QAnswer KG: Designing a portable Question Answering System over RDF data

Dennis Diefenbach[1][0000−0002−0046−2219], José Giménez-García[2], Andreas Both[3,4], Kamal Singh[2], and Pierre Maret[2]

[1] The QA Company SAS, France, dennis.diefenbach@qanswer.eu
[2] Université de Lyon (France), CNRS UMR 5516 Laboratoire Hubert Curien
jose.gimenez.garcia@univ-st-etienne.fr kamal.singh@univ-st-etienne.fr
pierre.maret@univ-st-etienne.fr
[3] DATEV eG, Germany, andreas.both@datev.de
[4] Anhalt University of Applied Science, Germany andreas.both@hs-anhalt.de

Abstract. While RDF was designed to make data easily readable by machines, it does not make data easily usable by end-users. Question Answering (QA) over Knowledge Graphs (KGs) is seen as the technology which is able to bridge this gap. It aims to build systems which are capable of extracting the answer to a user's natural language question from an RDF dataset.
In recent years, many approaches were proposed which tackle the problem of QA over KGs. Despite such efforts, it is hard and cumbersome to create a Question Answering system on top of a new RDF dataset. The main open challenge remains portability, i.e., the possibility to apply a QA algorithm easily on new and previously untested RDF datasets.
In this publication, we address the problem of portability by presenting an architecture for a portable QA system. We present a novel approach called QAnswer KG, which allows the construction of on-demand QA systems over new RDF datasets. Hence, our approach addresses non-expert users in QA domain.
In this paper, we provide the details of QA system generation process. We show that it is possible to build a QA system over any RDF dataset while requiring minimal investments in terms of training. We run experiments using 3 different datasets.
To the best of our knowledge, we are the first to design a process for non-expert users. We enable such users to efficiently create an on-demand, scalable, multilingual, QA system on top of any RDF dataset.

Keywords: Question Answering · RDF · Knowledge Graphs · Portability · QAnswer · On-demand

1 Introduction

In the last decade, a large number of datasets were published using the RDF standard. RDF allows storing data using a flexible and extensible schema, thus making it possible to store very heterogeneous data. An RDF dataset is generally

referred to as a Knowledge Graph (KG). Nowadays, there are KGs about general knowledge, publications, music, geography, life sciences, and many more[5]. The data published, using the RDF standard, and accessible on the World Wide Web is part of the Semantic Web or Web 3.0.

One of the main goals of the Semantic Web is that data can be easily processed by machines. In contrast, the Web 2.0 concepts mostly address end-users. Semantic Web makes data easily accessible by machines. However, it becomes relatively difficult to interpret for end users, although the contained information is extremely valuable for them.

End-users can access RDF data in different ways. Formats and methods like Turtle, N-triples, JSON-LD, etc., make it possible to access RDF data through serialization. Other possibilities include user interfaces for faceted search on RDF data (like LodLive[6]). Moreover, there exists SPARQL[7], a standardized query language for RDF that allows to retrieve complex information from any RDF dataset. All these possibilities require considerable technical knowledge. Thus, they are restricted only to expert users.

In contrast, Question Answering (QA) over Knowledge Graphs (KGs) aims at accessing RDF data using natural language questions. This is generally accomplished by converting a user's question (expressed in natural language) to a corresponding SPARQL query, whose result set is the answer to the question. This process should be performed in an automatic way. This allows also the non-expert users to access RDF data. While a lot of research was done in the last decade addressing this problem, in general, all proposed solutions queried one or a very few specific RDF datasets. The main problem that was not addressed was portability, i.e., the ability to easily apply and port the developed algorithm to new datasets. This observation is the motivation of our research question: *Is it possible to develop a QA approach which can be easily applied to new datasets with little to no manual work?*

We build on top of a recently proposed approach, namely QAnswer [5] to construct a portable QA system that is multilingual, robust, scalable and supports multiple KGs. The QAnswer approach has already been successfully applied to a number of different, well-known datasets including Wikidata, DBpedia, DBLP, Freebase, MusicBrainz, SciGraph and LinkedGeoData [9].

We design and present an architecture for training and running a QA system. This actually results in an out-of-the-box QA system for a user-defined RDF dataset. We call it *QAnswer KG*. Using our approach, we enable any (non-expert) dataset owners to efficiently create a QA system on top of their dataset, so that it can be accessed and consumed by end users.

This paper is organized as follows. First, we examine related works in Section 2. Then we summarize the approach of QAnswer in Section 3. In Section 4, we present our perspective on the relation between RDF data and the questions

[5] A comprehensive overview of open RDF datasets is available at `http://lod-cloud.net`.

[6] `http://en.lodlive.it`

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

that can be answered on top of it. This section also describes the process of constructing a portable QA system based on the QAnswer approach and the process of training it is using new questions. Additionally, the limitations of the current approach are discussed. We conclude with Section 5.

2 Related Work

Question Answering (QA) over Knowledge Graphs (KGs) is an extensive research area with many challenges. For a global overview, we refer to [8]. The main challenge that is addressed in this work is *portability*, i.e., the ability to easily adapt a QA system to a new RDF dataset. The lack of portability of existing approaches is mainly caused by two problems:

Problem 1) Many approaches rely on machine learning algorithms having a large number of learning parameters and requiring a lot of data. Among them, especially deep learning approaches became very popular in recent years like Bordes et al. [3] and Zhang et al. [18]. The main drawback of these approaches is the training data itself. Creating a new training dataset for a new KG is very expensive. For example, Berant et al. [2], report that they spent several thousand US dollars for the creation of the WebQuestions dataset using an online crowdsourcing marketplace (Amazon Mechanical Turk). This dataset contains 5810 questions. The systems evaluated over the SimpleQuestions[8] dataset (one of the most commonly used benchmarks for studying single-relation factoid questions) use 75910 question-answer pairs for training. The dependency on such large training datasets makes these approaches non-portable unless it is possible to spend very significant effort.

Problem 2) Existing question answering systems depend on KG-specific external tools like entity linkers. Moreover, these works often use manually implemented rules adapted to the addressed KG. This is the case of Xser [17], gAnswer [19] or QuerioDali [14]. These factors limit portability.

For these reasons, *portability problem is not solved* (i.e., existing approaches working on one RDF dataset cannot be easily ported to a new dataset). Hence, up to now the goal of making any RDF dataset accessible via natural language has still not been achieved.

The observation that it is hard and cumbersome to build a QA system from scratch, leads to the idea of creating frameworks that allow the integration of existing techniques and services in a modular way. At least four frameworks tried to achieve this goal: QALL-ME [12], openQA [15], the Open Knowledge Base and Question-Answering (OKBQA) challenge[9] and Qanary [16,4,10]. While Qanary achieved to integrate a consistent number of tools, most of them only work for specific KGs and the portability problem is carried over from the integrated tools.

[8] c.f., `https://research.fb.com/downloads/babi/`
[9] `http://www.okbqa.org/`

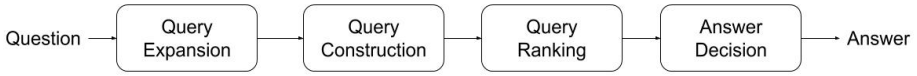

Fig. 1. QAnswer workflow

3 QAnswer Approach

In this section, we describe the workflow used by the QAnswer system to retrieve the answers for a given question formulated in natural language. For more details please refer to [5]. Figure 1 illustrates the QAnswer workflow, consisting of four steps: (1) Query Expansion, (2) Query Construction, (3) Query Ranking, and (4) Answer Decision, described in the following subsections. For the rest of this section, we use a running example "What planets belong to the solar system", queried over Wikidata[10] to describe our approach.

(1) Query Expansion: In this step, all possible n-grams from the textual question (with n taking values ranging from 1 to the number of words in the question) are mapped, if possible, to resources in the given KG. Hence, we intend to identify all possible interpretations of n-grams in a given question. Considering our example, the 1-gram sequences "solar" and "system" are mapped to the resources `Q29441547` (ID of "family name") and `Q58778` ("set of interacting or interdependent components"), among others; but the 2-gram "solar system" is mapped to `Q544` ("planetary system of the Sun"). The 1-gram "belong" is mapped to `Q4884518` (a band with that name), while the 2-gram "belong to" is mapped to the property `P361` ("part of"). Consequently, there are many possible mappings from the question to resources, but only a small subset of them is the correct one. In the following steps, all the possible combinations of mappings to resources are created, and then one of them is chosen in order to get the correct answer.

(2) Query Construction: In the second step, all possible SPARQL queries are generated from combinations of the resources identified in the previous step. To that end, we extract triple patterns from the KG by using the distance in the graph between the resources in it. Then each query is created by combining triple patterns that share a variable. In Figure 2, some example queries (i.e., candidates for a correct interpretation) for our running example are shown.[11]

(3) Query Ranking: In this step, the queries created in the previous step are ranked by a pre-trained machine learning component using a set of features. The goal is to rank the correct query in the top position. Among others, the following features are used for this purpose:

[10] `http://www.wikidata.org`

[11] We use the following RDF prefixes:
`PREFIX wdt: <http://www.wikidata.org/prop/direct/>`
`PREFIX wd: <http://www.wikidata.org/entity/>`

#	SPARQL query	Interpretation
1.	`SELECT DISTINCT ?s1 WHERE { ?s1 wdt:P398 wd:Q544 . }`	this query gives the astronomical bodies of the solar system
2.	`SELECT DISTINCT ?o1 where { wd:Q37532538 wdt:P282 ?o1 . }`	this outputs the writing system of the family name "belong"
3.	`?s1 wdt:31 wd:Q634 . ?s1 ?p1 wd:Q544 . }`	the searched query gives back the planets that have any relation with the Solar System
4.	`VALUES ?s0 { wde:Q544 } }`	the query just gives back the resource of the Solar System which would correspond to the question "Solar System?"

Fig. 2. Examples of queries, generated by QAnswer, with their corresponding interpretation of the question: "What planets belong to the solar system?".

- Number of words in the question string that are associated with resources in the SPARQL query.
- Similarity of the resource's label to the associated n-gram.

(4) Answer Decision: Finally, the query ranked in the top position from the previous step is analyzed. The goal is to decide if it is an appropriate answer to the question, i.e., to determine if it expresses the user's intention. For example, if the first ranked query would be Query 4 in Figure 2 (i.e., the query which just returns the information about "what the solar system is"), then the confidence should be low and no answer should be given. On the contrary, if the first ranked query is Query 1 in Figure 2, the confidence model should output that this is the right interpretation and provide the corresponding answer set.

This concludes the general description of the approach. For more details please see [9].

Advantages: The QAnswer approach departs from the traditional ways used in Question Answering domain (e.g., using linguistic features for Entity Recognition and Entity Disambiguation). However, it provides a number of advantages:

- **Robustness**: users are not limited to questions formulated using correct natural language. Our system supports keyword-based questions (e.g., "planets solar system"), or malformed questions (e.g., "planets that solar system belong"). The algorithm is robust enough to deal with all these scenarios [5][12].

[12] Note that spelling mistakes are treated in a separated process.

- **Multilingualism**: our approach can be applied to other languages without changes. In a previous work, it was shown that the algorithm works for English, German, French, Italian, Spanish, Portuguese, Arabic, and Chinese [9].
- **Multi-Knowledge-base**: our approach allows querying multiple knowledge bases at the same time [5].
- **Precision and Recall**: our approach has been tested on multiple benchmarks and can compete with most of the existing approaches [5].

In conclusion, QAnswer is designed to work with any RDF dataset and has several crucial features distinguishing it from QA systems designed for single datasets.

```
PREFIX vsw:    <http://vocabulary.semantic-web.at/cocktails>
PREFIX vswo:   <http://vocabulary.semantic-web.at/cocktail-ontology>
PREFIX sch:    <http://schema.org/>
PREFIX rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:   <http://www.w3.org/2000/01/rdf-schema#>

vsw:2d85fb1b    rdf:type        vswo:Cocktail ;
                rdfs:label      "Margarita"@en,"Upside Down Margarita"@en ;
                sch:description "The margarita is a Mexican ..."@en ;
                vswo:consistsOf vsw:1439e6c3, vsw:7dede323, vsw:88f5de3d .
vswo:Cocktail   rdfs:label      "Cocktail"@en .
vsw:1439e6c3    rdfs:label      "Cointreau"@en .
vsw:7dede323    rdfs:label      "Tequila"@en .
vsw:88f5de3d    rdfs:label      "Lime juice"@en .
vswo:consistsOf rdfs:label      "ingredients"@en, "consists of"@en,
                                "contains"@en, "made up"@en .
```

Question	**Answer**
Give me all cocktails.	Margarita
What is Margarita?	The margarita is a Mexican cocktail ...
Margarita cocktail?	The margarita is a Mexican cocktail ...
What are the ingredients of Margarita?	Cointreau, Tequila, Lime juice
What is Margarita made of?	Cointreau, Tequila, Lime juice
The ingredients of Margarita are what?	Cointreau, Tequila, Lime juice
ingredients Margarita?	Cointreau, Tequila, Lime juice
Give me cocktails containing tequila.	Margarita
Which cocktails have as ingredient Cointreau?	Margarita
cocktails containing tequila and cointreau	Margarita

Fig. 3. Upper part: A snippet of the Cocktail KG with information about a cocktail that, in English, is called "Margarita" or "Upside Down Margarita". It contains the facts that we are speaking about a cocktail, some names are described, a description and the ingredients are provided. **Lower Part**: Questions that can be answered using the snippet above. Note that many more questions can be answered, including the different variations of the questions above.

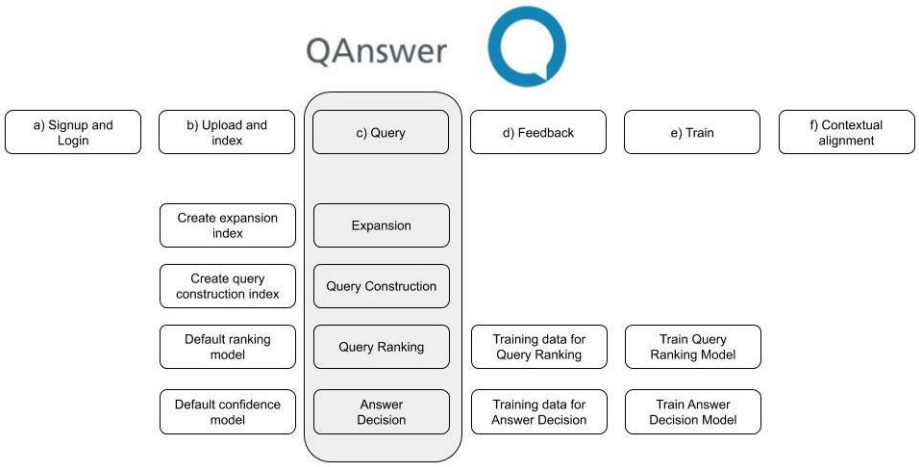

Fig. 4. Overview of QAnswer KG. The gray box shows the original QAnswer pipeline of Figure 1.

4 QAnswer KG: A RDF to QA approach

In this section, we describe the relation between an RDF dataset and the questions that can be answered by using QAnswer KG. In Section 4.2, we will describe the limitations of the proposed approach and will discuss how these limitations position our work with respect to the state-of-the-art.

In the following, another running example is used that is the small KG *cocktails*[13] providing information about cocktails. A snippet can be found in Figure 3 (above). The KG contains some cocktails including a short description, an image, and the ingredients. The triples contained in the snippet in Figure 3 (above) can be used to answer the questions of Figure 3 (below) as well as their variations. Note that there is a clear correspondence between the information encoded into the RDF dataset and the questions that can be answered using this information.

4.1 QAnswer KG

We are now going to describe the QAnswer KG process which encapsulates QAnswer for generating a QA system on top of an RDF dataset. The global architecture is depicted in Figure 4.

Initiation: The system reserves space for the new QA system and creates directories to allow the upload of new datasets (Figure 4a).

[13] The full KB is available at `https://qanswer.univ-st-etienne.fr/cocktails.nt`. It is published under a CC BY-NC license.

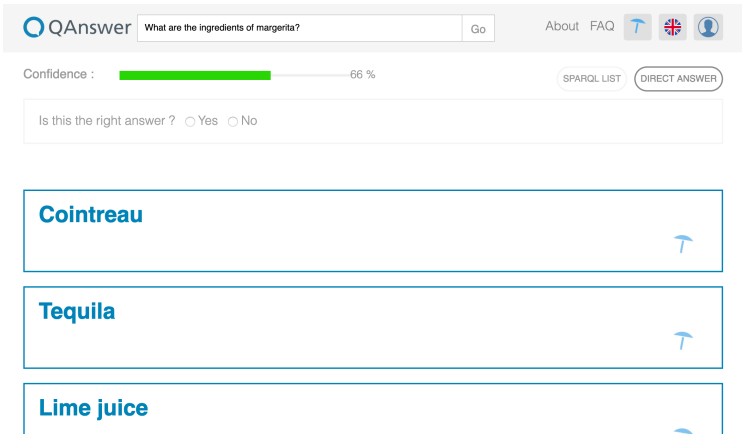

Fig. 5. Result for the question "What are the ingredients of Margarita?". Note that the system allows giving feedback by replying to the question: "Is this the right answer?" (options: Yes/No).

Indexing: After the dataset is uploaded (see Figure 4b), it is parsed and indexed. In particular, the indexes for query expansion step (1) and query construction step (2) are created. Both the query ranking step (3) and the answer decision step (4) are models built using machine learning. We already provide some default models for these steps. Moreover, we construct a SPARQL endpoint that is used to execute the generated SPARQL queries.

Query: Now, by means of its default algorithms provided initially, QAnswer KG can already answer user's natural language queries on the dataset. This step corresponds to the original QA pipeline of QAnswer (Figure 4c). For the "cocktails" dataset, QAnswer KG can, for example, answer to "What is a Margarita?" or "What are the ingredients of Margarita?" (see Figure 5).
The achieved initial and ad-hoc results may not be satisfying. The next section introduces the training of a machine learning model to adapt it to the particular dataset uploaded. Without this learning step, the initial default model can always be used.

Feedback and Training: Each time one asks a question, the QAnswer KG generates an interpretation and computes a confidence ratio. The system considers the answer to be correct if the confidence is higher or equal than 50% and wrong if it is lower than 50%. By using the feedback functionality, i.e., by getting the user's feedback to the question: "Is this the right answer?" (see Figure 5), QAnswer KG learns to choose the right interpretation and correctly compute the confidence.

For example, the default model, which is provided initially, responds "Cocktail" to the query "Give me all cocktails?". However, all generated interpretations can be shown to the user. By capturing the click on the right interpre-

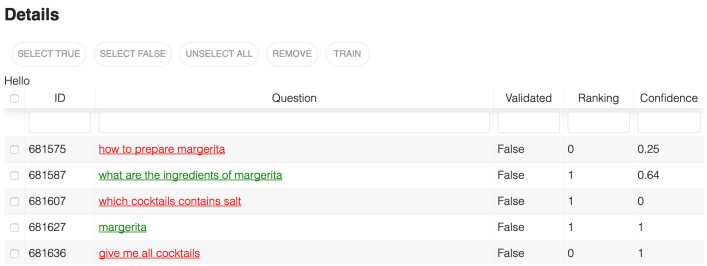

Fig. 6. Evaluation screen for the questions where feedback was provided. Red questions indicate that they will not be answered correctly according to the current model while questions marked green will be answered correctly. By clicking on the button "train", the model will be retrained and will learn from the given feedback.

tation, QAnswer KG learns to deliver the right answer (in this case: a long list of cocktails), but with low confidence. Using the feedback functionality, the system stores the given example for training (Figure 4d). After processing a set of questions, and by capturing the feedback, the system creates a training set. Such a training set for the cocktail dataset can be downloaded at `https://qanswer.univ-st-etienne.fr/questions_cocktail.nt`. For the questions on which feedback was given, QAnswer KG also provides an overview of how it performs on these questions (see Figure 6, "Training Evaluation"). At this stage, QAnswer KG is able to create an improved model that adapts to the specific training dataset (Figure 4e). This is done by retraining the underlying machine learning models. Note that this training process demands very light investment. This is because the system is only asking users to, optionally, provide feedback in the form of yes and no.

Contextual Information display: In the default setup, the system output is always provided by displaying the resulting `rdfs:label`. However, depending on the RDF datasets, there is potentially lot of contextual information that can be shown like descriptions, images, maps, and videos. QAnswer KG allows the display of these contextual pieces of information when the properties in the KG are specified (Figure 4f). Examples of properties in the case of the cocktail KG are:

- `http://www.w3.org/2004/02/skos/core#definition` gives a short description of the entity,
- `http://vocabulary.semantic-web.at/cocktail-ontology/image` indicates an image of the resource and
- `http://www.w3.org/2004/02/skos/core#exactMatch` indicates a DBpedia out-link.

There are two options to make QAnswer KG aware of such semantics: (1) One aligns the KG with the default properties of Wikidata described in the next

paragraph[14], and (2) One can specify the properties with the mapping interface, as shown in Figure 7. Figure 8 shows the displayed information with contextual information.

As for the option (1), by default, we align the KG with the following Wikidata properties:

- `http://schema.org/description`, to provide a description
- `http://www.wikidata.org/prop/direct/P18`, to provide an image. The object is expected to be a link to an online available image file.
- `http://www.wikidata.org/prop/direct/P625`, to visualize a geographic location. The object is expected to be a literal with datatype `http://www.opengis.net/ont/geosparql#wktLiteral` like `Point(12.482777777778 41.893055555556)^^http://www.opengis.net/ont/geosparql#wktLiteral`.
- External links can be expressed using the following properties:
  - `http://www.wikidata.org/prop/direct/P856` to show a link to the homepage
  - `http://www.wikidata.org/prop/direct/P1651` to show a YouTube link
  - `http://www.wikidata.org/prop/direct/P2037` to show a GitHub link
  - `http://www.wikidata.org/prop/direct/P2002` to show a Twitter link
  - `http://www.wikidata.org/prop/direct/P2013` to show a Facebook link
  - `http://www.wikidata.org/prop/direct/P2003` to show an Instagram link
  - `http://www.wikidata.org/prop/direct/P496` to show an ORCID link
  - `http://www.wikidata.org/prop/direct/P356` to show a DOI link

4.2 Limitations

In this section, we describe the current limitations of QAnswer KG and discuss how these limitations can be positioned with respect to the state of the art QA research.

Limitation 1: A URI will only be used to generate SPARQL queries if the question contains (up to stemming) the literal attached via one of the following properties:

- `http://www.w3.org/2000/01/rdf-schema#label`
- `http://purl.org/dc/elements/1.1/title`
- `https://schema.org/name`
- `http://www.w3.org/2004/02/skos/core#prefLabel`
- `http://www.w3.org/2004/02/skos/core#altLabel`

[14] The cocktail KG where the above URIs are substituted can be downloaded at `https://qanswer.univ-st-etienne.fr/cocktails_align.nt`.

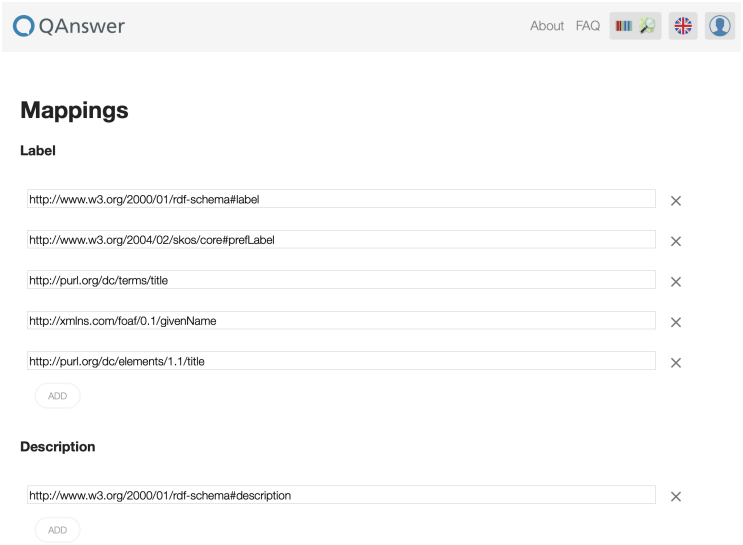

Fig. 7. This interface allows specifying properties that should be taken into consideration when displaying contextual information. For example, by adding a property "P" to the description section, QAnswer KG will use the information attached to "P" to render a description.

The intention is inspired by commonly used approaches to express the corresponding resource. While these properties are used by default, it is possible to specify custom properties.
Moreover, the language tag is important. In QAnswer KG the user has to select the language of the questions asked. If the literal has a language tag, the label will only be searched in questions where the corresponding language is selected. If no language tag is attached, the corresponding label will be searched in all questions independently of the selected language. For example, we assume that an RDF graph is given as shown in Figure 9. Given the listed triples, the URI `vsw:2d85fb1b` will only be used in a SPARQL query if the question either contains "Margarita" or "Upside Down Margarita". Moreover, "Margarita" will be used for any language while "Upside Down Margarita" only if English is selected as a language (due to the language tag `en`). The URI `vswo:consistsOf` will be used to construct SPARQL queries if the question contains "consistsOf", "contains", "made up", "ingredients" or other equivalent expressions after stemming. This is for example the case for the expression "contained" which, after stemming, is same as "contains".
In particular, note that with a graph similar to as follows, it will not be possible to answer any question since no labels are attached:

```
vsw:Margarita vswo:consistsOf vsw:Cointreau .
```

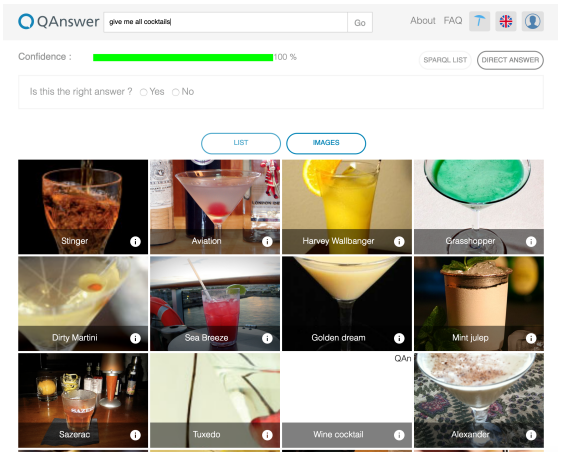

Fig. 8. Result set with contextual information.

```
vsw:2d85fb1b     rdfs:label  "Margarita" .
vswo:consistsOf  rdfs:label  "consists of"@en, "contains"@en,
                             "made up"@en, "ingredients"@en .
```

Fig. 9. Example graph (a subset of the triples provided in the cocktail KG).

Additionally, note that, for humans, even if the name of the URI is meaningful, according to RDF standard the above graph is equivalent to:

```
vsw:2d85fb1b  vswo:1234  vsw:1439e6c3 .
```

Hence, even for human users, it does not express the previous information.

Limitation 2: We assume that the RDF dataset does not use any form of reification. Recall that, RDF is perfectly suited to represent binary statements like "Margarita contains Cointreau" which can be represented as the triple (Margarita, contains, Cointreau). Reified statements are used when there is the need to speak about a binary statement like in: "Margarita contains 50 ml of Cointreau". In this case, a triple is not enough to represent this piece of information. The Semantic Web Community proposed a series of models to represent this type of information. For a full overview of the presented models, we refer to [13]. One of the models is *n-ary relations* (the reification model used by Wikidata), where the knowledge would be represented as:

```
vsw:Margarita  vswo:consistsOf_IN  _:b1 .
_:b1 vswo:consistsOf_OUT vsw:Cointreau .
_:b1 vswo:quantity "50 ml" .
```

Another model is RDF reification which was proposed during the standardization of RDF. The knowledge would be represented as:

```
vsw:Statement
    rdf:type       rdf:Statement ;
    rdf:subject    vsw:Margarita ;
    rdf:predicate  vswo:consistsOf ;
    rdf:object     vsw:Cointreau ;
    vswo:quantity  "50 ml" .
```

QAnswer KG was not designed to cope with such representations and it is not clear how it behaves when they are indexed in QAnswer KG. We consider this as a future challenge.
Querying KGs, which contain reified statements, is a poorly addressed research topic. Let's consider the three main QA benchmarks and the systems evaluated on them, namely SimpleQuestions, QALD, and WebQuestions. SimpleQuestions is based on a non-reified version of Freebase so this problem is not addressed in the benchmark. QALD is based on DBpedia which does not contain reified statements. Consequently, all systems evaluated over QALD also do not tackle this problem. Finally, WebQuestions considers full Freebase and therefore its reified structure. However, by reviewing the QA systems evaluated over WebQuestions, it can be seen that more than 60% of the systems ignore the reified structure by deleting the contextual information (like it is done in Bordes et al. [3]). The remaining approaches were only evaluated over Freebase and none among them was evaluated over KGs.

Limitation 3: The complexity of the SPARQL query that can be generated is limited. The queries can be of type `ASK`, `COUNT`, and `SELECT` and may contain up to 3 triple patterns.
Again let's compare this with respect to the state-of-the-art with three main QA benchmarks. All questions over SimpleQuestions are SELECT queries with one triple pattern. WebQuestions does not contain the answer queries, but only the labels of the answers. However, [1] achieved high accuracy by only matching to a non-reified or a reified statement which corresponds to SPARQL queries with a maximum of 3 triple patterns. Finally, the QALD benchmark contains some queries with aggregates or SPARQL operators like ORDER BY and FILTER. Anyways, our analysis shows that 66 % of the questions in QALD-9 can be answered using the syntax supported by QAnswer KG. Moreover, most of proposed approaches over QALD do not generate these kind of queries.

We above, we described the three main limitations of QAnswer KG and explained how our work can be positioned in the state-of-the-art after considering such limitations.

4.3 Experiment

To prove the portability of our approach, we let users test our system on three datasets.

- A **cocktails** dataset: the dataset used as a running example in the previous sections, i.e., a dataset containing cocktails with their ingredients and preparation.

- An **HR** dataset: the dataset contains information about employees of a company. The information includes their skills, the spoken languages, the languages they can program and their images.
- A **EU** dataset: i.e., a dataset containing information on the European Union about their member states, their capitals and ministries. This dataset is multilingual.

The users who set up the systems were knowledge engineers, i.e., persons who are familiar with knowledge representation, but not with question answering. The users checked the datasets beforehand to know which information they encode, i.e., which questions can be answered using it. The users generated some benchmarks for the datasets using the feedback functionality described in Section 4.1. The statistics of the three datasets and the statistics of the benchmarks are reported in Table 10.
In many cases, the users do not need to know the answer, but can make an educated guess about the correctness. This can be done by verifying the generated SPARQL query. For example, assume the user asks "What is the capital of Slovenia?", but he/she does not know the capital. The user can check the SPARQL query if it is "Slovenia capital ?o" then he/she can click on yes to provide feedback to express that the SPARQL query correctly reflects the question. It is assumed that the data and knowledge encoded in the KG are correct.
The efforts in re-training the system can be quantified with the number of questions asked by the users. These are reported in Figure 10. The number of questions for the EU dataset is higher than the others. But only 1/6th of the questions were formulated. The remaining 5/6th were automatically translated to 5 different languages to ensure that the system also works in other languages.
We report the F-Measure of the 3 benchmarks by using the default model and the trained model in Table 10. We can see that, before training, the default model generalizes quite well, while after training we get very good performances. This shows that it is possible to reuse the described QA system across different domains.

Dataset	**#Triples**	**#Properties**	**#Questions**	**F-Measure default**	**F-Measure train**
Cocktails	10.253	90	27	0.37	0.92
HR	4394	48	259	0.52	0.97
EU	4.835.856	992	844	0.70	0.90

Fig. 10. Statistics of the three considered datasets and their benchmark. Note that the benchmark over the EU dataset is multilingual.

4.4 Service and Demo

An online tutorial describing the process is available at `https://qanswer.univ-st-etienne.fr/docs/doc0`. The QAnswer KG approach was implemented

as a service and is available for public use. A demo is available at `https://www.qanswer.eu/qa`. It facilitates access to several datasets using the QAnswer KG technology. Many well-known and widely used datasets, such as DBpedia, Wikidata, DBLP, OpenStreetMap, MusicBrainz, and ORCID are provided. Thus, users can ask questions using natural language to retrieve information from these datasets. The demo supports the following languages: English, German, French, Italian, Spanish, Portuguese, Dutch, Chinese, Arabic, and Japanese. The quality of the created question answering system is inherited from the QAnswer approach (cf., [5] for details on the benchmark results).

5 Conclusion and Future Work

We have presented QAnswer KG, a novel approach that is able to generate on-demand QA systems for any RDF dataset. It addresses one of the major open challenges in the domain of QA over KGs, namely, portability.
The QAnswer KG approach is designed on top of the QAnswer technology that is in turn encapsulated inside the QAnswer KG. QAnswer provides major features, e.g., it is robust with respect to new questions from the user, allows multilingual question, and can be used to query multiple KGs at the same time. The QAnswer technology was extensively tested on a wide variety of benchmarks showing that it can compete with most of the state-of-the-art solutions.
Our approach enabled non-expert users to create QA systems on top of new RDF datasets. There is little to no knowledge about QA required to establish a system by a user as shown in the demo. Therefore, QAnswer KG achieves portability for RDF-driven QA system. We believe that QAnswer KG represents an important contribution for the Semantic Web Community since it will enable data owners to expose their datasets directly to end-users, and therefore make the Semantic Web more useful and popular.
In the future, we plan to extend the functionality of the QAnswer KG service by integrating additional services: (A) SPARQLtoUser (cf., [6]), a service capable of transforming a SPARQL query into a user understandable representation, (B) SummaServer [11], a service that selects between all triples associated to an RDF entity, the most important ones, (C) a service to allow users to disambiguate between different entities, as described in [7]. Note that these services are already used when querying some KGs like Wikidata, DBpedia and DBLP, but they are not sufficiently generalized to work over any KG.

Note: There is a patent pending for the presented approach. It was filed on January 18th, 2018 at the EPO (number EP18305035.0).
Acknowledgment: We want to thank Semantic Web Company to let us use the cocktails dataset.

References

1. Bast, H., Haussmann, E.: More accurate question answering on freebase. In: Proceedings of the 24th ACM International on Conference on Information and Knowl-

edge Management. ACM (2015)
2. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic Parsing on Freebase from Question-Answer Pairs. In: EMNLP (2013)
3. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)
4. Both, A., Diefenbach, D., Singh, K., Shekarpour, S., Cherix, D., Lange, C.: Qanary a methodology for vocabulary-driven open question answering systems. In: ESWC 2016 (2016)
5. Diefenbach, D., Both, A., Singh, K.D., Maret, P.: Towards a question answering system over the semantic web. Semantic Web Journal (2019)
6. Diefenbach, D., Dridi, Y., Singh, K.D., Maret, P.: Sparqltouser: Did the question answering system understand me? In: Joint Proceedings of BLINK2017: 2nd International Workshop on Benchmarking Linked Data and NLIWoD3: Natural Language Interfaces for the Web of Data co-located with 16th ISWC. (2017), http://ceur-ws.org/Vol-1932/paper-01.pdf
7. Diefenbach, D., Hormozi, N., Amjad, S., Both, A.: Introducing feedback in Qanary: How users can interact with QA systems. In: ESWC P&D (2017)
8. Diefenbach, D., Lopez, V., Singh, K., Pierre, M.: Core Techniques of Question Answering Systems over Knowledge Bases: a Survey. Knowledge and Information systems (2017)
9. Diefenbach, D., Migliatti, P.H., Qawasmeh, O., Lully, V., Singh, K., Maret, P.: QAnswer: A question answering prototype bridging the gap between a considerable part of the LOD cloud and end-users. In: The World Wide Web Conference. pp. 3507–3510. ACM (2019)
10. 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)
11. Diefenbach, D., Thalhammer, A.: Pagerank and generic entity summarization for RDF knowledge bases. In: ESWC. pp. 145–160. Springer (2018)
12. Ferrández, Ó., Spurk, C., Kouylekov, M., al.: The QALL-ME framework: A specifiable-domain multilingual Question Answering architecture. Journal of Web Semantics (2011)
13. Giménez-García, J.M., Zimmermann, A., Maret, P.: Ndfluents: An ontology for annotated statements with inference preservation. In: The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Proceedings, Part I. pp. 638–654 (2017). https://doi.org/10.1007/978-3-319-58068-5_39
14. López, V., Tommasi, P., Kotoulas, S., Wu, J.: Queriodali: Question answering over dynamic and linked knowledge graphs. In: ISWC (2016)
15. Marx, E., Usbeck, R., Ngonga Ngomo, A., Höffner, K., Lehmann, J., Auer, S.: Towards an open question answering architecture. In: SEMANTICS (2014)
16. Singh, K., Both, A., Diefenbach, D., Shekarpour, S.: Towards a message-driven vocabulary for promoting the interoperability of question answering systems. In: ICSC 2016 (2016)
17. Xu, K., Feng, Y., Zhao, D.: Xser@ QALD-4: Answering Natural Language Questions via Phrasal Semantic Parsing (2014)
18. Zhang, Y., Liu, K., He, S., Ji, G., Liu, Z., Wu, H., Zhao, J.: Question answering over knowledge base with neural attention combining global knowledge information. arXiv preprint arXiv:1606.00979 (2016)
19. Zou, L., Huang, R., Wang, H., Yu, J.X., He, W., Zhao, D.: Natural language question answering over RDF: a graph data driven approach. In: Proc. of the 2014 ACM SIGMOD international conference on Management of data. ACM (2014)