Towards Knowledge Graph-Agnostic SPARQL Query Validation for Improving Question Answering

Aleksandr Perevalov[1,2][0000−0002−6803−3357], Aleksandr Gashkov[3][0000−0001−6894−2094], Maria Eltsova[3][0000−0003−3792−8518], and Andreas Both[2,4][0000−0002−9177−5463]

[1] Anhalt University of Applied Sciences, Köthen, Germany
[2] Leipzig University of Applied Sciences, Leipzig, Germany
[3] Perm National Research Polytechnic University, Perm, Russia
[4] DATEV eG, Nuremberg, Germany

Abstract. A Knowledge Graph Question Answering (KGQA) system needs to generate a SPARQL query over a knowledge graph (KG) that is reflecting a user's information need expressed by the given natural-language question. Yet, many of these generated queries might be completely mismatching. To deal with this problem, we developed a KG-agnostic approach that is intended to increase the KGQA quality while validating SPARQL query candidates and finally removing the incorrect ones. In this demonstration, we provide the research community a Web user interface and a RESTful API to experiment with the processing of our approach and experience the possible impact of such an approach.

Keywords: question answering · knowledge graphs · query validation · query ranking · API · demonstrator.

1 Introduction

The Web was established to manifest a major information source for many people worldwide. While aiming at better knowledge modeling and representation, the Semantic Web initiative was proposed and is permanently growing. The objective of this initiative is to make Web data machine-readable and machine-understandable by describing concepts, entities, and relations between them [1]. Today, the Semantic Web may be considered as a giant Knowledge Graph (KG). In this regard, Knowledge Graph Question Answering (KGQA) systems are actively developing already for more than a decade [4,5].

KGQA systems typically transform a natural-language question to produce a set (or ranked list) of SPARQL[5] *Query Candidates* (QC), i.e., a common interface is present for input information (question text) and output information (SPARQL query), where the first QC should be usable to retrieve the correct answer from the considered KG. There are different structures in the

[5] `https://www.w3.org/TR/rdf-sparql-query/`

KGs (e.g., different basic graph modeling patterns[6]). However, the *label attribute* (e.g., `rdfs:label`[7]) is one of the few common properties used very often in KGs. Therefore, such commonly used attribute like `rdfs:label` enables us to transform the queries in a way that they can be "understood" without depending on structures of particular KGs. In our previous research, we introduced a KG-agnostic approach for validating SPARQL query candidates with the intention to decide whether one could be used to retrieve the correct answer to the given question or not [8].

The approach allows filtering out the incorrect SPARQL queries from the query candidates list by training a binary classifier on positive and negative examples (considering questions and queries) to distinguish between correct and incorrect queries, respectively. Hence, the possible impact of the eliminating the incorrect query candidates is as follows: (1) improving the QA quality by moving the correct QCs to the top of the list and by reducing the cardinality of the QCs list, consequently (2) improving the efficiency of a KGQA system if it executes top-N candidates from the list, and (3) decreasing the probability of misinformation by removing all the QCs given an unanswerable question.

In our recent research papers [7,8], we have demonstrated the applicability of the approach on different KGQA benchmarks' datasets. Specifically, we observed that the query validation approach helps to relatively increase QA quality by up to 382% given Precision@1 (=NDCG@1) score.

In this demo[8], we are providing an interactive Web user interface (Web UI) and a corresponding RESTful API for interacting with the SPARQL query validation model. The Web UI also enables its users to send the feedback given a particular prediction to contribute for further evaluation and optimization of the approach. The API provides the research community with the opportunity to integrate the corresponding approach into their QA systems and to test it interactively.

This paper is structured as follows: Section 2 briefly describes the suggested approach. In Section 3 the demo application is presented. We draw conclusions in the Section 4.

2 Query Candidate Validation

Our approach is based on the assumption that a SPARQL query expressing a user's information need can be transformed to a natural-language text (i.e., verbalized) that should be similar to the original question. Therefore, we assume that it is possible to establish such a decision algorithm that can determine whether a SPARQL query is correctly representing a question (i.e., to validate a SPARQL query with respect to the original question).

[6] `https://www.w3.org/TR/rdf-sparql-query/#BasicGraphPatterns`
[7] `https://www.w3.org/2000/01/rdf-schema#label`
[8] The demo application and the video of presentation are available at `http://demos.swe.htwk-leipzig.de/sparql_query_validation_demo`.

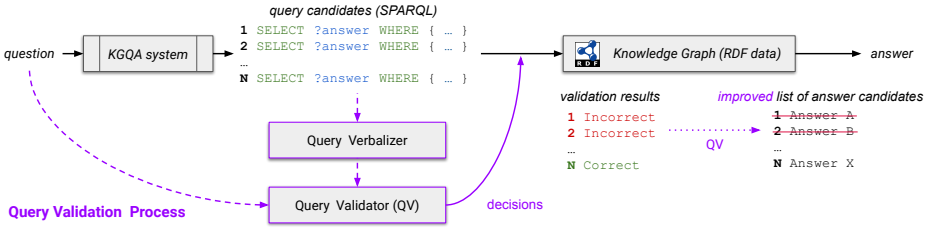

Fig. 1. General overview of the Query Validation process. The core component is the *Query Validator* intended to filter incorrect query candidates.

The general overview of the query validation approach is given in Figure 1. The core component of the entire process is the *Query Validator (QV)* module. This module receives question text and verbalizes SPARQL query as an input. The query is verbalized by replacing URIs with its `rdfs:label` value and leaving all the variable names while removing the rest of the query [8]. The QV module makes binary decisions on whether a *Query Candidate* (QC) is correct or not, given a textual question (in our research paper QAnswer [3] is used for computing the QCs). Therefore, for each QC a decision is done, the set of these decisions form the list of *validation results*. If a decision was negative, then a QC is eliminated, otherwise it stays in the list of query candidates. Hence, a new (probably improved) list of validated query candidates is created. Thereafter, (depending on the used KGQA system) all validated queries might be executed on a KG to get the actual answer. Finally, the obtained answers are used for answering the question. The more detailed description of the QV approach is available in our corresponding research paper [8].

3 Demo Application

The demo application consists of two parts: Web interface and RESTful API. The interface allows a user to "play" with the QV approach in interactive mode (see Figure 2). One should fill the "Question text" and "SPARQL query" fields. Alternatively, one of the examples on the right side of the screen can be selected to make the system infer the validation decision in the "Results" area. The examples are taken from the well-known QA benchmark dataset LC-QuAD [6]. Each result contains the entered question, the query and its verbalization, the confidence score, the validation decision, and also will show two feedback buttons. The RESTful API is used by the Web UI and can also be called independently. Therefore, we require the same input parameters for the `GET` request (`question` and `query`). The response returns `validationResult` and `confidence`. We intend to use the users' feedback collected in our future work to improve our approach. We encourage researchers and developers to integrate the QV approach in their systems and applications using the provided API or even establish reusable QA components (e.g., using a QA framework like [2].

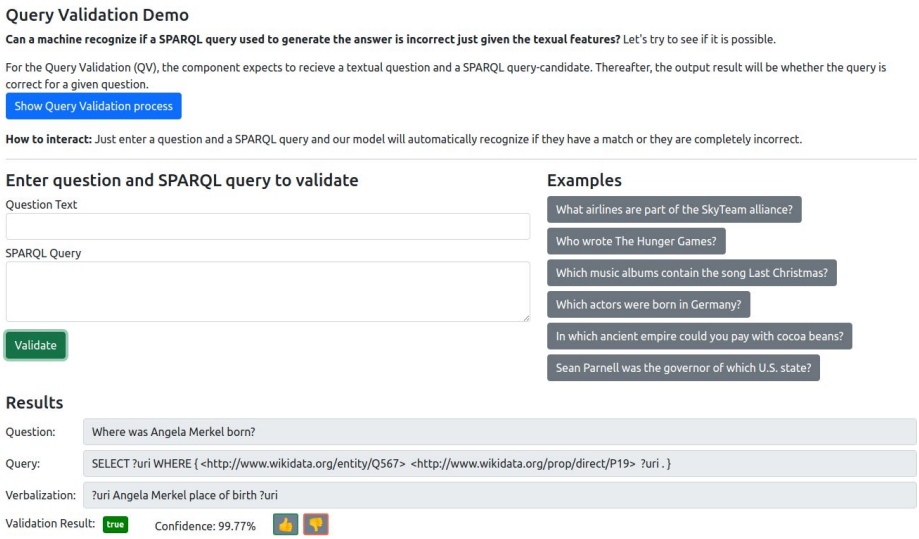

Fig. 2. Screenshot of the demo interface

4 Conclusion

Providing high quality for Question Answering is crucial to enable information access for any user (without the need to learn a technical query language like SPARQL). Here, we provide access to an KG-agnostic approach for analyzing SPARQL queries and deciding whether these queries are correct w.r.t. the given question or not. Hence, our demonstrator enables users to experience possible straightforward verbalizations of generated SPARQL queries, which hopefully will also inspire additional integrations and novel approach with the intention to increase the quality of QA methods. Additionally, our implemented API can be used to test the functionality with an own dataset and to easily integrate our approach into existing QA systems. In the future, we will integrate different verbalizations into our approach to enable users to experience the impact of different verbalization methods.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5), 34–43 (2001), `http://www.jstor.org/stable/26059207`
2. Both, A., Diefenbach, D., Singh, K., Shekarpour, S., Cherix, D., Lange, C.: Qanary–a methodology for vocabulary-driven open question answering systems. In: The Semantic Web. Latest Advances and New Domains. European Semantic Web Conference (ESWC 2016). pp. 625–641. Springer, Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-34129-3_38

3. Diefenbach, D., Both, A., Singh, K., Maret, P.: Towards a question answering system over the semantic web. Semantic Web **11**, 421–439 (2020). https://doi.org/10.3233/SW-190343
4. Diefenbach, D., Lopez, V., Singh, K., Maret, P.: Core techniques of question answering systems over knowledge bases: a survey. Knowledge and Information systems **55**(3), 529–569 (2018). https://doi.org/10.1007/s10115-017-1100-y
5. Dimitrakis, E., Sgontzos, K., Tzitzikas, Y.: A survey on question answering systems over linked data and documents. Journal of intelligent information systems **55**(2), 233–259 (2020). https://doi.org/10.1007/s10844-019-00584-7
6. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia. In: The Semantic Web – International Semantic Web Conference 2019 (ISWC 2019). pp. 69–78. Springer (2019). https://doi.org/10.1007/978-3-030-30796-7_5
7. Gashkov, A., Perevalov, A., Eltsova, M., Both, A.: Improving the question answering quality using answer candidate filtering based on natural-language features. In: 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2021) (2021)
8. Gashkov, A., Perevalov, A., Eltsova, M., Both, A.: Improving question answering quality through language feature-based sparql query candidate validation. In: European Semantic Web Conference (ESWC 2022). Springer (2022)