.N., Bühmann, L., Unger, C.: HAWK – hybrid question answering using linked data. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 353–368. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18818-8_22
10. Winkler, W.E.: The state of record linkage and current research problems. Statistical Research Division, US Census Bureau, Citeseer (1999)
11. Xu, K., Zhang, S., Feng, Y., Zhao, D.: Answering natural language questions via phrasal semantic parsing. In: Zong, C., Nie, J.Y., Zhao, D., Feng, Y. (eds.) Natural Language Processing and Chinese Computing. CCIS, vol. 496, pp. 333–344. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45924-9_30
12. Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., Weikum, G.: Natural language questions for the web of data. In: Proceedings of the 2012 Joint Conference on EMNLP and CoNLL, pp. 379–390. Association for Computational Linguistics (2012)
13. Yih, S.W.-T., Chang, M.-W., He, X., Gao, J.: Semantic parsing via staged query graph generation: question answering with knowledge base