Semantic Web 1 (0) 1–5
IOS Press

MQALD: Evaluating the impact of modifiers in Question Answering over Knowledge Graphs

Lucia Siciliani [a,*], Pierpaolo Basile [a], Pasquale Lops [a], and Giovanni Semeraro [a]
[a] *Department of Computer Science, University of Bari Aldo Moro, Via E. Orabona 4, 70126, Bari, Italy*
E-mails: lucia.siciliani@uniba.it, pierpaolo.basile@uniba.it, pasquale.lops@uniba.it, giovanni.semeraro@uniba.it

Editors: First Editor, University or Company name, Country; Second Editor, University or Company name, Country
Solicited reviews: First Solicited Reviewer, University or Company name, Country; Second Solicited Reviewer, University or Company name, Country
Open reviews: First Open Reviewer, University or Company name, Country; Second Open Reviewer, University or Company name, Country

Abstract. Question Answering (QA) over Knowledge Graphs (KG) has the aim of developing a system that is capable of answering users' questions using the information coming from one or multiple Knowledge Graphs, like DBpedia, Wikidata and so on. Question Answering systems need to translate the question of the user, written using natural language, into a query formulated through a specific data query language that is compliant with the underlying KG. This translation process is already non-trivial when trying to answer simple questions that involve a single triple pattern and becomes even more troublesome when trying to cope with questions that require the presence of modifiers in the final query, i.e. aggregate functions, query forms, and so on. The attention over this last aspect is growing but has never been thoroughly addressed by the existing literature. Starting from the latest advances in this field, we want to make a further step towards this direction by giving a comprehensive description of this topic, the main issues revolving around it and, most importantly, by making publicly available a dataset designed to evaluate the performance of a QA system in translating such articulated questions into a specific data query language. This dataset has also been used to evaluate the best QA systems available at the state of the art.

Keywords: Question Answering, Knowledge Graphs, Dataset, Benchmark, SPARQL modifiers

1. Introduction

Question Answering (QA) is a challenging task that allows users to acquire information from a data source through questions written using natural language. The user remains unaware of the underlying structure of the data source, which could be anything ranging from a collection of textual documents to a database. For this reason, QA systems have always represented a goal for researchers as well as industries since it allows the creation of a Natural Language Interface (NLI). NLIs are capable of making accessible, especially to non-expert users, a huge amount of data that would have been undisclosed otherwise.

Several criteria can be used to categorize QA systems like the application domain (closed or open), or the kind of question that the system can handle (factoid, causal, and so on) [1]. Nevertheless, one of the most distinguishing aspects is the structure of the data source used to retrieve the answers, since this feature deeply affects the structure of the QA system and the methods that can be applied to retrieve the right answer.

*Corresponding author. E-mail: lucia.siciliani@uniba.it.

QA over structured data constitutes the root of the research revolving around this topic since it started as an attempt to create an NLI for databases [2]. Even if this research started in the late sixties, QA over structured data still remains a challenge and the recent advances in the Semantic Web have brought more attention to this topic. In fact, the spread and the constant evolution of Knowledge Graphs, e.g. DBpedia [3], Wikidata [4], has raised the urge to find novel ways to make easily accessible this great amount of data.

RDF[1] and SPARQL[2] represent the two standards chosen by the W3C for data description language and data query language for information resources of the Web. Expressing an information need using SPARQL to retrieve specific information is surely a nontrivial task, especially for users that have no knowledge of this kind of language. For this reason, QA systems have to fulfill the role of interpreters, which take charge of the translation of the users' information needs into SPARQL queries.

However translating the natural language, which is inherently ambiguous, into a formal query using a data query language like SPARQL is not easy. The main problem that makes this process particularly difficult is recognized in the literature as *lexical gap*. The lexical gap indicates the syntactic distance that exists between the natural language question and the correspondent SPARQL query and in order to bridge it, it is necessary to fulfill several Natural Language Processing tasks.

The first one is Entity Linking, i.e. find a match between portions of the question to entities within the KG. This process can be straightforward if there is string matching between the question and the label of a resource in the KG: in the question *"Which films were produced by Steven Spielberg?"*, *Steven Spielberg* can easily be mapped to its resource, e.g. `wiki:`[3]`Q8877` in Wikidata or `dbr:`[4] `Steven_Spielberg` in DBpedia through the label of the resources. However, many factors can easily make this task nontrivial. For example, there could be a periphrasis like in the question: *"When was born the President of the U.S.A.?"*.

Another important issue is related to the ambiguity of the natural language. In the question *"Who directed Philadelphia?"*, the word *Philadelphia* could refer to both the film or the city. In such cases, it is necessary to exploit the context provided by the question to select the right resource.

Similar issues can be found when trying to deal with the problem of Relation Linking, which requires a map between segments of the question and relations in the KG. Here the lexical gap can be even wider since there can be a great variety of expressions that refer to a certain relation. Questions like *"Who is the wife of Barack Obama?"*, *"Who is the spouse of Barack Obama?"*, *"Who married Barack Obama?"* always refer to the same relation, which is `dbo:`[5]`spouse` in DBpedia and `wiki:Property:P26` in Wikidata.

These two issues can be combined in several ways. An interesting example is represented by requests like *"Give me all Dutch parties"*. Here, to find the answer, the phrase *Dutch parties* requires a combination of Entity Linking and Relation Extraction and the relation is implicit. Moreover, not one but two relations need to be concatenated to obtain the final query, since we are looking for something that is a political party (first relation) and it has to be based in the Netherlands (second relation).

These two problems, intertwined, represent a struggle for any QA system and several techniques have been applied in the literature to overcome them. However, there is another kind of lexical gap that has to be considered. For example, to find an answer for questions like *"Which presidents were born in 1945?"*, mapping phrases of the question to the right resources in the KG is not sufficient: the question must be translated in a query which involves the proper SPARQL modifier. By SPARQL modifiers we refer to all the constructs that alter the basic SPARQL structure made up of a `SELECT` clause followed by a `WHERE` clause that encloses a graph pattern composed by one or more triple patterns. In the previous example, the question *"Which presidents were born in 1945?"* has to be translated into a query containing the `FILTER` modifier like shown in Listings 1.

```
SELECT DISTINCT ?uri WHERE {
?uri a dbo:President.
?uri dbo:birthDate ?date.
FILTER regex(?date, "^1945") }
```

Listing 1: SPARQL query for *"Which presidents were born in 1945?"*

[1] https://www.w3.org/TR/rdf11-concepts/
[2] https://www.w3.org/TR/rdf-sparql-query/
[3] https://www.wikidata.org/wiki/
[4] http://dbpedia.org/resource/
[5] http://dbpedia.org/ontology/

The SPARQL query language makes available several modifiers: query forms (like `ASK`, `DESCRIBE`, etc.), solution sequence modifiers (e.g. `ORDER BY`, `LIMIT`, `OFFSET`), functions (e.g. `FILTER`, `COUNT`, `SUM`, `AVG`, `NOW`, `YEAR`, `MONTH`). These modifiers can be used alone for simpler queries, but more frequently they have to be combined to obtain a SPARQL query which is correct and allows to retrieve all the right answers.

The construction of queries containing modifiers still represents an issue that has been explicitly explored in literature only by few works [5], [6], and [7], due to the complexity of the task and the lack of resources that can help researchers to develop and evaluate novel solutions. With this paper, we want to make a step forward towards this direction, and propose MQALD a dataset created to allow KGQA systems to take on this challenge and overcome the issues related to this specific research area.

The paper is organized as follows: in Section 2 we describe the state-of-the-art datasets and evaluation metrics used in this research area; in Section 3 we introduce our MQALD dataset, explaining how it was created and examining the modifiers it contains; in Section 4 we introduce the QA systems chosen for the evaluation and describe how it was performed; in Section 5 we analyze the results obtained by the chosen systems, and finally, in Section 6, we will draw the conclusion and propose some insights for future research.

2. Related Work

The development of a unified dataset for evaluating KGQA systems started in 2011 with the first edition of the Question Answering over Linked Data (QALD)[6] challenge. Within each edition of the challenge, the organizers always proposed different tasks, the most important being QA over DBpedia. The first dataset proposed for this task was composed by just 100 questions, equally split into a train and test set, but after each edition, this set of questions was modified and further expanded leading to a total number of 408 questions in the training set and 150 questions in the test set for the last edition of the challenge. QALD datasets contain questions that are manually compiled and curated to include different grades of complexity. Every dataset is composed of a set of English questions that (since QALD-3) are also translated in several languages, like German, Spanish, Italian, and French among others. Each question is associated with its answer (which can be a resource or a literal value) and, most importantly, also the SPARQL query that translates the input question.

Besides the QALD dataset, there are other datasets which have been proposed to benchmark QA systems over KGs, i.e. WebQuestions [8], SimpleQuestion [9], and LC-QuAD [10]. WebQuestions and SimpleQuestion were both built for Freebase [11].

WebQuestions contains 5,810 questions which were obtained by exploiting the Google Suggest API and the Amazon Mechanical Turk. Due to this generation process, questions within this dataset usually follow similar templates and each of them revolves around a single entity of the KG. All questions are associated with an entity of Freebase that can be found in the question and the right answer.

SimpleQuestion instead is the dataset with the highest number of questions and it was created to provide a richer dataset. It consists of 108,442 questions that have been created in two steps: in the first one, a list of facts or triples were extracted directly from Freebase, then these triples were sent to human annotators which were in charge of translating such triples into natural language questions. Like in the QALD dataset, each question is associated with its SPARQL version. The dataset derives its name from the fact that all the questions can be answered by exploiting just one relation of the KG.

Finally, LC-QuAD is also generated using templates that allow to formulate questions that are translated into SPARQL queries that require 2-hop relations. These templates are then reviewed by human annotators that verify the correctness of the questions. However, unlike SimpleQuestion and WebQuestions, the KG used for generating this dataset is DBpedia. The final dataset is composed by of 5,000 questions, with the corresponding SPARQL translations and the templates used for generating them.

As aforementioned, QALD represents not only a dataset but also a challenge that makes available a benchmark for QA systems. Factoid QA inherited its evaluation metrics from those of classical evaluation systems, thus estimating the performance in terms of Precision, Recall, and F-measure. Since the seventh edition of the challenge, the GERBIL benchmark platform [12, 13] was used to evaluate the performance of the participating systems. GERBIL, which is publicly

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

available online[7], embeds several QA systems and allows to evaluate different QA datasets, including those distributed during all the QALD challenges.

3. MQALD

3.1. Extracting mods from QALD

To evaluate the performance of QA systems over complex queries, we decided to isolate questions that require a modifier from the training and test sets of the QALD datasets creating a new one called MQALD[8]. As stated in Section 2, in each edition of the challenge the dataset is obtained as an extension of the dataset of the previous edition, although we noticed there was not a complete overlap. For this reason, we decided to merge the questions contained in the last three editions of the QALD challenge over DBpedia: QALD-7 [14], QALD-8 [15], and QALD-9 [16]. The merge of the datasets was done separately for both training and test data. Since training data could have been used to develop the systems, we excluded from the test set those questions that also appeared within the training data. This can happen since a more recent QALD test set can contain questions belonging to a previous QALD training set.

We thus obtained six files:

- **MQALD_train**: contains the merge of all questions coming from the training sets of QALD-7, QALD-8, QALD-9;
- **MQALD_train_MODS**: contains only the questions that require a modifier extracted from MQALD_train;
- **MQALD_train_NO_MODS**: contains only the questions that do not require a modifier extracted from MQALD_train;
- **MQALD_test**: contains the merge of all questions coming from the test sets of QALD-7, QALD-8, QALD-9;
- **MQALD_test_MODS**: contains only the questions that require a modifier extracted from MQALD_test;
- **MQALD_test_NO_MODS**: contains only the questions that do not require a modifier extracted from MQALD_test;

[7] http://gerbil-qa.aksw.org/gerbil/

[8] The dataset is available on Zenodo: http://doi.org/10.5281/zenodo.4050353

Dataset	#questions
MQALD_train	462
MQALD_train_MODS	154
MQALD_train_NO_MODS	308
MQALD_test	115
MQALD_test_MODS	41
MQALD_test_NO_MODS	74

Table 1
Number of questions included in each file.

The number of questions included in each file is reported in Table 1. Each file is in JSON format and compliant with the QALD dataset structure, i.e. each question is represented by a JSON Object which is then stored in a JSON array that contains all the questions. The only exceptions in the structure of each question are represented by the presence of the attribute `qald-version` which indicates the QALD dataset of origin and a further JSON array, called `modifiers`, that contains all the modifiers occurring in the SPARQL query (this array is empty if there are no modifiers). An example of a JSON Object representing a question is given in Listing 2.

```
{
"id":22,
"qald-version":"9",
"aggregation":true,
"hybrid":false,
"modifiers":[
"HAVING","GROUP BY","COUNT"
],
"question":[
{
"string":"Which countries have more
   than ten volcanoes?",
"keywords":"countries,  more than
   ten volcanoes",
"language":"en"
},
...
],
"answertype":"resource",
"onlydbo":true,
"query": {
"sparql":"SELECT DISTINCT ?uri WHERE
    { ?x a <http://dbpedia.org/
   ontology/Volcano> ; <http://
   dbpedia.org/ontology/
   locatedInArea> ?uri . ?uri a <
```

```
    http://dbpedia.org/ontology/
    Country> } GROUP BY ?uri HAVING
    ( COUNT(?x) > 10 )"},
"answers":[
{
"head":
{
"vars":["uri"]
},
"results":
{
"bindings":[
{
"uri":
{
"type":"uri",
"value":"http://dbpedia.org/resource
    /Ecuador"}
},
...
]
}
}
]
}
```

Listing 2: JSON structure of a question within the dataset

3.2. Modifiers within the dataset

It is important to notice that none of the QALD datasets contains every possible keyword and function available within the SPARQL Query Language, but just a small subset of them. In this section, we will describe in detail the modifiers that are contained in the dataset through real examples extracted from MQALD. The modifiers will be listed according to their frequency. The occurrences of each modifier within the training dataset, the test dataset as well as the total number of occurrences are reported in Table 2.

3.2.1. LIMIT

The `LIMIT` modifier is the most frequent among all the others in the dataset. Usually, it is used in combination with `ORDER BY` and `OFFSET` which allows answering questions containing superlatives, i.e. *"Who is the tallest player of the Atlanta Falcons?"* or *"For which label did Elvis record his first album?"*.

Modifier	MQALD_train	MQALD_test	Total
LIMIT	48	11	59
ORDER BY	45	9	54
FILTER	34	16	50
ASK	40	3	43
UNION	33	9	42
OFFSET	33	5	38
COUNT	29	7	36
GROUP BY	3	2	5
HAVING	3	1	4
YEAR	2	2	4
NOW	0	2	2

Table 2
Frequencies of each modifier within the dataset.

Another combination is composed by the previous one (`LIMIT`, `ORDER BY`, and `OFFSET`) with the addition of the `COUNT` aggregation function for questions like *"Which country has the most official languages?"* where the `ORDER BY` needs to be applied to the object of the `COUNT` function as illustrated by Listing 3.

```
SELECT DISTINCT ?uri WHERE {
?uri a dbo:Country;
dbp:officialLanguages ?language.}
ORDER BY DESC(COUNT(?language))
OFFSET 0 LIMIT 1
```

Listing 3: SPARQL query for *"Which country has the most official languages?"*

The `LIMIT` modifier is also used by itself in some questions, even if its usage can be quite controversial. An example is represented by the question *"With how many countries Iran has borders?"* that translates into a SPARQL query with the `LIMIT` modifier set to 8. This is correct since by executing the query against DBpedia without this modifier, only the first 8 results represent the right answer (i.e. the resources corresponding to the countries Iran has borders with), while the following are just literals not relevant with the question. However, in this case, the usage of the `LIMIT` modifier is deeply connected to the structure of DBpedia.

3.2.2. ORDER BY

The second most common modifier is `ORDER BY`, which is used to alter the order of the solution sequence. It is important to notice that within the dataset the `ORDER BY` modifier is always used in conjunction with other modifiers with exception of one question.

The only question where ORDER BY appears by itself is *"What are the top selling luxury vehicle brands in Germany?"*, where the answer is represented by all the results of an ordered list.

3.2.3. FILTER

The FILTER keyword is used to restrict the number of results that match with the graph pattern specified just before the FILTER itself. The FILTER keyword precedes an expression which can be constituted by a wide variety of operators that are categorized according to the number of parameters involved.

There are *unary operators* like "!" which represents the logical NOT and "BOUND" that checks if its argument is bounded to a value. These operators can be used alone or in conjunction to answer questions like *"Is Frank Herbert still alive?"*. In this case, the answer can be found only checking if the object of the relation res:Frank_Herbert dbo:deathDate ?date does not exist in the KG (i.e. FILTER(!BOUND(?date))).

We also have *binary operators* that comprise logical connectives AND "&&" and OR "||", as well as all the other test operators between two values (like ==, !=, >, <, and so on). These are used to answer questions like*"Which caves have more than 3 entrances?"* creating a SPARQL query as shown by Listing 4.

Finally, there is also a single *ternary operator* which is REGEX, that checks if a certain string (first parameter) matches a regular expression (second parameter). The third parameter is an optional flag, which specifies a case-insensitive pattern match). This ternary operator can be used in questions like *"Is the wife of president Obama called Michelle?"* to check if the name *"Michelle"* is contained within the label of the resource dbo:Michelle_Obama.

```
SELECT DISTINCT ?uri WHERE {
?uri a dbo:Cave;
dbp:entranceCount ?entrance
FILTER (?entrance>3)}
```

Listing 4: SPARQL query for *"Which caves have more than 3 entrances?"*

3.2.4. ASK

Among all the possible SPARQL Query forms besides SELECT, the only one which is present within the dataset is ASK which is reasonable since QA systems on KGs are usually factoid QA systems. The ASK query form is necessary to construct questions that require a Yes/No answer thus returning a Boolean value, like *"Did Socrates influence Aristotle?"*.

```
ASK WHERE {
res:Aristotle
dbo:influencedBy res:Socrates }
```

Listing 5: SPARQL query for *"Did Socrates influence Aristotle?"*

3.2.5. UNION

The UNION modifier is used to merge the results of two different patterns. Within the dataset, it is usually employed to take into account more relations for obtaining the full set of answers as shown in Listing 6.

```
SELECT DISTINCT ?uri WHERE {
?uri a dbo:Company
{?uri dbo:location dbr:Munich}
UNION
{?uri dbo:headquarter dbr:Munich}
UNION
{?uri dbo:locationCity dbr:Munich}}
```

Listing 6: SPARQL query for *"Give me all companies in Munich."*

Despite being correct, the use of the UNION modifier in this query is associated to how the KG is structured, i.e. it is necessary to know that for retrieving all the companies that are in Munich we need three relations (dbo:location, dbo:headquarter, and dbo:locationCity).

3.2.6. OFFSET

The OFFSET modifier appears to be particularly trivial within the dataset. The majority of times, the queries include an OFFSET of 0, which has no effect over the results and therefore could also be neglected. However, there are some questions where there is an OFFSET of a different number that allows answering correctly to the input question but appears controversial. Let us consider the question *"How many foreigners speak German?"*. This question is translated into a SPARQL query where OFFSET 1 is applied over the object of the relation dbr:German_language dbp:speakers ?Ger_lang. This relation has as object three literal values, namely: *"L2 speakers: 10–15 million"*, *"as a foreign language: 75–100 mil-*

lion", and *"million to 95 million"*, so actually the use of `OFFSET 1` allows to pick up the second literal which represents the right answer. Nevertheless, obtaining this SPARQL query without knowing how the KG is structured is infeasible, especially considering that in this case (but this happens frequently in DBpedia), the property `dbp:speakers` does not have any specification with respect of its domain and range thus the object of this property is not consistent (there can be strings, numbers or even resources). A similar scenario occurs for the question *"Where is the most deep point in the ocean?"* where a `OFFSET 13` is required to retrieve the answer.

The other existing case of non-zero `OFFSET` ensues from the question *"What is the second highest mountain on Earth?"*, where the use of `OFFSET 1` is justified by the semantics of the question.

```
SELECT DISTINCT ?uri WHERE {
?uri a dbo:Mountain;
dbo:elevation ?elevation}
ORDER BY DESC(?elevation)
OFFSET 1 LIMIT 1
```

Listing 7: SPARQL query for *"What is the second highest mountain on Earth?"*

3.2.7. COUNT

Besides from being used in combination with `ORDER BY` and `LIMIT`, the `COUNT` modifier is used alone, for example in questions that start with *"How many"*, like *"How many films did Leonardo DiCaprio star in?"*.

3.2.8. GROUP BY - HAVING

The `GROUP BY` modifier is used to divide the result set into groups which can then be used as input to an aggregation function like `COUNT`. If the query requires to filter the result of aggregation, then it must be introduced with the modifier `HAVING` (instead of the keyword `FILTER`), like it happens for the question *"Which countries have more than ten caves?"*.

```
SELECT DISTINCT ?uri WHERE {
?uri a dbo:Country. ?cave a dbo:Cave
{ ?cave dbo:location ?uri } UNION
{ ?cave dbo:location ?loc .
?loc dbo:country ?uri } }
GROUP BY ?uri
HAVING ( COUNT(?cave) > 10 )
```

Listing 8: SPARQL query for *"Which countries have more than ten caves?"*

3.2.9. YEAR - NOW

SPARQL makes available several functions for dates and times, even though `YEAR` and `NOW` are the only two occurring in the dataset. The `YEAR` function returns the year part of the date given as argument, while the `NOW` function does not require an argument and simply returns the current time (formatted according to the XSD dateTime format). These functions can be used to answer very peculiar questions, like *"Give me all American presidents of the last 20 years."*.

```
SELECT DISTINCT ?uri WHERE {
?uri rdf:type dbo:Person;
dct:subject
dbc:Presidents_of_the_United_States;
dbo:activeYearsEndDate ?d
FILTER ((year(now())-year(?d))<=20)}
```

Listing 9: SPARQL query for *"Give me all American presidents of the last 20 years."*

3.3. MQALD Extension

From Table 1 is possible to note how the ratio between questions with and without modifiers is still unbalanced within the QALD datasets. We thus decided to further expand the MQALD dataset by adding more questions containing modifiers.

The proposed extension, MQALD_ext[9], contains 100 novel questions involving different modifiers which were manually created by a human annotator. Each question has been translated into four languages: English, Italian, French, and Spanish. In particular, for the Italian language, questions were checked by native speakers.

Questions were built paying attention so that the modifiers occurring in each SPARQL query were:

- mandatory: the use of the modifier is indispensable to retrieve the right answer;
- KG independent: modifiers are not used to adapt the query to a particular structure of the underlying KG.

[9] Available on Zenodo: http://doi.org/10.5281/zenodo.4050353

Taking into account these two aspects is particularly important since in this way it is possible to guarantee that each modifier is connected to a specific semantics of the natural language question. As discussed in section 3.2, exceptions to these rules can easily lead to questions that are impossible to answer without knowing the structure of the KG. The most noticeable examples were represented by questions containing *UNION* and *OFFSET* modifiers. Listings 10 and 11 show two examples of those queries coming from MQALD_ext, where these modifiers were used respecting KG independence.

```
SELECT ?uri WHERE {
?uri rdf:type dbo:Band.
?uri dbo:genre dbr:Electronic_music.
{{?uri dbo:hometown dbr:France}
UNION
{?uri dbo:hometown dbr:Italy}}}
```

Listing 10: SPARQL query for *"Give me all the electronic music bands having hometown in France or in Italy."*

```
SELECT ?country WHERE
{?country rdf:type dbo:Country;
dbo:populationTotal ?poptot}
ORDER BY DESC(xsd:integer(?poptot))
LIMIT 1 OFFSET 4
```

Listing 11: SPARQL query for *"Which is the fifth most populous country in the world?"*

Nevertheless, it is important to notice that a periodic maintenance is needed to keep all the questions consistent with the KG since KG like DBpedia are constantly being updated. Table 3 shows the distribution of the modifiers in our expansion.

4. Evaluation

We compare the performance of the most advanced QA systems for KGs over the different datasets we constructed. The goal of our evaluation is to examine how QA systems perform on questions that require one or more modifiers within the final SPARQL query and how this result impacts the performance of the system itself. The QA systems chosen for the evaluation are the following:

Modifier	ext
LIMIT	25
ORDER BY	31
FILTER	40
ASK	19
UNION	11
OFFSET	6
COUNT	26
GROUP BY	15
HAVING	10
YEAR	3
NOW	0

Table 3
Frequencies of each modifier within MQALD_ext.

- **gAnswer2**[10]: the system proposed by Hu et al. [17] that won the QALD-9 challenge. The method used by this system consists in transforming the input question into a Semantic Query Graph, a particular graph where each node corresponds to a resource and each vertex to a relation of the underlying KG. Resources are found using DBpedia Lookup, while relations are extracted using a Multi-Channel Convolutional Neural Network. Next, the Semantic Query Graph is translated into a SPARQL question that is then used to obtain the final answer;
- **QAnswer**[11]: a QA system developed by Diefenbach et al. [18]. It represents the extension of the WDaqua-core0 system [19] that took part in the QALD-9 challenge gaining the second place. The approach behind QAnswer is based upon four steps: the first one consists in retrieving all the resources that can be linked to a question by considering its n-grams, the second phase uses these resources to create all the possible queries according to specific patterns, in the third step these queries are then ranked according to several features and finally the query that obtained the highest score is executed and the answer is returned to the user;
- **TeBaQA**[12]: created by Peter Nancke et al. from Leipzig University in Germany, exploits a template-based approach. The SPARQL templates are generated upon the questions available for the QALD challenge. Unfortunately, this work is still

[10] http://ganswer.gstore-pku.com/api/qald.jsp
[11] http://qanswer-core1.univ-st-etienne.fr/api/gerbil
[12] http://139.18.2.39:8187/

unpublished so the details of the approach are still unknown. TeBaQA ranked third during the QALD-9.

As reported in Section 2, GERBIL represents a platform that allows evaluating QA systems using several different datasets, even customized ones and it is also used to obtain the results for all the QALD challenges. However, an evaluation with GERBIL was infeasible for several reasons. First of all, among the QA systems connected to GERBIL, only gAnswer2 was able to return any result. For all the other systems, GERBIL returned one of two error messages, namely *"The annotator couldn't be loaded"* or *"The annotator caused too many single errors"*. Secondly, the experiment configuration of GERBIL does not allow to create an experiment with both a custom system answer file and a custom dataset: these two options can only be used separately.

For these reasons we decided to implement an evaluation tool[13] following the guidelines reported in the QALD-9 report [16]. In particular, the evaluation script computes for each query q a set of metrics, such as precision (P), recall (R), and F-measure according to the following equations:

$$P(q) = \frac{\#correct\ system\ answers\ for\ q}{\#system\ answers\ for\ q}$$

$$R(q) = \frac{\#correct\ system\ answers\ for\ q}{\#gold\ standard\ answers\ for\ q}$$

$$F(q) = \frac{2 \times P(q) \times R(q)}{P(q) + R(q)}$$

Moreover, there are additional information to take into account:

- If the golden answer-set is empty and the system retrieves an empty answer, precision, recall and F-measure are equal to 1.
- If the golden answer-set is empty but the system responds with any answer-set, precision, recall, and F-measure are equal to 0.
- If there is a golden answer but the the QA system responds with an empty answer-set, it is assumed that the system could not respond. Then the precision, recall, and F-measure are equal to 0.
- In any other case, the standard precision, recall, and F-measure are computed.

In our paper, we consider the macro measures as the QALD challenge: we calculated precision, recall and F-measure per question and averaged the values at the end. As adopted in QALD, for the final evaluation, the Macro F1 QALD metric is used. This metric uses the previously mentioned additional information with the following exception:

- If the golden answer-set is not empty but the QA system responds with an empty answer-set, it is assumed that the system determined that it cannot answer the question. Then the precision is set to 1 and the recall and F-measure to 0.

5. Results

The results of the evaluation of the three mentioned QA systems, i.e. gAnswer, QAnswer, and TeBaQA are reported in Table 4.

Overall, an analysis of the performance of the systems in terms of F1-QALD measure for the datasets without modifiers (no_mods) and those containing all the questions (with and without modifiers), confirms the results reported by the QALD-9 challenge: gAnswer represents the best system at the state-of-the-art for this task, followed by QAnswer and TeBaQA.

However, this ranking changes when comparing the results for the train/test data containing only questions that require modifiers. In fact, in this case, QAnswer outperforms the results obtained by gAnswer. To obtain more insights regarding this aspect, we checked which questions that require modifiers are answered by each system within the test set, and the results are shown in Table 5 for questions contained in MQALD and Table 6 for questions belonging to our extension. Unfortunately, only the QAnswer API allows to obtain the SPARQL translation of the question created by the system but by analyzing Precision, Recall and F-measure calculated for each question as well as the set of answers returned by each system, we can have an insight on how they handle modifiers.

Out of the 41 questions that compose the MQALD test dataset, the number of questions where the F-score is not equal to zero is 3 for gAnswer, 7 for QAnswer, and 3 for TeBaQA. Most of the questions answered by the three systems involve the `UNION`

[13] The evaluation script and the code used for developing the dataset is available on github: https://github.com/lsiciliani/MQALD

	gAnswer35				QAnswer				TeBaQA			
	P	**R**	**F**	**F1-Q**	**P**	**R**	**F**	**F1-Q**	**P**	**R**	**F**	**F1-Q**
qald-9_test	.607	.316	.296	**.416**	.459	.222	.197	.299	.644	.141	.139	.231
mqald_train_all	.577	.334	.313	**.423**	.529	.334	.302	.409	.602	.154	.144	.246
mqald_train_MODS	.350	.054	.027	.094	.356	.223	.175	**.274**	.479	.051	.047	.092
mqald_train_NO_MODS	.474	.466	.456	**.470**	.612	.392	.369	.478	.657	.199	.186	.306
mqald_test_all	.590	.295	.284	**.393**	.465	.232	.203	.309	.634	.155	.156	.249
mqald_test_MODS	.608	.049	.054	.091	.391	.139	.087	**.205**	.671	.045	.048	.085
mqald_test_NO_MODS	.580	.431	.412	**.495**	.515	.269	.260	.353	.606	.215	.215	.317
mqald_ext	.472	.032	.032	.060	.320	.093	.071	**.144**	.483	.036	.045	.068
mqald_test_MODS_ext	.497	.037	.031	.069	.550	.083	.056	**.145**	.531	.039	.046	.072
mqald_test_all_ext	.526	.173	.162	**.260**	.394	.161	.137	.228	.562	.104	.109	.176

Table 4

Results of the three systems over the six datasets. The metrics used are Precision (P), Recall (R), F-Measure (F), and F1-QALD measure (F1-Q).

gAnswer					
id	**Question**	**Modifiers**	**P**	**R**	**F**
7	Which software has been published by Mean Hamster Software?	UNION	1	1	1
19	Give me all cars that are produced in Germany.	UNION	.920	.191	.316
24	Which countries are connected by the Rhine?	UNION	1	.833	.909
QAnswer					
id	**Question**	**Modifiers**	**P**	**R**	**F**
1	What is the highest mountain in Germany?	LIMIT, ORDER BY	.001	1	.002
5	Which airports are located in California, USA?	UNION	1	.351	.520
7	Which software has been published by Mean Hamster Software?	UNION	1	1	1
8	How many grand-children did Jacques Cousteau have?	COUNT	1	1	1
12	What is the longest river in China?	LIMIT, OFFSET, ORDER BY	.016	1	.032
15	Is Pamela Anderson a vegan?	ASK	1	1	1
31	Which politicians were married to a German?	UNION	.063	.091	.074
TeBaQA					
id	**Question**	**Modifiers**	**P**	**R**	**F**
6	Which countries in the European Union adopted the Euro?	UNION	.5	.022	.043
14	How many awards has Bertrand Russell?	COUNT	1	1	1
24	Which countries are connected by the Rhine?	UNION	1	.833	.909

Table 5

List of the questions answered by each system over the mqald_test_MODS dataset.

modifier. As stated in Section 3.2, within the QALD datasets the `UNION` modifier is exclusively used to merge the results of different triple patterns rather than encapsulating a particular semantics and sometimes are even unnecessary due to the constant evolution of DBpedia. This is the example of the question *"Which software has been published by Mean Hamster Software?"* where the `UNION` modifier is not needed at all since the query `SELECT DISTINCT ?uri WHERE {?uri onto:publisher res:Mean-`

gAnswer					
id	**Question**	**Modifiers**	**P**	**R**	**F**
164	Give me all Stephen King books for which the publication date is known.	FILTER	.488	.976	.651
189	Give me the list of who directed and produced Saving Private Ryan.	UNION	.600	.750	.667
226	What are the first publication date and the last publication date of Mazinger Z?	UNION	1	.500	.667
243	Which Elvis songs are part of the Elvis is Back album?	FILTER, ORDER BY	.106	1	.192
QAnswer					
id	**Question**	**Modifiers**	**P**	**R**	**F**
162	What is the lowest mountain in Germany?	LIMIT, ORDER BY	.001	1	.002
169	Is Leonardo Da Vinci the author of the Mona Lisa?	ASK	1	1	1
170	What are the first 10 works in alphabetical order made by Leonardo da Vinci?	LIMIT, ORDER BY	.286	1	.444
182	Which films of the Jurassic Park saga has Steven Spielberg directed?	FILTER	.667	1	.800
183	How many movies has George Lucas directed?	COUNT	1	1	1
189	Give me the list of who directed and produced Saving Private Ryan.	UNION	1	.250	.400
194	Is Barack Obama a politician?	ASK	1	1	1
220	Which films in which Catherine Zeta-Jones starred in have earned less than the initial budget?	FILTER, ORDER BY	.077	1	.143
226	What are the first publication date and the last publication date of Mazinger Z?	UNION	1	.5	.667
231	Who are the author and publisher of Don Quixote?	UNION	1	.5	.667
238	How many cartoons are produced by Walt Disney?	COUNT	1	1	1
TeBaQA					
id	**Question**	**Modifiers**	**P**	**R**	**F**
170	What are the first 10 works in alphabetical order made by Leonardo da Vinci?	LIMIT, ORDER BY	.250	.1	.143
174	Give me all the electronic music bands having hometown in France or in Italy.	UNION	.020	.043	.027
176	What are the birthplace and the death place of Elvis Presley?	UNION	1	.500	.667
186	What are the birth names of Tom Cruise and Whoopi Goldberg?	UNION	1	.500	.667
190	What are the capitals of Italy and France?	UNION	1	.500	.667
201	How many children does Barack Obama have?	COUNT	1	1	1
231	Who are the author and publisher of Don Quixote?	UNION	1	.500	.667
239	Give me the population density of Warsaw as well as its population total.	UNION	1	.500	.667

Table 6

List of the questions answered by each system over the mqald_ext dataset.

_Hamster_Software} is sufficient to retrieve all the answers. This could be due to a mistake or to a lack of update of the dataset and allows gAnswer and QAnswer to obtain the right answer without actually using the modifier. Another case is represented instead by questions like *"Which countries are connected by*

the Rhine?" where the UNION modifier is well used since it is necessary to retrieve the whole set of answers provided by the QALD. In this case, gAnswer and TeBaQA manage to return an answer but only the precision is equal to 1, meaning that not all the answers have been retrieved and UNION was not used.

The same happens for all the questions where the F-score is lower than 1. For example, QAnswer answers the question *"What is the longest river in China?"* by returning all the rivers that flow through China and, of course, the longest river is also included in this set. The use of the three modifiers (LIMIT, OFFSET, and ORDER BY) would have allowed to properly order the list by the rivers' length and pick the first element of the ordered list. With this result, we can deduce that modifiers were not used to retrieve the answer.

Also for the question *"How many grand-children did Jacques Cousteau have?"* we can observe a misleading behaviour since the number of grand-children of Jacques Cousteau actually coincides with the number of his children, thus QAnswer could answer correctly by just considering the property dbp:children. It is important to notice that the property dbp:children does not exist in the current version of DBpedia: we just have the property dbo:child which has as object the resources of two of the four children of Jacques Cousteau. This implies that also this question should be updated properly in order to be compliant with the KG. Nevertheless, the property dbp:children is available in the 2016-10 version of DBpedia and the object of this property is 4, which can be obtained without modifiers and erroneously used as the correct answer.

There is only one question that requires an ASK among all the others: "Is Pamela Anderson a vegan?". Only QAnswer returns the right answer but using a SPARQL query that is completely unrelated to the question that just checks the existence of the resource dbr:Pamela_Anderson.

An exception is represented by the question *"How many awards has Bertrand Russell?"* which requires the COUNT modifier since this information is not available in any other way from DBpedia.

Regarding the questions included in MQALD_ext, 4 answers were provided by gAnswer, 11 by QAnswer, and 8 by TeBaQA. The only two systems that manage to return a full answer, with Precision, Recall, and F-measure at 1, are QAnswer and TeBaQA. As stated in Section 3.3, MQALD_ext questions are formulated such that modifiers are necessary to retrieve the correct set of answers thus by analyzing these scores it is possible to easily identify which modifiers are covered by which system. QAnswer is able to manage the ASK and COUNT modifiers while TeBAQA is capable to handle the COUNT modifiers as well. Of course, if a system is able of handling a specific modifier it does not mean that it will answer all the questions containing that modifier since the lexical gap still represents a major problem in the translation from natural language to SPARQL.

Overall, the results show how all the systems under analysis do not perform well at handling modifiers just with a few exceptions. Nevertheless, the capability of QAnswer to handle ASK and COUNT modifiers and better bridge the lexical gap by at least returning partial answers granted it the best score among the three selected systems.

6. Conclusions and Future Works

In this paper, we have introduced MQALD, a dataset built upon the last three editions of the QALD to identify and collect all the questions that contain modifiers. These modifiers were then analyzed to better understand their functioning considering the SPARQL syntax and how they can reflect specific semantics of the natural language query.

Moreover, we also proposed an extension to MQALD (MQALD_ext), containing novel questions that have been manually created to ensure that modifiers would be necessary and independent from the KG.

Through the evaluation of the performance of the top three systems available at the state-of-the-art, emerged that there is still much work that must be done to create a QA system capable to handle this kind of questions, not only from an architectural point of view but also in creating and updating the existing resources to allow these systems to be fairly compared.

As future work, we plan to further encourage the research and development of new solutions to improve the performance of QA systems over KG handling questions involving modifiers, which are very common in natural language. The first step towards this direction would be to properly revise the available resources, fixing the datasets so that question/query pairs are compliant with the last version of DBpedia, and making sure that the use of a specific modifier is the only way to retrieve right answers. Next, it would be beneficial for the whole community to further expand this dataset, so that all the modifiers available in SPARQL are properly represented by a sufficient amount of questions.

References

[1] A. Mishra and S.K. Jain, A survey on question answering systems with classification, *Journal of King Saud University-Computer and Information Sciences* **28**(3) (2016), 345–361.

[2] I. Androutsopoulos, G.D. Ritchie and P. Thanisch, Natural language interfaces to databases–an introduction, *Natural language engineering* **1**(1) (1995), 29–81.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, Dbpedia: A nucleus for a web of open data, in: *The semantic web*, Springer, 2007, pp. 722–735.

[4] D. Vrandečić and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* **57**(10) (2014), 78–85.

[5] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann and A.-C. Ngonga Ngomo, Survey on challenges of question answering in the semantic web, *Semantic Web* **8**(6) (2017), 895–920.

[6] D. Diefenbach, V. Lopez, K. Singh and M. Pierre, Core Techniques of Question Answering Systems over Knowledge Bases: a Survey, *Knowledge and Information systems* (2017).

[7] L. Siciliani, D. Diefenbach, P. Maret, P. Basile and P. Lops, Handling Modifiers in Question Answering over Knowledge Graphs, in: *International Conference of the Italian Association for Artificial Intelligence*, Springer, 2019, pp. 210–222.

[8] J. Berant, A. Chou, R. Frostig and P. Liang, Semantic parsing on freebase from question-answer pairs, in: *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1533–1544.

[9] A. Bordes, N. Usunier, S. Chopra and J. Weston, Large-scale simple question answering with memory networks, *arXiv preprint arXiv:1506.02075* (2015).

[10] P. Trivedi, G. Maheshwari, M. Dubey and J. Lehmann, Lc-quad: A corpus for complex question answering over knowledge graphs, in: *International Semantic Web Conference*, Springer, 2017, pp. 210–218.

[11] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[12] R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann et al., GERBIL: general entity annotator benchmarking framework, in: *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 1133–1143.

[13] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. Ngonga-Ngomo, C. Demmler and C. Unger, Benchmarking question answering systems, *Semantic Web* **10**(2) (2019), 293–304.

[14] R. Usbeck, A.-C.N. Ngomo, B. Haarmann, A. Krithara, M. Röder and G. Napolitano, 7th open challenge on question answering over linked data (QALD-7), in: *Semantic Web Evaluation Challenge*, Springer, 2017, pp. 59–69.

[15] R. Usbeck, A.-C.N. Ngomo, F. Conrads, M. Röder and G. Napolitano, 8th challenge on question answering over linked data (QALD-8), *language* **7** (2018), 1.

[16] R. Usbeck, R.H. Gusmita, A.-C.N. Ngomo and M. Saleem, 9th challenge on question answering over linked data (QALD-9), in: *Semdeep/NLIWoD@ ISWC*, 2018, pp. 58–64.

[17] S. Hu, L. Zou and X. Zhang, A state-transition framework to answer complex questions over knowledge base, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2098–2108.

[18] D. Diefenbach, A. Both, K. Singh and P. Maret, Towards a question answering system over the semantic web, *Semantic Web* (2018), 1–19.

[19] D. Diefenbach, K. Singh and P. Maret, Wdaqua-core0: A question answering component for the research community, in: *Semantic Web Evaluation Challenge*, Springer, 2017, pp. 84–89.