HAL
open science

SPARQLtoUser: Did the question answering system understand me?

Dennis Diefenbach, Youssef Dridi, Kamal Singh, Pierre Maret

To cite this version:

Dennis Diefenbach, Youssef Dridi, Kamal Singh, Pierre Maret. SPARQLtoUser: Did the question answering system understand me?. ISWC 2017, Oct 2017, Vienne, Austria. hal-01637139

HAL Id: hal-01637139

https://hal.science/hal-01637139v1

Submitted on 17 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SPARQLtoUser: Did the question answering system understand me?

Dennis Diefenbach[1], Youssef Dridi[2], Kamal Singh[1], Pierre Maret[1]

[1] Université de Lyon, CNRS UMR 5516 Laboratoire Hubert Curien
{dennis.diefenbach,kamal.singh,pierre.maret}@univ-st-etienne.fr
[2] Université de Saint-Etienne
youssef.dridi@etu.univ-st-etienne.fr

Abstract. In Question Answering (QA) systems, user questions are automatically converted into SPARQL queries. We present SPARQLtoUser, a simple method to produce a user understandable version of a SPARQL query so that the users can check what the QA system understood. Compared to existing similar methods, SPARQLtoUser is multilingual and is not restricted to one single KB, but easily portable to any KBs. This paper presents our method to generate the representation of SPARQL queries, and compares this method to existing ones. Moreover, we present the results of our user study that show that users have difficulties to understand whether the responses of a QA system are correct or not, with or without SPARQL representations.

Keywords: Question answering system, User feedback, User interaction, SPARQLToUser, SPARQL2NL, Spartiqulator

1 Introduction

Question answering (QA) systems over Knowledge Bases (KB) are aiming to directly deliver an answer to a user's question. Achieving this with high precision is one important research area [6]. Unfortunately the error rates are still very high. For example on popular benchmarks like WebQuestions [1], SimpleQuestions [2] and QALD[3][9] F-measures of respectively 60 %, 70 %, 50 % are realistic. Also the precision is similar meaning that QA systems often do not retrieve the right answer for the user. Since the questions in the above mentioned benchmarks are generally all answerable over the given data, the F-measures in real scenarios are even lower. It is therefore important that users are able to understand if the information provided by a QA system is the one they asked for, so that they can easily estimate if they can believe the answers or not. In the following, we present a tool called SPARQLtoUser, which given a SPARQL query, produces a representation of it that can be shown to end-users. We compare our method with existing ones, namely SPARQL2NL [8] and Spartiqulator [7]. Moreover, we show for the first time that users have difficulties to distinguish right answers from false ones.

[3] http://www.sc.cit-ec.uni-bielefeld.de/qald/

2 Related work

QA over Knowledge Bases is a well-studied problem in computer science. For an overview of the published approaches we refer to [6].
The task of presenting a SPARQL query to an end-user was tackled by verbalizing the query, i.e. by reformulating the SPARQL query using natural language. There are mainly two works that address this task: SPARQL2NL [8] and Spartiqulation [7]. Both tools can be used to translate a SPARQL query to an English sentence. Both approaches were tested on DBpedia and MusicBrainz.
Two main applications in the QA domain are used to motivate the need of such tools. The first was pointed out by Ell et al. [7]: "The verbalization allows the user to observe a potential discrepancy between an intended question and the system-generated query.". The second was pointed out by Ngonga Ngomo et al. [8]. In this case, the idea is to verbalize all the SPARQL queries generated by a QA system and to let the user, if the answer is wrong, the possibility to choose a different SPARQL query by selecting the corresponding verbalization. We are not aware of any test of the efficacy of such approaches.

3 A simple representation of a SPARQL query for the user.

Users can ask questions in various languages, and they may target various KBs. Therefore, a generic method to build a representation of SPARQL queries (generated automatically from user questions) which is easily understandable by the users, should support multi-linguality and should be easily portable to different KBs. In this section, we present our method called SPARQLtoUser. We explain its origin and its mechanism.

The need for such a tool was derived from our QA webservice available under `www.wdaqua.eu/qa`. It uses in the backend a QA system called WDAquaCore0 [5] which is integrated into the Qanary Ecosystem [4] and is exposed to end-users through a reusable front-end called Trill [3]. The QA system is multilingual, and we are able to port it easily to different KBs.

We observed that in many situations users would not be able to understand what was the interpretation of their questions by the QA system and, even as a developer, it was difficult to understand it by looking at the SPARQL query. Since the QA system supports different languages, and since different KBs can be queried, we came to develop a generic solution which is valid for any new languages and for new domains (i.e. KBs). In particular, the multilingual constraint brought us to not to try to verbalize the query, but to find a more schematic representation. An example can be seen in Figure 1. The missing verbalization can be seen as an additional burden for the user. One objective of this work is to measure and compare this burden to other approaches.

To generate the representation, we keep the original order of the triple patterns of the query and we substitute the URIs with the label in the corresponding language,

i.e., the triple patterns

$$\begin{array}{lll} s_{1,1} & s_{1,2} & s_{1,3} \\ \vdots & & \\ s_{k,1} & s_{k,1} & s_{k,1} \end{array}$$

(where s indicates a slot) are converted to

$$\begin{array}{lllll} f(s_{1,1}) & slash(s_{1,1}) & g(s_{1,2}) & slash(s_{1,3}) & f(s_{1,3}) \\ \vdots & & & & \\ f(s_{k,1}) & slash(s_{k,1}) & g(s_{k,2}) & slash(s_{k,3}) & f(s_{k,3}) \end{array}$$

where

$$f(s_{i,j}) = \begin{cases} label(s_{i,j}), & \text{for } s_{i,j} \text{ a URI} \\ \text{""}, & \text{for } s_{i,j} \text{ a variable} \\ s_{i,j}, & \text{for } s_{i,j} \text{ a literal} \end{cases}$$

$$g(s_{i,j}) = \begin{cases} label(s_{i,j}), & \text{, for } s_{i,j} \text{ a URI} \\ \{label(s) \mid (s \text{ a uri such that if } s_{i,j} \text{ is substituted with} \\ s \text{ the triple patterns has a solution in the KB })\} & \text{, for } s_{i,j} \text{ a variable} \end{cases}$$

$$slash(s_{i,j}) = \begin{cases} / & \text{, for } s_{i,j} \text{ a URI} \\ \text{""} & \text{, for } s_{i,j} \text{ a variable} \end{cases}$$

As a concrete example the query:

```
PREFIX wd : <http://www.wikidata.org/>
SELECT DISTINCT ?x
WHERE {
    wd:entity/Q79848 wd:prop/direct/P1082 ?x .
} limit 1000
```

is converted to:

```
Southampton (city in Hampshire, England) / population
```

since "Q79848" refers to "Southampton" and "P1082" to the "population".
We want to briefly describe the substitution of the predicates. If a predicate is an URI, we just put its label. If it is a variable, we expand it using all the properties that potentially could be placed in the variable position. This is motivated by the following example. In QA often hidden relations are expressed in the question. For example for the question "films by Charlie Chaplin" the proposition "by" can have different meanings like "film editor, screenwriter, director, composer, producer, cast member". WDAquaCore0 therefore produces as an interpretation:

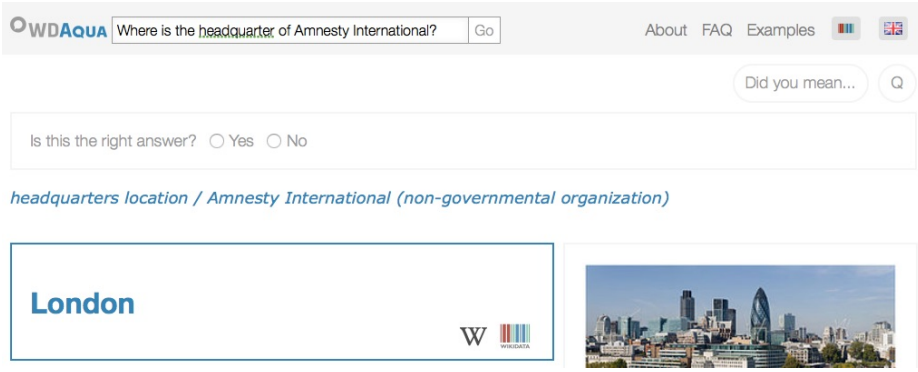

Fig. 1: Snapshot of the Trill front-end. The interpretation generated by SPARQLtoUser is shown in blue over the answer "London").

```
SELECT DISTINCT ?x
WHERE {
    ?x ?p1 <http://www.wikidata.org/entity/Q11424> .
    ?x ?p2 <http://www.wikidata.org/entity/Q882> .
}
limit 1000
```

where the properties are variables. SPARQLtoUser expands the variables ?p1 and ?p2 with the above described meanings. By showing this information, we want to show to the user what the QA system understood and give the users the lexicon to narrow down the search.
Note that the label function can differ from the KB. Generally the strategy is to grab just the label, i.e. rdfs:label. But in some cases this does not identify the entity in an unambiguous way. For example in Wikidata the monument "Colosseum" and the band "Colosseum" have the same label. To identify the entity in a unique way, we add the description in parenthesis, i.e. "Colosseum (elliptical amphitheatre in Rome)" and "Colosseum (British band)". Depending on the KB a different property can be used to grab a verbalization of the resource and a property disambiguating it. Moreover for count queries, we surround the query with "How many" and for an ask query with "check". We limited the support of SPARQLtoUser to the types of queries that WDAquaCore0 can produce, but one can easily think about extension for this approach for other types of queries and aggregators.
Note that the approach is easily portable to a new KB and that it is also multilingual in the sense that if the KB has not English labels the approach still can be applied. Table 1 gives an overview of the main differences between SPARLQ2NL, Spartiqulation and SPARQLtoUser.

The code of SPARQLtoUser can be found at `https://github.com/WDAqua/SPARQL_to_User`. It is released under the MIT licence. The service is exposed as a webservice under the url `https://wdaqua-sparqltouser.univ-st-etienne.fr/sparqltouser`. A description of the API can be found under `https://github.com/WDAqua/SPARQL_to_User/blob/master/README.md`.

Table 1: Main features of SPARQLToUser, SPARQL2NL, Spartiqulator

Tool	SPARQL support	Datasets (tested on)	Languages	Portability
SPARQL2NL	part of SPARQL 1.1 syntax	DBpedia, MusicBrainz	English	no
Spartiqulator	part of SPARQL 1.1	DBpedia, MusicBrainz	English	yes
SPARQLToUser	triple patterns, ASK and SELECT queries, COUNT modifiers	DBpedia, MusicBrainz, Wikidata, Dblp	multilingual	yes

4 A user study to identify the efficacy of SPARQL representations for users.

As mentioned in the related works, one of the main motivations of SPARQL2NL and Spartiqulation was to allow users to better interact with QA systems. While both works performed a users study to test the quality of the verbalisation process, the impact of the SPARQL query representation was never analyzed in the context of QA.

We prepared a survey to test that up to which degree a SPARQL query representation helps the user to understand the interpretation of a user's question by a QA system. To test the effectiveness of a SPARQL representation, we conducted a controlled experiment using A/B testing. In this approach, two versions of the same user-interface, with only one different UI-element, are presented to two different groups of users. The users are not aware that two different UI-elements are being tested. During A/B testing the objective is to measure whether the difference in the UI has an impact on some measurable variable, for example, the click-through rate of a banner advertisement.

Our objective was to test the impact of the SPARQL representation on the ability of a user, to understand the interpretation of a question by a QA system. We, therefore, choose 10 questions and presented to the users the answers that a QA system could have produced. Out of the 10 questions 4 were answered correctly while 6 were answered wrongly. Since our objective is to find out if users are able to determine whether the given answer is correct or not, we ask the users if they believe that the given answer is correct or not. Our objective in the A/B testing is to maximize the number of times the user believes the answer is right, when the answer is actually right, plus the number of times the user believes the answer is wrong, when it is actually wrong. We created 3 different versions of the same UI. The first, denoted **A**, does not contain any representation of the generated SPARQL query, the second, denoted **B**, contains the representation produced by SPARQLtoUser, and the third, denoted **C**, the one produced by SPARQL2NL. We did not test the interpretations given by Spartiqulation because the approach is similar to that of SPARQL2NL. An example of the 3 versions shown to the users can be seen in Figure 2. The list of all questions together with the answers can be found in Table 2. We tried to take questions that do not cover general knowledge so that it is highly probable that the users do not know the correct answer.

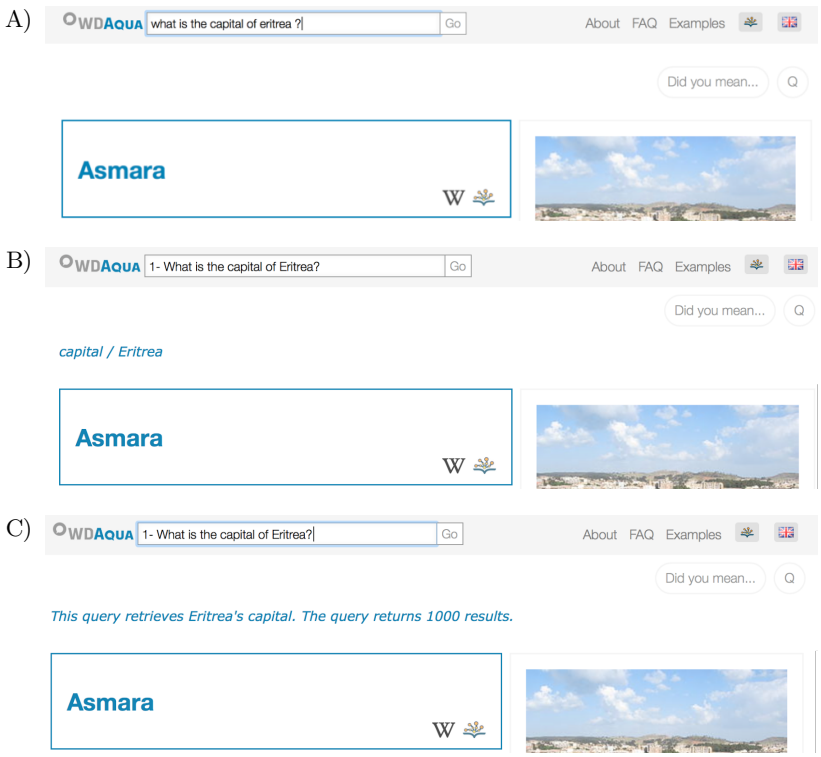

Fig. 2: Three compared versions of the Trill front-end for the question "What is the capital of Eritrea?". A) with no SPARQL representation, B) with the representation of SPARQLtoUser, and C) with the representation of SAPRQL2NL.

The questions with incorrect answers generally present the problem of ambiguity. Sometimes the information provided with the answer, like the description or the top-k properties, allows to decide if the answer is correct or not. In other situations, the user can only choose to trust or not the given answer. The 3 forms send to the groups A, B, C can be found respectively under:

A) `https://goo.gl/forms/vCN7aZEaUAxWNMse2`
B) `https://goo.gl/forms/1fgauBGn5rb4JJ492`
C) `https://goo.gl/forms/WTibSJHjEBRgnaS62`.

The summary of the experimental results can be found in Table 2. In total 3 groups of 16 people answered our survey, i.e. for each version of the UI we have a sample size of 160 (16 people for 10 questions).

The experiments lead to the following conclusions. In general, users have big difficulties to understand the interpretation given by a QA system to the questions asked, independently of the approach. Without a representation of the SPARQL query the error rate is 34 %. With the representation given by SPARQLtoUser the error

N.	Question	Answer	correct	A	B	C
1	What is the capital of Eritrea?	Asmara	yes	12/4	13/3	14/2
2	What is the capital of Ghana?	Koumbi Saleh	no	14/2	12/4	14/2
3	Who is the president of Eritrea ?	Isaias Afwerki	yes	14/2	9/7	12/4
4	In which country was Claudia Cardinale born?	Italy	no	6/10	9/7	8/8
5	Where did Karl Marx die ?	Ladenburg	no	8/8	11/5	14/2
6	In which museum is the Davide of Michelangelo ?	Galleria Borghese	no	12/4	10/6	12/4
7	In which city was Michael Jackson born ?	Fairefax, Virginia	no	9/7	10/6	10/6
8	What is the population of Canada?	36286425	yes	9/7	11/5	9/7
9	Where did Mussolini die ?	Forlì	no	8/8	12/4	9/7
10	Who is the doctoral supervisor of Albert Einstein ?	Alfred Kleiner	yes	14/2	13/3	13/3
Total				106/54	110/50	115/45

Table 2: This table shows the results of the user study. 10 questions different questions where shown to 48 people. These where divided into 3 equally sized groups and received 3 different versions of a UI following an A/B testing methodology. The three groups are denoted A,B,C. The UI shown to group A did not contain any representation of the SPARQL query, group B got the representation of SPARQLtoUser and group C the representation of SPARQL2NL.

rate is 31 % and with the one of SPARQL2NL it is 28 %. Unfortunately, there is no significant difference between the different approaches. Also the statistical power is not high enough to say that there is no significant difference, i.e. we can not state with high enough probability that it makes no different to put or not a SPARQL representation. This is an unexpected result for us. We thought that the impact of the representation of the SPARQL query would be high enough to show a significant difference with a sample size of 160. The experiment shows that the impact is lower than what we expected.

We want to discuss some unexpected observations. For example question 3 got better results without any SPARQL representation. Question 3 was "Who is the president of Eritrea ?". The presented answer is correct and the description of the answer says that clearly: "Isaias Afwerki Tigrinya is the first President of Eritrea, a position he has held since its independence in 1993.". In this case the representation of the SPARQL query given to the user seems to distract the user, since many users answer the question correctly without any interpretation. For the question "In which country was Claudia Cardinale born?" we assume that the QA system did a mistake when disambiguating the entity and returned the result for "Claudia Octavia" instead of "Claudia Cardinale". Both the representations of SPARQLtoUser and SPARQL2NL contain the string "Claudia Octavia" and not "Claudia Cardinale". Still many users believed that the answer is correct. Apparently, here the representation of the SPARQL query was ignored by many users.

5 Conclusion

Since current QA systems over KB still have low F-measures, it is important to let the users understand how the QA system interpreted their questions. We have presented a new representation of a SPARQL query for users. It is not based on verbalization, as done in previews works, but it is based on a simple rewriting of the query. The advantage is that this representation can be used with any language and it can be easily ported to new KBs.
We have for the first time conducted a user study to measure the impact of SPARQL representations on the understand-ability of users of the interpretation of a question by a QA system. We found that if an impact exists then it is much smaller than we believed. More research in this area with quantitative methodologies is needed. In the future, we want to repeat our experiment with a larger number of participants and with more variations of the UI. In particular, we want to create a version that puts more emphasis on the SPARQL representation and one version that clearly states "I interpreted your question as: (followed by the interpretation)". The interaction of the users with a QA system is a poorly addressed, but interesting field.

Acknowledgments Parts of this work received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 642795, project: Answering Questions using Web Data (WDAqua).

References

1. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic Parsing on Freebase from Question-Answer Pairs. In: EMNLP (2013)
2. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)
3. Diefenbach, D., Amjad, S., Both, A., Singh, K., Maret, P.: Trill: A reusable Front-End for QA systems. In: ESWC P&D (2017)
4. Diefenbach, D., Singh, K., Both, A., Cherix, D., Lange, C., Auer, S.: The Qanary Ecosystem: getting new insights by composing Question Answering pipelines. In: ICWE (2017)
5. Diefenbach, D., Singh, K., Maret, P.: Wdaqua-core0: A question answering component for the research community. In: ESWC, 7th Open Challenge on Question Answering over Linked Data (QALD-7) (2017)
6. Diefenbach, D., Lopez, V., Singh, K., Pierre, M.: Core Techniques of Question Answering Systems over Knowledge Bases: a Survey. Knowledge and Information systems (2017 (to appear))
7. Ell, B., Vrandečić, D., Simperl, E.: Spartiqulation: Verbalizing sparql queries. In: Extended Semantic Web Conference. pp. 117–131. Springer (2012)
8. Ngonga Ngomo, A.C., Bühmann, L., Unger, C., Lehmann, J., Gerber, D.: Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In: Proceedings of the 22nd international conference on World Wide Web. pp. 977–988. ACM (2013)
9. Unger, C., Ngomo, A.C.N., Cabrio, E., Cimiano: 6th Open Challenge on Question Answering over Linked Data (QALD-6). In: The Semantic Web: ESWC 2016 Challenges. (2016)