HOBBIT
Holistic Benchmarking of Big Linked Data

Collaborative Project

HOBBIT – Holistic Benchmarking of Big Linked Data

Project Number: 688227 Start Date of Project: 01/12/2015 Duration: 36 months

Deliverable 6.1.2 Second Version of the Question Answering Benchmark

Dissemination Level	Public
Due Date of Deliverable	Month 18, 31/05/2018
Actual Submission Date	Month 18, 30/05/2018
Work Package	WP 6
Task	T 6.1
Type	Other
Approval Status	Final
Version	1.0
Number of Pages	21
Filename	D6.1.2_Second_Version_of_QA_Benchmark.pdf

Abstract: This report describes the results of the tasks related to the second version of the question answering benchmark. It includes a description of the work that has been carried out as well as a detailed description of the different available experiments.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688227.

History

Version	Date	Reason	Revised by
0.0	01/05/2018	First Draft	Giulio Napolitano
0.1	09/05/2018	Internal review comments	Mohnish Dubey
0.2	22/05/2018	Peer review comments	Irini Fundulaki
1.0	25/05/2018	Approval	Jens Lehmann

Author List

Organisation	Name	Contact Information
Fraunhofer IAIS	Giulio Napolitano	Giulio.napolitano@iais.fraunhofer.de

Executive Summary

This report describes the task 6.1 activities of the HOBBIT project for the second version of the Question Answering Benchmark. The main content of this document is a detailed description of the activities from 1st of June 2017 until the end of May 2018. The report includes information about the work that has been carried out by Fraunhofer IAIS and other participating partners. It also describes in detail the three experiment tasks of the QA Benchmark and walks through an example experiment.

Table of Contents

List of Figures

1. Introduction

Since the compilation of the first version of the present document, one year ago, the pace and amount of research on question answering over large-scale RDF data have not decreased. After the first excitement, regular interactions with consumer question answering systems such as Amazon Alexa or Google Assistant have become more and more natural. As consumers' expectations around the capabilities of systems able to answer questions formulated in natural language (QA systems) keep growing, so is the availability of such systems in various settings, devices and languages. Use cases are proliferating well beyond the trivial questions about the current traffic and weather or about the date of birth of a famous person. Making use of contextual information and clues, systems have been developed in a range of settings, e.g. to assist in disaster scenarios or in childcare tasks.

Strictly speaking, the general goal of a QA system is to compute the correct answer to a factoid natural language query, given a number of structured datasets which constitute its "knowledge base". Available systems for consumers, such as those mentioned above, comprise complex ecosystems of APIs and toolkits, which make possible for developers to produce QA or dialogue systems which may retrieve information from a number of knowledge bases. As such, they may embed their QA capabilities within a more versatile environment with dialogic and small talk capabilities. Our benchmark do not test these capabilities. Rather, it concentrates on some of the challenges (choke points) that QA systems may encounter in real settings in answering individual, factoid questions regardless of context or dialogue history.

These challenges are in constant evolution as the systems and methodologies addressing them are. The need for comparable and reliable rating of QA systems then arises.

1.1 Question Answering over Linked Data

A system able to answer natural language questions, on the basis of structured datasets, can adopt a number of strategies. Most typically, however, such a system will attempt to analyse the question, translate it into an equivalent form in a query language, which is finally run on the dataset (knowledge base) to retrieve the answer. Figure 1 summarises the basic steps of the process.

In general, the underlying datasets are organized in factual triples including a subject, predicate and object of a so-called statement according to the RDF standard. A triple consist of an entity as subject which the factual statement is about, a predicate forming the attribute of the entity and an object as the value for the attribute. SPARQL can be used as a query language to gather answers from the dataset. Therefore, the main task of a QA system is to translate a natural language question into an appropriate SPARQL query which delivers the correct answer.

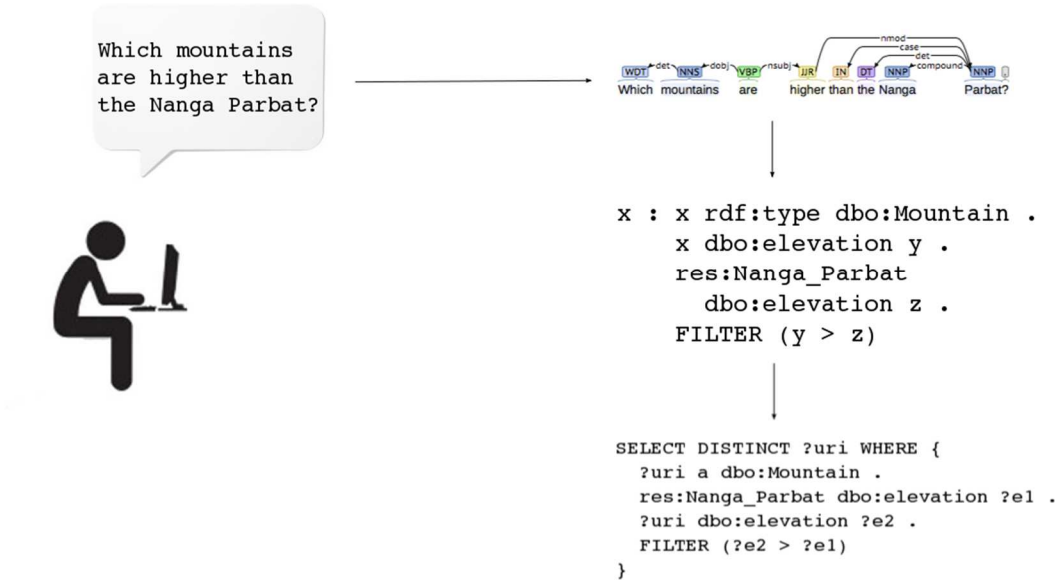

Figure 1 - From question to query

In HOBBIT, the knowledge base we use for the QA benchmarks is DBpedia. Factual information retrieved from Wikipedia is organised in DBpedia in the form of a huge knowledge graph, a set of labelled nodes (entities) and directed arcs (relations) connecting them, similarly to the tiny fragment shown in Figure 2:

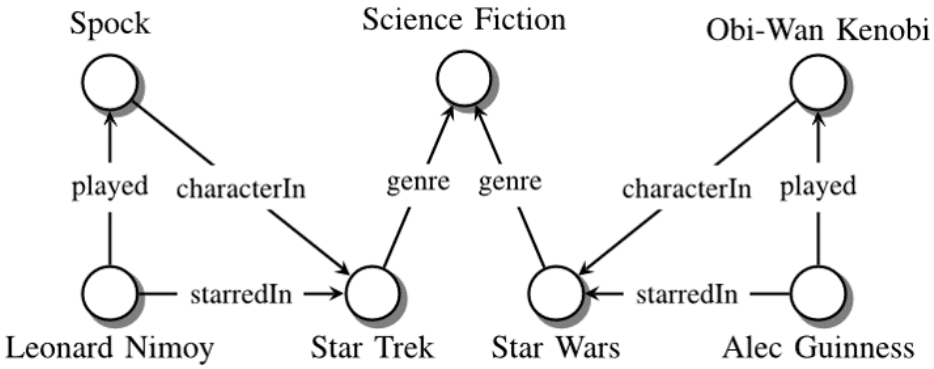

Figure 2 - A simple graph of facts

Usually, relations between entities are represented as triples of subject-predicate-object, analogous to (Leonard_Nimoy, played, Spock). DBpedia is, generally speaking, the connected graph defined by a very large set of such triples, which describe "facts" stated in Wikipedia. Such a graph can be queried by means of the SPARQL query language, an example of which is at the bottom of Figure 1.

Previous work on benchmarking question answering systems include Yahoo! Semantic Search[1], GERBIL QA[2] and QALD[3]. The benchmarking of a number of QA systems typically consists in assigning to each system one or more performance scores following their computation of answers to the same set of natural language questions. Thus, the QA benchmark makes it possible to rank question answering systems based on their performance and to formulate statements about their excellence and quality (or their lack thereof). Our goal within HOBBIT project has been to provide a platform for benchmarking QA systems on a number of tasks and related key performance indicators (KPIs). The benchmarking of QA systems results in a score that, typically, corresponds to the fraction of correct answers from the participating system on the given natural language questions.

2. Choke Points and Goals

Tasks and KPIs are specific to the challenge(s) that a system intends to address. For instance, if a QA system aims at answering complex queries, precision and recall are relevant indicators; for another system that aims at answering a large number of simple queries in a short time, some responsiveness metric will also be relevant. The HOBBIT Question Answering Benchmark focuses on:

- Multilinguality (Multilingual QA task), that is the ability of a system to answer the same questions formulated in multiple languages.
- Source heterogeneity (Hybrid QA task), in our case the ability to address questions that may only be answered by resourcing to the combination of structured datasets and information expressed in free text.
- Scalability (Large-scale QA task), here the performance of a system in answering questions at an increasing pace.

2.1 Multilingual QA Task

As the interest in automatic question answering applications widens beyond academia and English-speaking corporate domains, it becomes increasingly compelling to provide solutions in languages other than English. We have created two subtasks within this task. With the first one, already in place in Version 1 of the benchmark, systems can be evaluated on a set of 50 hand-crafted questions, each available in eight languages including English, German, French, Spanish, Italian, Dutch, Romanian and Farsi. Each question is also associated with a SPARQL query for the English version of DBpedia.

The second, new benchmark comprises 200 semi-automatically generated questions each available in English, German and Italian. Each question is also associated with three SPARQL queries, capable to retrieve the correct answer from the English, German and Italian versions of DBpedia respectively.

[1] http://krisztianbalog.com/2011-03-03/yahoo-semantic-search-challenge/ retrieved on 02/05/2017

[2] http://gerbil-qa.aksw.org/gerbil/config/ retrieved on 02/05/2017

[3] http://qald.sebastianwalter.org/ retrieved on 02/05/2017

2.2 Hybrid QA Task

Although well curated, comprehensively structured knowledge bases represent the ideal source of information for an automated answering system. In reality, these sources only constitute a negligible portion of potentially usable information which is (so far) still expressed in the format preferred by human beings: free text. This will remain the case for the foreseeable future, hence versatile systems should also be able to cope with unstructured information. The Hybrid QA task enables the benchmarking of such systems by challenging them with hybrid questions, that is questions requiring the retrieval of information from both a subset of a structured source and an associated free-text document. In our case, these correspond to a set of DBpedia triples and a textual abstract, respectively.

2.3 Large-Scale QA Task

Finally, as big data becomes bigger and faster, so does the requirement for QA systems to scale up in terms of volume and velocity. For the first version of the large-scale QA task, we had automatically generated 1.6M questions (and their associated SPARQL queries) from an initial 150 question template set, by substituting entities in this small set with other entities of the same type. The questions were sent to the system to be benchmarked at growing frequency, putting the system under increasing stress.

For the second version of this benchmark, we have increased the complexity of the questions, that is the corresponding SPARQL queries are not limited to one triple in the WHERE clause but also include patterns of two or three triples. The number of questions is now just below 2,000 and we introduced some spelling mistakes and anomalies, as a way to simulate a noisy real-world scenario in which questions may be served to the system imperfectly as a result, for instance, of speech recognition failures or typing errors.

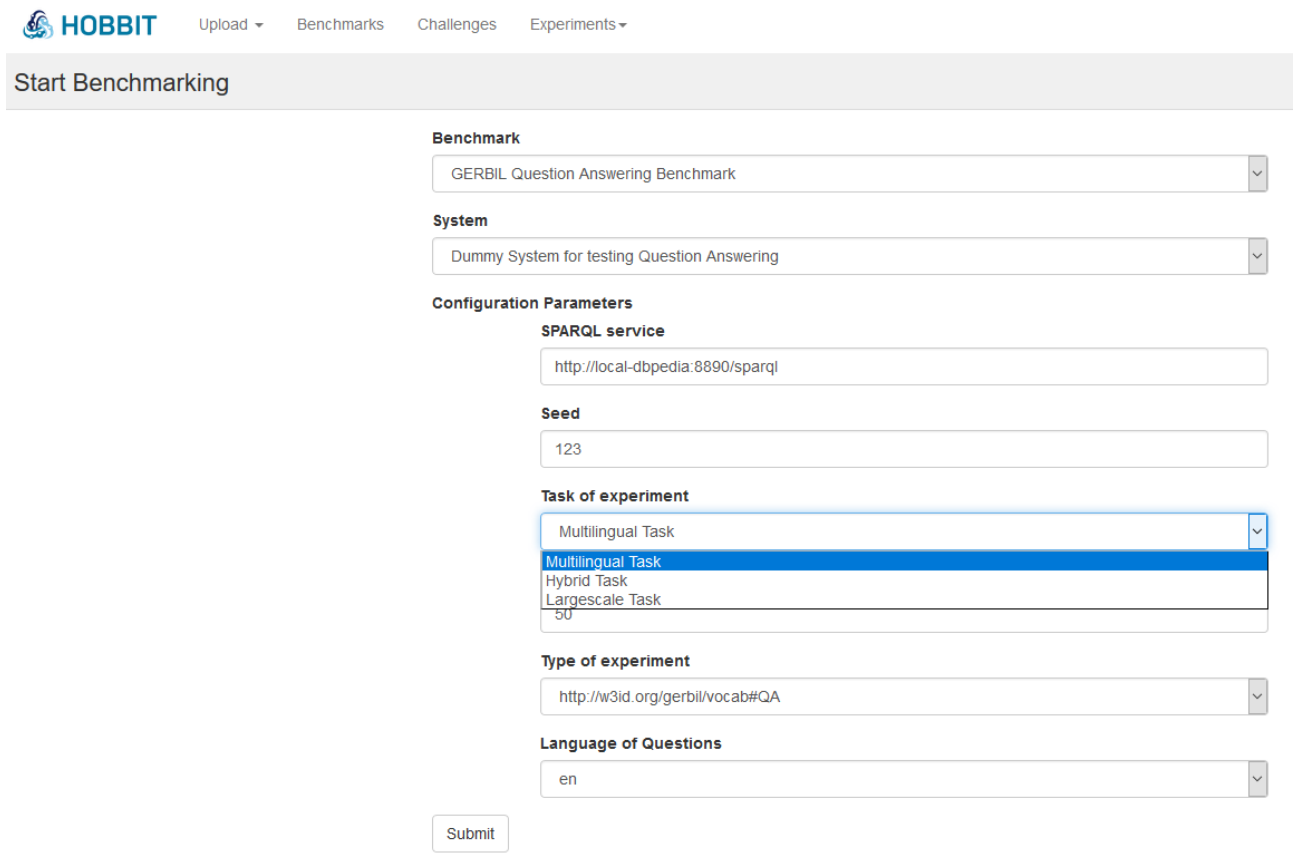

Figure 3 - Selection of experiment task in the Question Answering Benchmark

The evaluation platform for all benchmarks were built upon the open source GERBIL QA benchmarking platform. The goals of the workpackage did not include development, evaluation or benchmarking of user interfaces, neither user surveys.

3. Work Performed and Results

The overall goal for the second version of the Question Answering Benchmark has been to consolidate and improve the tasks, the underlying code modules and the datasets from Version 1. The benchmark is fully integrated in the HOBBIT platform and the following is a brief overview of its architecture.

3.1 Benchmark Architecture within HOBBIT

In Figure 4the outline architecture of the QA benchmark components (light orange boxes) are shown in relation to the HOBBIT platform components (light blue boxes) and the system being benchmarked (grey box). In brief, the Benchmark Controller initiates the deployment of other components and controls the whole benchmarking process, using HOBBIT platform-compatible protocols and communicating with the Command Queue of the platform. The Task Generator consumes the dataset file from the Data Generator, by extracting the natural language questions and, if relevant to the task, related information and sends them to the Benchmarked System. At the same time, the corresponding SPARQL queries are also run against the knowledge base and the results are used as the gold standard. The gold standard answers and the corresponding Benchmarked System answers are finally sent to the Evaluation Storage, on the basis of which the Evaluation Module will calculate the KPIs of the Benchmarked System.

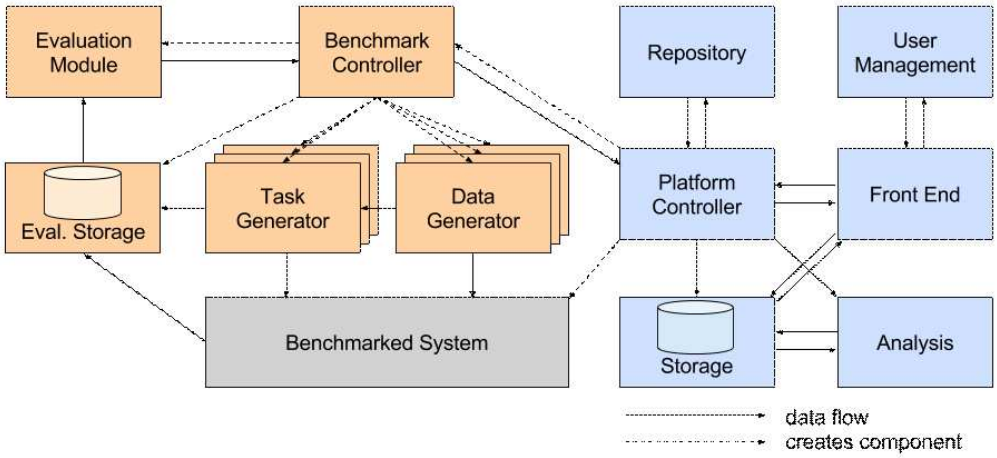

Figure 4 - Question Answering Benchmark components (light orange) in their interaction with HOBBIT platform components (light blue) and the Benchmarked System (grey).

As mentioned above, the question answering benchmark is divided into three tasks. Each task tackles a different number of choke points. As Key Performance Indicators, we set the usual suspects: precision, recall, F1-score and, in the large-scale task, we also consider the ability of the system to avoid failure upon an ever increasing volume of questions. In the following, we provide a more detailed description of the three tasks we compiled.

As mentioned above, all answers of all tasks are generated by the task generator at runtime. We set up private local SPARQL endpoints for all sources so that, on the one hand, we protect the public endpoint from massive querying and, on the other hand, we match the systems‘ answers against the correct gold standard answers retrieved at the same time from the same static, underlying database, while the public live versions might change from time to time.

Finally, following code review and dataset reorganisation we optimised the software modules of the benchmark, which is now much more efficient and error-free.

3.2 Task 1: Multilingual Question Answering

3.2.1 Task 1a: English DBpedia

The Multilingual Question Answering on the English DBpedia task consists of 50 natural language questions with each query available in eight different languages. Available languages include English, German, French, Spanish, Italian, Dutch, Romanian, and Farsi. Additionally, besides the complete question there are keywords available for the systems.

Example:

en: Are penguins endangered?	penguins, endangered
de: Sind Pinguine vom Aussterben bedroht?	Pinguine, Aussterben, Bedrohung
it: I pinguini sono una specie in pericolo?	pinguini, specie, pericolo

Together with question and keywords, the corresponding DBpedia SPARQL query and the correct answer are available to the evaluation component. The DBpedia query for the sample question above would be

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/
ASK WHERE {
          ?uri dbo:family dbr:Penguin .
          ?uri dbo:conservationStatus “EN“ .
}
```

This query executed on the DBpedia public SPARQL endpoint[4] returns the answer *true* .

This task is a straightforward question answering challenge. Each QA system is able to choose a language in which it performs the natural language analysis and the SPARQL query conversion best. Possible analysis problems in one language can be compensated by making additional sense from a different language.

Tackled choke points of this task include all of the mentioned choke points in chapter 2.

3.2.2 Task 1b: Localised DBpedias

This task consists of 200 natural language questions each available in English, German and Italian. Here we followed the same approach as LC-QuAD:[5] instead of generating SPARQL queries from natural

[4] http://dbpedia.org/sparql/ retrieved on 02/05/2017

[5] Trivedi, Priyansh, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. "Lc-quad: A corpus for complex question answering over knowledge graphs." In International Semantic Web Conference, pp. 210-218. Springer, Cham, 2017.

language questions, we first composed groups of three equivalent (=corresponding to the same question) SPARQL queries for the three DBpedias in English, German and Italian. Then, from the SPARQL queries, we generated the corresponding natural language questions for each language.

Initially, a whitelist of predicates from the DBpedia ontology that are in use by all three language localisations was generated. Then, in each localisation, a set of RDF triples was generated from the DBpedia graphs. Each triple in each set contained one of those predicates, while subject and object were entities which were linked by the *sameAs* relation to the corresponding entities (for subject and object respectively) in the RDF triples in the other languages. In other words, for each RDF triple in one language, a corresponding triple also exists in the DBpedia graphs in the other two languages. These initial sets (one per language) of single triples were then expanded, by using an analogous mechanism, to sets of two and three triples linked by common entities.

By means of the same query templates used by LC-QuAD, each single or multiple triple in all sets was converted into a SPARQL query. As a result, three types of SPARQL queries were obtained, of increasing complexity, that is queries containing one, two or three triples. These are illustrated below, together with their templates and following these notation conventions:

- `e`: entity (KB resource)
- `o`: relation (ontology predicate)
- `e1 o1 e2`: entity 1, relation 1, entity 2. `e1` ≠ `e2`
- `e o e`: the meaning of this triplet format is {Subject Relation/Predicate Object}

Type 1: One triple queries

There are two cases for the one triple queries (i.e. containing a single triple in the WHERE clause) corresponding to two templates:

1) `e1 o ?ans`: where the question is asked about the subject and the relation. The answer is the object.

The following example shows a question, its original SPARQL query (all the entities are there) and the Target query, where the answer entity is missing and the question should reflect the meaning of this query.

- "NL" is the natural language question
- "SPARQL" is the target query that the question should capture with its meaning
- "Triples" is the full query or predicates with all the entities and relations. It is there provided to help formulating a question the better captures the SPARQL query

```
{
    "NL": "Who is the founder of the New Statesman Newspaper?",
    "SPARQL":"<http://dbpedia.org/resource/New_Statesman>
<http://dbpedia.org/ontology/founder> ?ans",
    "Triples":"<http://dbpedia.org/resource/New_Statesman>
<http://dbpedia.org/ontology/founder>
<http://dbpedia.org/resource/Beatrice_Webb>"
}
```

2) `?ans o ?e2`: where the question is asked about the subject and the relation. The answer is the subject.

```
{
```

```
          "NL": "What did the English actor Michael Palin star in?",
          "SPARQL":"?ans                    <http://dbpedia.org/ontology/starring>
<http://dbpedia.org/resource/Michael_Palin>",
          "Triples":  "<http://dbpedia.org/resource/Monty_Python's_Flying_Circus>
<http://dbpedia.org/ontology/starring>
<http://dbpedia.org/resource/Michael_Palin>"
      }
```


Type 2: Two triple queries

There are four cases for the two triple queries:

1) `?x o1 e2 . ?x o2 ?ans`

In this case, ?x is an unknown variable and ?ans is the target answer. The intuition behind this case is that the question is asked about something X and X has some relation with Y, such as: *what is the name of Y where Y is the son of X?*


```
        {
          "NL": "What is the name of the coach of an Italian professional
basketball team?",
          "SPARQL": "?x <http://dbpedia.org/ontology/league>
<http://dbpedia.org/resource/Lega_Basket_Serie_A> . ?x
<http://dbpedia.org/ontology/coach> ?ans",
          "Triples": "<http://dbpedia.org/resource/Pallacanestro_Olimpia_Milano>
<http://dbpedia.org/ontology/league>
<http://dbpedia.org/resource/Lega_Basket_Serie_A>.
<http://dbpedia.org/resource/Pallacanestro_Olimpia_Milano>
<http://dbpedia.org/ontology/coach>
<http://dbpedia.org/resource/Jasmin_Repe\u0161a>"
      }
```


2) `e1 o1 ?x . ?x o2 ?ans`


```
        {
          "NL": "What is the leader party of a district in the city of
Liedersdorf?",
          "SPARQL": "<http://dbpedia.org/resource/Liedersdorf>
<http://dbpedia.org/ontology/city> ?x . ?x
<http://dbpedia.org/ontology/leaderParty> ?ans",
          "Triples": "<http://dbpedia.org/resource/Liedersdorf>
<http://dbpedia.org/ontology/city> <http://dbpedia.org/resource/Allstedt> .
<http://dbpedia.org/resource/Allstedt>
<http://dbpedia.org/ontology/leaderParty>
<http://dbpedia.org/resource/Christian_Democratic_Union_of_Germany>"
      }
```


3) `?x o1 e2 . ?ans o2 ?x`[6]

4) `e1 o1 ?x . ?ans o2 ?x`

[6] For brevity, the examples for the remaining cases are omitted.

Type 3: Three triples

There are also four cases for the three-triple queries, i.e. queries with three triples in the WHERE clause:

1) `e1 o1 ?x . ?ans o2 e1 . ?ans o3 ?x`
2) `e1 o1 ?x . ?ans o2 e1 . ?ans o3 ?x`
3) `e1 o1 ?x . e1 o2 ?ans . ?x o3 ?ans`
4) `e1 o1 ?x . ?x o2 ?ans . ?ans o3 e1`

After the generation of the SPARQL queries, we manually created the corresponding natural language question in English by inspection of each query ("NL" in the example above). A subsequent review was employed to filter out those queries that resulted in unnatural or too convoluted natural language questions. Then, the English questions were automatically translated using the publicly available Google Translate service[7] and each translation was manually inspected and, where required, corrected. The following is an example of the resulting question items constituting the final dataset: three SPARQL queries for the English, German and Italian DBpedias respectively and the corresponding three natural language questions.


```
        {
            "id": 1,
            "onlydbo": true,
            "query": [
                {
                    "language": "en",
                    "sparql": "SELECT DISTINCT ?ans Where
{<http://dbpedia.org/resource/Ultima_VI:_The_False_Prophet>
<http://dbpedia.org/ontology/developer> ?ans}"
                },
                {
                    "language": "de",
                    "sparql": "SELECT DISTINCT ?ans Where
{<http://de.dbpedia.org/resource/Ultima_VI:_The_False_Prophet>
<http://dbpedia.org/ontology/developer> ?ans}"
                },
                {
                    "language": "it",
                    "sparql": "SELECT DISTINCT ?ans Where
{<http://it.dbpedia.org/resource/Ultima_VI>
<http://dbpedia.org/ontology/developer> ?ans}"
                }
```


[7] https://translate.google.com

```
            ],
            "question": [
                {
                    "language": "en",
                    "string": "Who designed Ultima VI: The False Prophet ?"
                },
                {
                    "language": "de",
                    "string": "Wer hat Ultima VI entworfen: Der falsche
Prophet?\n"
                },
                {
                    "language": "it",
                    "string": "Chi ha progettato Ultima VI: The False
Prophet?\n"
                }
            ]
    }
```

3.3 Task 2: Hybrid Question Answering

The Hybrid Question Answering task is different from the straightforward task 1 in such a way that the QA system not only needs to retrieve one piece of information from the database but also a second piece of information that is only available in the textual abstract and not in the triplified data. Only the combination of both the structured and the unstructured information leads to the correct answer.

Example:

Give me the names of all of the people who signed the American Declaration of Independence.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?uri WHERE {
          ?uri rdf:type dbo:person .
          ?uri text:\"signed\" .
            {
                    ?uri text:\"American Declaration of Independence\" .
            } UNION {
                    ?uri text:\"Declaration of Independence\" .
            }
}
```

The fact that someone is a person is represented in the triplified data and can be retrieved by "?uri rdf:type dbo:person ." However, the fact that a person signed the American Declaration of Independence is not represented in the data but mentioned in the textual abstract of the resource. Retrieving the words "*signed*" and "*Declaration of Independence*" or "*American Declaration of Independence*" in the abstracts of all resources that also have the class "*person*" leads to the correct answer:

<http://dbpedia.org/resource/John_Hancock>
<http://dbpedia.org/resource/John_Adams>
<http://dbpedia.org/resource/Benjamin_Franklin>
...
<http://dbpedia.org/resource/Thomas_Jefferson>
<http://dbpedia.org/resource/Lyman_Hall>
<http://dbpedia.org/resource/George_Walton>

In this task there are no keywords provided along with the questions. Tackled choke points include all of the mentioned choke points and the challenging fact that part of the information must be extracted from natural language text.

3.4 Task 3: Large-Scale Question Answering

The Large-scale Question Answering task adds another dimension to the evaluation. Systems are not only scored according to their correct answers but also in their capability to cope with increasing demand for answers without failure. In addition to precision, recall and F1-score, we introduced a new KPI for this task, the "response power", which is the harmonic mean of three measures: precision, recall and the ratio between processed questions (an empty answer is considered as processed, a missing answer is considered as unprocessed) and total number of questions sent to the system.

The new dataset was derived from the recently developed LC-QuAD dataset.[8] The original LC-QuAD dataset contains 5,000 questions of variable complexity, compiled on the basis of query templates which are instantiated by seed entities from DBpedia into normalised natural question structures. These structures are then transformed into natural language questions by native speakers.

Our dataset was created by manually paraphrasing 2200 questions from the LC-QuAD dataset. The paraphrased questions where ordered by their Vector Extrema Cosine Similarity score[9] to the original questions and only the first 1830 questions were retained. As in the original source questions, we intentionally left in typos and grammar mistakes as a way to reproduce a noisy real-world scenario, in which questions may be served to the system imperfectly as a result, for instance, of speech recognition failures or typing errors. A sample dataset entry is shown below:

```
{
            "hybrid": "false",
            "question": [
                  {
                        "string": "Which comic characters are painted by
                                    Bill Finger?",
```

[8] http://lc-quad.sda.tech/

[9] Sharma, Shikhar, Layla El Asri, Hannes Schulz, and Jeremie Zumer. "Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation." arXiv preprint arXiv:1706.09799 (2017).

```
                        "language": "en"
                }
        ],
        "onlydbo": true,
        "query": {
                "sparql": "SELECT DISTINCT ?uri
                                WHERE
                                {?uri
                        <http://dbpedia.org/ontology/creator>
                        <http://dbpedia.org/resource/Bill_Finger>  .
                        ?uri <http://www.w3.org/1999/02/22-rdf-
                        syntax-ns#type>
                        <http://dbpedia.org/ontology/ComicsCharacter
                        >}"
        },
        "aggregation": false,
        "_id": "f0a9f1ca14764095ae089b152e0e7f12",
        "id": 0
}
```


The benchmark sends one question at the start, two more questions after one minute and continues to send *n+1* new questions after *n* minutes. One minute after the last set of questions is dispatched, the benchmarks closes and the evaluation is generated as explained above. The 1830 questions in the dataset allow the running of the benchmark for one hour.

4. Example Experiments

In this chapter, we walk through the main system benchmarking steps with the HOBBIT Question Answering Benchmark.

We used a number of Question Answering test systems to run our experiments. One simple system, illustrated in the following screenshots, always returned correct answers. We also used a system that, for each questions, randomly returns an answer that is either *empty*, *wrong* or *correct* or simply does not return an answer. Finally, as system which returns a fixed proportion of empty, wrong, correct or no answers was also used to test the new KPI (Response Power) introduced for the Large Scale task. The following Figure 5 shows the summary of the settings before the experiment is executed.

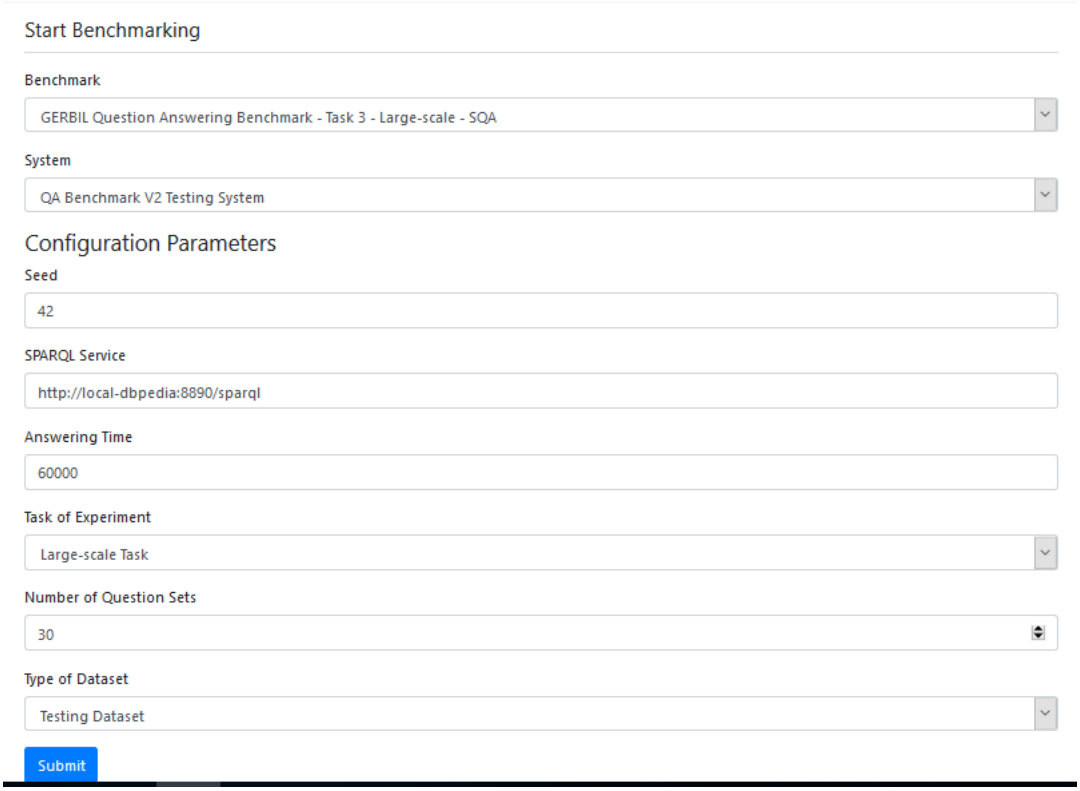

Figure 5 - Large Scale experiment: summary of configuration parameters before experiment execution

The system was set for the Large Scale QA task with an amount of 30 question sets, which in this task means the benchmark will send 30 packets of questions, increasing in size by one question each time. The Answering Time of 60000 milliseconds indicates that the packets will be sent to the Benchmarked System every minute. Once the experiment has been started by the user it will line up in the HOBBIT experiment queue and eventually be executed when previous experiments are completed (Figure 6).

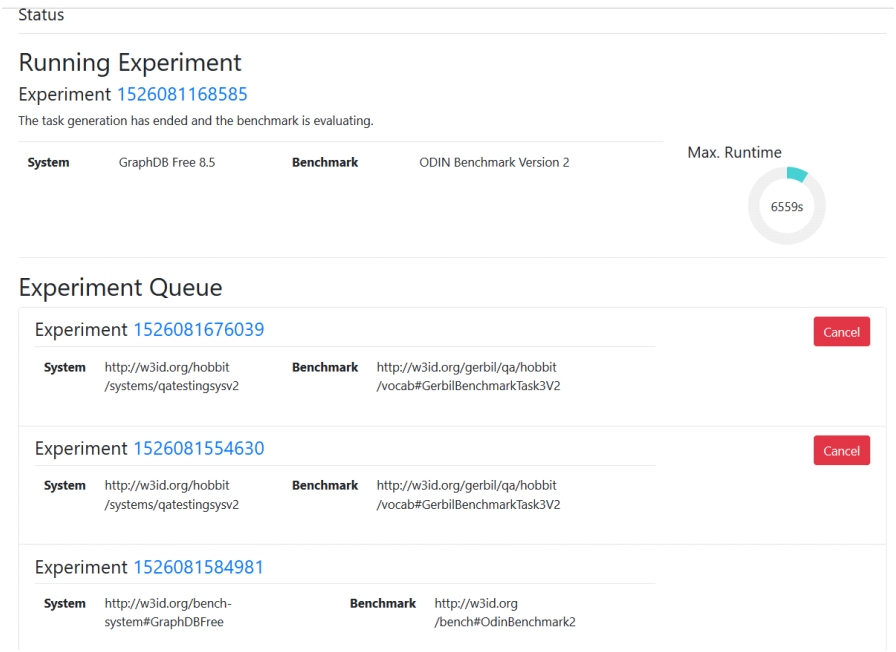

Figure 6 - Running Large Scale QA experiments: experiment queue.

Once the experiment has finished, the results are displayed (Figure 7).

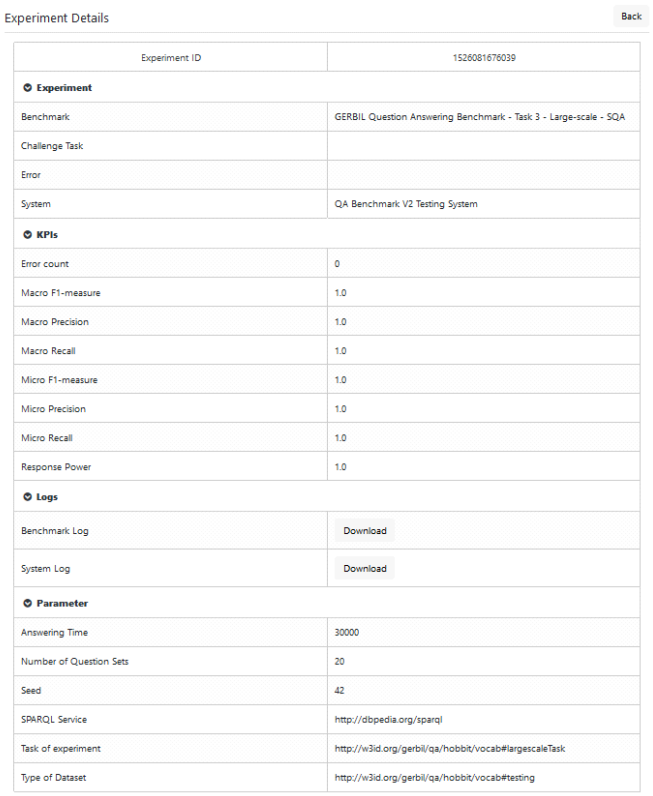

Figure 7 - Results for the Question Answering test system in a Large Scale QA experiment

In the example experiment, the system scored perfect results as it was always returning the correct answer for every question. Every finished experiment is repeatable and gets a citable URL.

5. Conclusion

We continue making an impact on the research community: we have run the QALD-7 challenge at the ESWC 2017 with Version 1 of the Question Answering Benchmark, while we are running the Scalable Question Answering (SQA) challenge at ESWC 2018, based on our newly developed Large Scale task. The winner will be announced at the end of this year's conference in Crete. The continuous acceptance by the Semantic Web community of our HOBBIT-based QA benchmark challenges is proof that our efforts are making a difference for the Question Answering community. With the continuous and expanding collaboration of cutting-edge initiatives and groups we will make sure this successful story does not fade out.