Improving Question Answering Quality Through Language Feature-Based SPARQL Query Candidate Validation

Aleksandr Gashkov[1], Aleksandr Perevalov[2,3], Maria Eltsova[1], and Andreas Both[3,4(✉)]

[1] Perm National Research Polytechnic University, Perm, Russia
[2] Anhalt University of Applied Sciences, Köthen, Germany
[3] Leipzig University of Applied Sciences, Leipzig, Germany
andreas.both@htwk-leipzig.de
[4] DATEV eG, Nuremberg, Germany

Abstract. Question Answering systems are on the rise and on their way to become one of the standard user interfaces. However, in conversational user interfaces, the information quantity needs to be kept low as users expect a limited number of precise answers (often it is 1) – similar to human-human communication. The acceptable number of answers in a result list is a key differentiator from search engines where showing more answers (10–100) to the user is widely accepted. Hence, the quality of Question Answering is crucial for the wide acceptance of such systems. The adaptation of natural-language user interfaces for satisfying the information needs of humans requires high-quality and not-redundant answers. However, providing compact and correct answers to the users' questions is a challenging task. In this paper, we consider a certain class of Question Answering systems that work over Knowledge Graphs. We developed a system-agnostic approach for optimizing the ranked lists of SPARQL query candidates produced by the Knowledge Graph Question Answering system that are used to retrieve an answer to a given question. We call this a SPARQL query validation process. For the evaluation of our approach, we used two well-known Knowledge Graph Question Answering benchmarks. Our results show a significant improvement in the Question Answering quality. As the approach is system-agnostic, it can be applied to any Knowledge Graph Question Answering system that produces query candidates.

Keywords: Question Answering over Knowledge Graphs · Query Validation · Query Candidate Filtering

1 Introduction

The Web has become the major knowledge source for many people worldwide. While aiming at efficient knowledge modeling and representation, the Semantic

A. Gashkov and A. Perevalov—Shared first authorship–these authors contributed equally to this work.

P. Groth et al. (Eds.): ESWC 2022, LNCS 13261, pp. 217–235, 2022.
https://doi.org/10.1007/978-3-031-06981-9_13

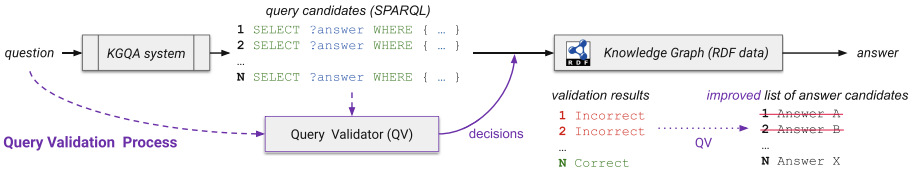

Fig. 1. General overview of the Query Validation process. The core component is the *Query Validator* intended to filter incorrect query candidates.

Web initiative was proposed and is permanently growing. The objective of this initiative is to make Web data machine-readable and machine-understandable by describing concepts, entities, and relations between them [6]. Hence, the Semantic Web may be considered as a giant Knowledge Graph (KG). In this regard, Knowledge Graph Question Answering (KGQA) systems are actively developing already for more than a decade [14,16]. These systems are bridging the gap between Linked Data and end-users by transforming natural-language (NL) questions into structured queries (e.g., represented as SPARQL[1]) to make the information accessible using NL requests.

When answering a question, KGQA systems often generate a ranked list of SPARQL queries that are considered to be capable of retrieving answers to a given question. Thereafter, a ranked Top-N of the retrieved answers is shown to the end-users (often N is 1). Thus, a *Query Candidate* is a SPARQL query generated by a KGQA system to retrieve data. An *Answer Candidate* is a result of a SPARQL Query Candidate execution which is proposed as a possible answer to a user. In this paper, we propose a *Query Validation* (QV) process that is intended to remove all queries that cannot be resolved to a correct answer from a query candidates list. This helps to reduce the number of incorrect query candidates (and therefore, answers) in the output and move the correct ones to the top of the list. In addition, unanswerable questions should be recognized, and, hence, an empty output should be presented for such questions (s.t., users are not confronted with incorrect/guessed answers).

The field of Answer Validation (AV) is well-researched for information retrieval (IR) and IR-based question answering (QA) and many approaches were proposed in the last decade [1,5,21,22,30,39,47]. However, there is just a very limited number of studies on AV and QV in the context of KGQA systems (e.g., [9,27]).

In this work, we propose a new system-agnostic QV approach that can determine whether a query candidate produced by a KGQA system is correct or not without executing the SPARQL query (see Fig. 1). The approach uses a straightforward process of converting a query candidate to NL representation and a fine-tuned classifier [11] to distinguish between correct and incorrect query candidates and, therefore, the answers. In Fig. 1, the process is visualized. To demonstrate the efficiency of the proposed QV approach, we utilize several well-known QA

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

quality metrics, such as Precision@k and NDCG@k (Normalized Discounted Cumulative Gain) [37]. To tackle the dilemma of how to address the difference between "guessing" answers versus providing an empty result, we introduce a new integral metric that takes into account correct, incorrect, and also empty answer sets. In addition, we consider the quality w.r.t. the unanswerable questions and the influence of our approach on the corresponding results. Given our experimental results on one KGQA system, QAnswer [12], and two benchmarking datasets, LC-QuAD 2.0 [17] and RuBQ 2.0 [35], we demonstrate that the QV approach provides a relative improvement of Precision@1 up to 204.6% (see Table 2) as well as it is improving other metrics significantly. Moreover, the approach enabled us to obtain almost 50% of correct answers for the unanswerable questions.

To increase the reproducibility of our work, we performed evaluation of experiments with the Gerbil [43] system that provides standardized shareable links to the experiments for KGQA systems. We provide the links to the Gerbil experiments, source code, and the experimental data[2] (our experimental data is also shared as an RDF Turtle dataset) as an online appendix. This paper is structured as follows. In the next section, the related work is presented followed by Sect. 3 which introduces our approach in detail. Section 4 highlights the used QA system, QV component, datasets and data preparation process. We describe in Sect. 5 how experiments were processed in general. Section 6 estimates the quality of the query validator as well as the impact of the QV process on the QA quality. Section 7 concludes the paper and outlines future work.

2 Related Work

Techniques that tackle the task of validating the answer were applied mainly in IR-based QA, which we mentioned in Sect. 1. IR-based QA systems are often required to rank huge amounts of candidate answers [26], e.g., the incorrect answer candidates in form of textual paragraphs have to be eliminated by the AV module. In [34], e.g., the AV process is performed on the basis of Expected Answer Type, Named Entities Presence, and Acronym Checking (only if a question is about an acronym). The authors mention that sometimes AV module is "too strict", i.e., it removes also correct answers.

However, the AV and QV processes in KGQA are not well investigated in the research community. Our previous paper [19] describes the novel approach for improving the QA quality where answer candidates are filtered just by evaluating the NL input (i.e., the user's question) and output (i.e., the system's answer), accordingly, it is a system-agnostic approach. Nevertheless, it requires well-formed NL answers that are hard to compute automatically.

On the other hand, there appeared recently some approaches to semantic parsing by treating it as a problem of semantic graph generation and re-ranking [27,31,45,46]. While Yih et al. [45] introduce grounded query graph candidates using a staged heuristic search algorithm and employs a neural ranking model for

[2] https://doi.org/10.6084/m9.figshare.19434515.

scoring and finding the optimal semantic graph, Yu et al. [46] utilize a hierarchical representation of KG predicates in their neural query graph ranking model. A local sub-sequence alignment model with cross-attention is presented in [31]. A slot-matching model to rank query graphs for complex KGQA [27] exploits the inherent structure of query graphs and uses multiple attention scores to explicitly compare each predicate in a query graph with the NL question.

Another topic which has being attracted more and more attention of the research community in recent years is the problem of unanswerable questions [2,20,23,40,44]. However, most of them deal with Machine Reading Comprehension, not KGQA. Unfortunately, different classifications of unanswerable questions (e.g., [2,23,24,44]) consider only the situation in which a (not answered) question has an answer (an answer that is available, but could not be computed by the QA-system, for any reason) and do not investigate the case when there exists no answers to a question, e.g., "What is the capital of Mars?". These two cases differ fundamentally for the field of QA, therefore, they need different approaches to be resolved. However, to distinguish these two cases is not important for this paper. Instead, we focus on deciding whether an answer (represented as query candidate) is correct or not. For this reason, we call all these questions *unanswerable questions*.

3 Approach

Our approach is based on the general assumption that a SPARQL query is expressing a question in a formal representation which can be translated back to an NL text that should be similar to the original question.

In a KGQA system, the generation of a SPARQL query given a question can be considered as a translation from NL to the formal language (cf. Fig. 1). We consider the direct comparison of SPARQL queries and NL questions as very challenging, especially for SPARQL over Wikidata [18] because of non-intuitive URIs naming convention, therefore, we convert SPARQL query candidates to a textual representation. For this purpose, the labels stored in the KG are used to convert all Semantic Web identifiers (e.g., https://www.wikidata.org/entity/Q84) to their textual representation (e.g., "London"). We call the process of generating an NL representation *verbalization of a SPARQL query*.

In our approach we assume that *given a NL question, a KGQA system produces a list of SPARQL query candidates ranked by a relevance score*, computed internally within the system, i.e., the first *query candidate will be used to compute the answer to show to an end-user*. The goal of our QV approach is to ensure that incorrect query candidates are filtered while relying only on the user's NL question and the computed SPARQL query candidates of the considered KGQA system. Hence, our QV process is intended to distinguish query candidates resulting in correct answers from those that would result in incorrect answers. Our approach is system-agnostic and does not require executing the SPARQL queries. In the following subsections, we describe the approach in detail.

```
# What is the cause and place of John Denver's death?
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT ?cause ?place WHERE {
  wd:Q105460 wdt:P509 ?cause .
  wd:Q105460 wdt:P20 ?place .
}
```

Fig. 2. An example of a correct SPARQL query candidate (using Wikidata).

3.1 SPARQL Query Candidates Verbalization

To convert SPARQL query candidates to NL answers, we use a straight-forward process where only the `WHERE` clause of the SPARQL query is considered. All entities and predicates are replaced by their labels, e.g., wd:Q5 is replaced by its English label "human". All variable names are kept as they are, e.g., `?o1, ?subject, ?answer`. It is worth mentioning that any other modifiers (e.g., `LIMIT, ORDER BY, GROUP BY`) in a query are removed in our current approach and do not influence the final NL representation[3]. Finally, all the labels are concatenated with each other in the order of appearance with the space separator.

Considering the SPARQL query presented in Fig. 2, our process computes the following *NL representation*: "`John Denver cause of death ?cause John Denver place of death ?place`". As many property and entity labels used in a question are often mirrored in its verbalization, our approach is based on the assumption that the QV classifier will be capable of determining such query candidate as a correct one (i.e., the user's question and the query verbalization are similar).

3.2 Query Validation Process

The intention of the proposed QV process is as follows: given a query candidates list, we are *aiming at excluding as many incorrect query candidates as possible while not removing the correct ones*. Thus, our approach increases the chances of showing a correct answer to a user of a QA system. In Fig. 3, several different QV cases are shown. All the incorrect query candidates were removed by the QV (e.g., in A') while the correct ones were left untouched (cf. A''). In an extreme case, a query candidates list contains only incorrect items (e.g., in D and E). In this case, the list should become empty after a perfect QV (cf. D' and D''). Likewise, there could be only correct query candidates in the list (not shown in Fig. 3). In this (unlikely) case, at least one query candidate should be recognized as correct by the QV.

[3] Measuring the impact on the verbalization regarding the QV results would be part of additional research.

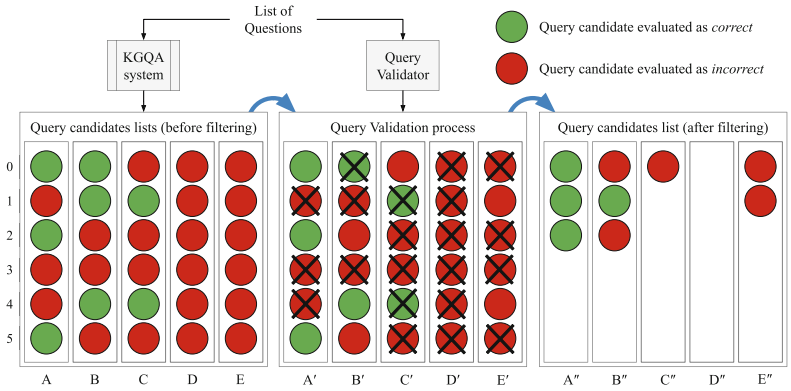

Fig. 3. An example of *query validation process* where a KGQA system proposed 6 ranked query candidates (index: 0–5) for each of the 5 questions (A–E).

3.3 Measures of Query Validation Efficiency

Classification Quality of the Query Validator. To measure the quality of the query validator, we use the following well-known metrics: True Positive Rate (TPR/Recall), True Negative Rate (TNR), Balanced Accuracy (BA), Precision, and F1 score. The metrics are calculated in the machine learning setting for binary classification [33].

Question Answering Quality. To measure the efficiency of QV process, we need to take into account two types of situations that may occur – (S_A) when a query candidate list generated by a KGQA system for a given question contains records with at least one correct query candidate, (S_B) when a query candidate list contains no correct items. In addition, the questions are divided into answerable and unanswerable.

The most common way to measure QA quality is to take the answers generated by the first-ranked query candidate and compare them to the gold standard answers. In this regard, to measure the efficiency of our QV approach, we use well-known Precision, Recall, and F1 score metrics calculated in the information-retrieval setting [37]. The metrics are calculated on the answers obtained before and after QV process. Thereafter, relative improvement is computed.

As our approach influences the whole query candidate list, the other metrics that take into account ranking order have to be considered. Therefore, we utilize Precision@k and NDCG@k [37], where $k \in [1, n]$ and n is the number of query candidates provided by the considered KGQA system.

It is reasonable to use the aforementioned metrics only w.r.t. situation S_A (mentioned at the beginning of this paragraph). While considering situation S_B, we propose the new metric *Answer Trustworthiness Score* (formally defined in Eq. 1 in Sect. 6.2). This metric "gives a reward" (bonus) when a correct answer is shown to a user, and "punishes" (penalty) the score otherwise. In addition, there

is a special case when the query candidate list is an empty set, although a correct answer would be available within the considered data. In this situation, the metric does not reward or punish the score, as it is considered to be an "honest" response by the QA system to provide a "don't know the answer" statement (and not "guess" an answer). However, if after the QV process the empty answer is presented to a user (instead of "guessing" an answer), the score will be higher as no punishment will be done. The intuition behind this metric is that *no answer is better than a wrong answer*[4]. Other relevant statistics should be calculated over all query candidate sets, such as average position of correct/incorrect query candidates, the average number of correct/incorrect query candidates in a list.

Finally, unanswerable questions have to be considered. For this class of questions, the expected response is defined as an empty query candidates list.

4 Material and Methods

To validate our approach, we used the state-of-the-art QA system QAnswer [12,13,15] and two well-known datasets – RuBQ 2.0 [35] and LC-QuAD 2.0 [17].

4.1 The KGQA System QAnswer

Out of many existing QA systems (e.g., DeepPavlov [10], Platypus [32], DeepgAnswer [25] etc.), we have chosen QAnswer because of its portability, accessibility [12] and its following features: robustness, multilingualism, support for multiple KGs (including Wikidata), and it provides high precision and recall [13]. QAnswer also provides an API to ask a question and receive the corresponding ranked query candidate list (of a maximum of 60 candidates). The limitations of QAnswer as described in [13] are not essential to this paper.

4.2 Datasets Overview

QA over KGs is a substantial task that matches a user's question to a query over a KG to retrieve the correct answer [38]. After several updates of the DBpedia [3] KG, many well-known datasets (QALD [42], LC-QuAD 1.0 [41], SimpleDBpediaQA [4] etc.) cannot be utilized on its latest version because a QA system compiled for the inundated version has stopped returning valid requests. Moreover, not all datasets (e.g., CSQA [36]) employ SPARQL as a formal representation (which is a requirement for our work). Some datasets (e.g., VANiLLa [7]) have a structure that does not enable to retrieve an answer without ambiguity. Therefore, it was decided to use the RuBQ 2.0 [35] and LC-QuAD 2.0 [17] datasets for our purpose on this step of our research.

[4] Example: Assuming a user asks for the red or green wire to be cut for defusing a bomb, then a guessed answer by the QA system might have a devastating result in real life.

RuBQ 2.0 Dataset. RuBQ 2.0 is the first Russian dataset for QA over Wikidata that consists of 2,910 questions of varying complexity, their machine translations into English without any post-editing, and annotated SPARQL queries, which are essential for our approach. Here, we only use the English questions. There are 2,357 unique entities, namely 1,218 in questions and 1,250 in answers, as well as 242 unique relations in the dataset. RuBQ 2.0 is split into development (580) and test (2,330) subsets in such a way to keep a similar distribution of query types in both subsets. 510 RuBQ 2.0 questions are unanswerable, which is a new challenge for KGQA systems to make the task more realistic. The fact that RuBQ 2.0 contains unanswerable questions was a strong incentive to use them in our evaluation.

We were not able to utilize the RuBQ 2.0 data split to dev/test parts, as the number of dev samples is too small for fine-tuning our Query Validator. To obtain the new split, we joined both parts and divided the entire dataset into new train/test parts in 80/20 split (see Sect. 4.3).

LC-QuAD 2.0 Dataset. LC-QuAD 2.0 (2nd instance of the Large-Scale Complex Question Answering Dataset) with 30,000 questions, their paraphrases, and their corresponding SPARQL queries is compatible with both Wikidata and DBpedia 2018 KGs. This dataset has a good variety and reasonable complexity levels for questions (e.g., multi-fact questions, temporal questions, and questions that utilize qualifier information). This dataset consists of 21,258 unique entities and 1,310 unique relations. LC-QuAD 2.0 contains 10 different types of questions (such as boolean, dual intentions, fact with qualifiers, and others) spread over 22 unique templates.

4.3 Data Preparation Process

The process of data preparation is analogous for all datasets considered. It consists of the following steps: (1) processing the questions with the KGQA system in order to get query candidate lists for each of them, (2) executing SPARQL queries from the candidate lists on Wikidata in order to get the answer sets, (3) comparing the answer sets from the query candidates with the "gold standard" answer sets from the dataset in order to determine whether a query candidate is correct or not[5], (4) transforming query candidates to NL (according to Sect. 3.1). The sample entry of a resulting dataset in the RDF Turtle format[6] is presented in Fig. 4. The dataset is available in the online appendix.

We summarized the information on the prepared data and divided it into 3 groups (cf. Table 1).

[5] A query candidate is considered as correct if $F1\ score(y_{pred}, y_{true}) = 1$, where y_{pred} – is the set of answers obtained with query candidate and y_{true} is the "gold standard" answer set.

[6] https://www.w3.org/TR/turtle/.

```
fqaqc:dataset-experiment:DATASET:TRAIN:19719-0
    rdf:type fqaac:AnswerCandidate ;
    fqaqc:relatedTo <urn:benchmark:qa:DATASET:TRAIN:19719> ;
    fqaqc:hasPositionBeforeFiltering "0"^^xsd:nonNegativeInteger ;
    fqaqc:hasSPARQL "SELECT * WHERE ?s ?p '0' "^^xsd:string ;
    fqaqc:hasSPARQLResult "{\"head\":{}, \"results\":{}}"^^xsd:string ;
    fqaqc:qaF1Score "0.6"^^xsd:double ; # F1 score(goldStd, sparqlRes)
    fqaqc:hasNaturalLanguageRepresentation "sparql2text"^^xsd:string .
```

Fig. 4. The sample example from the prepared dataset in RDF Turtle format. Where `fqaac` is a local RDF `PREFIX`.

```
fqaac:DATASET:19719-0 fqaac:confidenceScore "0.95"^^xsd:double .
```

Fig. 5. Example of RDF Turtle representation of the query validator output.

Table 1. The statistics of the prepared datasets. The training subset is used only for training and validation of the QV and is not considered in this table. The testing subset is used only for the KGQA system evaluation. AQ – answerable questions in the dataset, uAQ – unanswerable questions in the dataset, Group A – questions where a correct query candidate is at the first position, Group B – questions where a correct query candidate is not at the first position, Group C – questions with zero correct query candidates. QC = ∅ – questions that were resolved with an empty query candidates list (not included in Group C).

Dataset	# AQ	# uAQ	# Group A	# Group B	# Group C	# QC = ∅
RuBQ 2.0	480	102	78	85	419	0
LC-QuAD 2.0	6001	0	958	1288	3733	22

4.4 BERT-Based Query Validator

For the QV classifier, we used the BERT model [11]. As the pre-training process of this model included *next sentence prediction task* (NSP)[7] [11], we intentionally fine-tune BERT using the same setting. While providing a question text as a first sentence and a NL representation of a query candidate as the next one, we follow the assumption that a question and a text representation form a valid pair. To create the training data for the QV classifier, we used the prepared data as follows: to each of the textual questions from the training subset, we assigned one NL representation of a randomly picked query candidate. As the output of the QV is a real value p, such as $\{p \in \mathbb{R} \mid 0 \leq p \leq 1\}$, we *empirically define a threshold for assigning a particular class label*. The target classification label $T = \{t^-, t^+\}$ equals t^+ if and only if the assigned query candidate is considered

[7] BERT was consciously trained to understand relationships between two consecutive sentences if the second sentence follows the first one (e.g., "[CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]) because many important downstream tasks such as QA and Natural Language Inference (NLI) are based on understanding the relationship between two sentences" [11].

as correct to a given question, otherwise, the target label equals t^-. The output of the classifier is represented in the RDF Turtle data as demonstrated in Fig. 5 on page 9.

5 Experimental Setup

We conduct our experiments as follows. In *the first step*, the Query Validators QV are trained separately on the corresponding training subsets D_i, where $D = \{\text{LC-QuAD}, \text{RuBQ}\}$, $D_i \in D$[8]. Hence, the *a single input of the query validator* QV_{D_i} is a pair (q, a), where $q \in D_i$ is the question text and a is the query candidate transformed to NL. To identify a specific question, we will use $q_{D_i,k}$, where k is the ID of the question in the dataset D_i (e.g., a question with `id=1` of the RuBQ 2.0 dataset will be represented by $q_{\text{RuBQ},1}$).

The output of the *target label of the query validator* is $T = \{t^-, t^+\}$, where t^- corresponds to the incorrect (q, a) pair (i.e., a is incorrect for q), and t^+ corresponds to the correct (q, a) pair. We used a balanced distribution (i.e., t^-: 50% to t^+: 50%). Table 2 presents the training data statistics of our QV models. The training data is split into two sets for training and validation (67%/33%) of the query validator.

In *the second step*, we apply the QV_{D_i} to the outputs of the QAnswer system. The outputs have been produced by feeding the questions from the test subset of D_i to the KGQA system (see Step (1) in Sect. 4.3). Thus, the output represents a ranked query candidate list $L_{D_i,q}$ produced for a given question q. L_{D_i} is the set of the query candidate lists for all $q \in D_i$ (i.e., L_{RuBQ} is referring to all candidate lists from questions of the RuBQ 2.0 dataset). $L_{\text{RuBQ},q}$ is a specific query candidate list of the question $q \in D_{\text{RuBQ}}$. Consequently, $L_{\text{RuBQ},q,1}$ is the query candidate at position 1 from $L_{\text{RuBQ},q}$), where $0 \leq |L_{D_i,q}| < n$ (where n is the maximum number of available candidates in a list of query candidates) and $|L_{D_i}| = |D_i|$.

After applying the particular QVs (QV_{RuBQ} and $QV_{\text{LC-QuAD}}$) to L_{RuBQ} and $L_{\text{LC-QuAD}}$ respectively, we obtain new filtered lists of query candidates (i.e., $\hat{L}_{\text{RuBQ}}$ and $\hat{L}_{\text{LC-QuAD}}$). Hence, if the prediction of QV_{D_i} was t^-, then a query candidate is eliminated from $L_{D_i,q}$. As we track the experimental data using RDF, we are capable of obtaining such information as: position before filtering (`fqaac:hasPositionBeforeFiltering`), is correct (`fqaac:qaF1Score = 1`) for each $L_{D_i,q}$. Therefore, we calculate a set of metrics for QA quality as proposed in Sect. 3.3.

6 Evaluation and Analysis

In this section, we describe the evaluation w.r.t. the two steps described in Sect. 5. First, we evaluated the quality of the QV itself (i.e., binary classification quality). Secondly, we evaluated the impact of the QV process on the QA quality.

[8] The trained Query Validators are available online;
LC-QuAD: https://huggingface.co/perevalov/query-validation-lcquad,
RuBQ: https://huggingface.co/perevalov/query-validation-rubq.

Table 2. Quality metrics of the trained dataset-specific Query Validators.

Query Validator	$\lvert t^- \rvert$	$\lvert t^+ \rvert$	TPR (Recall)	TNR	BA	Precision	F1 score
QV_{RuBQ}	9040	9040	0.9805	0.8968	0.9386	0.8874	0.9316
$QV_{\text{LC-QuAD}}$	24045	24045	0.9846	0.9854	0.9850	0.9854	0.9849

Table 3. Evaluation of QA quality improvement using the Gerbil system.

	Micro			Macro		
	Precision	Recall	F1 score	Precision	Recall	F1 score
RuBQ 2.0						
Before QV_{RuBQ}	0.0456	0.4910	0.0834	0.4531	0.4469	0.4462
After QV_{RuBQ}	0.1389	0.5000	0.2174	0.4594	0.4562	0.4505
Improvement in %	204.61	1.83	160.67	1.39	2.08	0.96
LC-QuAD 2.0						
Before $QV_{\text{LC-QuAD}}$	0.1621	0.2984	0.2100	0.5094	0.5191	0.4982
After $QV_{\text{LC-QuAD}}$	0.3561	0.3679	0.3619	0.5341	0.5495	0.5247
Improvement in %	119.68	23.29	72.33	4.85	5.86	5.32

6.1 Answer Validation Classifier Evaluation

In our task *the importance of a false negative prediction is higher than a false positive one*. If the only one correct query candidate from $L_{D_i,q}$ is eliminated by the false negative error, it will inevitably lead to 0% of QA quality, which is not the case for false positive errors. The results regarding the quality of the QV are presented in Table 2. With these results we prove that it is possible to obtain comparable classification quality w.r.t. the well-formed query candidate verbalizations[9]. The obtained Recall score shows that the classifier is capable of avoiding many false negative predictions, which matches our requirements. Thus, even by using our straight-forward query candidate verbalization method (cf. Sect. 3.1), the trained QV models can distinguish between correct and incorrect (q, a) pairs.

6.2 Question Answering Quality Improvement

In the following, the complete evaluation process is described. We applied our QVs to the outputs of the QAnswer system to remove incorrect query candidates (cf. Table 1). In the following paragraphs, by the term *before the QV*, we imply the original results from the QAnswer system (L_{D_i}). The term *after QV* implies the results after applying the QV ($\hat{L}_{D_i}$).

[9] In our previous study, we already compared QV's quality using different query candidate verbalization methods [19].

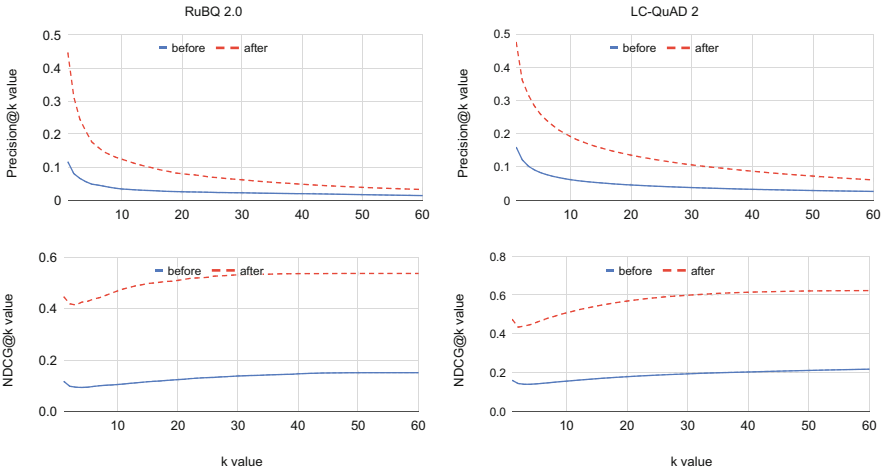

Fig. 6. Precision@k, NDCG@k before and after Query Validation process

Improvement of General Question Answering Quality. The standard metrics for QA systems are based on the computed results of the first element (i.e., in our evaluation, just the first element of QAnswer's answer candidate list is used). We used Gerbil[10] [29,43] for calculating automatically Micro/Macro Precision, Recall and F1 score from our computed data in a comparable and citable form. As the input of the evaluation, the subset of questions was used where QAnswer was computing at least one correct answer candidate for a question $q \in$ Group A $\cup$ Group B (cf. Table 1). The results in Table 3[11] show an improvement of up to 204.61% (micro precision improvement w.r.t. RuBQ 2.0) while the overall macro quality was improved for both datasets. Hence, our approach is capable of improving the answer quality of QA systems w.r.t. the given metrics.

Improvement w.r.t. Groups A and B. In this paragraph, we demonstrate the impact of our approach w.r.t. the whole list of query candidates. To do this, we also selected $L_{D_i,q}$ such that they contain at least one correct query candidate (i.e., Group A $\cup$ Group B). Thereafter, we measured Precision@k and NDCG@k, where $k \in [1, 60]$ before and after the QV process. Hence, we show the impact of our approach w.r.t. the different sizes of $L_{D_i,q}$ and not considering only the first query candidate (i.e., $L_{D_i,q,1}$) as done in the previous paragraph. The results of calculations are presented in Fig. 6. In the charts, we recognize significant improvement w.r.t. all k values (e.g., 382% w.r.t. Precision@1 (=NDCG@1) on

[10] http://gerbil-qa.aksw.org/gerbil/, version 0.2.3.

[11] Our results are available online. LC-QuAD 2.0: http://gerbil-qa.aksw.org/gerbil/experiment?id=202112080001 and http://gerbil-qa.aksw.org/gerbil/experiment?id=202112080002; RuBQ 2.0: http://gerbil-qa.aksw.org/gerbil/experiment?id=202112090005 and http://gerbil-qa.aksw.org/gerbil/experiment?id=202112090006.

RuBQ 2.0 and 297% on LC-QuAD 2.0 respectively). This tendency is discovered on both datasets and metrics, thus, showing that the proposed approach is not a specialized solution for a particular setting. However, this experiment does not show the impact of our method on the query candidate lists provided by QAnswer that do not contain correct candidates (i.e., Group C). In the next subsection, we discuss this aspect in detail.

Improvement w.r.t. Group C. In this paper, we follow the assumption that *no answer is better than a wrong answer* (cf. Sect. 3.3). It is driven by the observation that an incorrect answer might confuse users who would often not be able to decide if the answer is correct or incorrect. Thus, using our method, we highlight the possibility that all incorrect query candidates of a question can be removed from $L_{D_i,q}$. If instead of an incorrect answer produced by a query candidate, a system should provide an empty answer (i.e., the QA system does not "guess" but explicitly expresses the missing capabilities to answer the given question), this will lead to an improved QA quality from users' perspective. The "standard" QA metrics (e.g., Precision, NDCG) are not reflecting this behavior. For example, a QA system that would provide 50% correct answers and 50% ("guessed") incorrect answers could have the same scores as a system with 50% correct results and 50% no answer (i.e., "don't know the answer") – which is not a valid approach from our point of view. To tackle this scenario, we defined the novel metric *Answer Trustworthiness Score* (ATS) that takes into account our initial assumption. If no such metric is used, "guessing an answer" by a QA system will statistically improve the QA quality. Therefore, we define here the metric ATS that in particular takes into account the number of questions that were answered with an empty result:

$$ATS(D_i) = \frac{\sum_{q \in D_i} f(q)}{|D_i|}, \text{ where } f(q) \begin{cases} +1 & \text{if } isCorrect(L_{D_i,q,1}) = True \\ 0 & \text{else if } L_{i,q} = \emptyset \\ -1 & \text{else} \end{cases} \quad (1)$$

where $isCorrect(L_{i,q,1}) = True$ if and only if for a question d the correct answer is shown (i.e., for an empty response no answer is shown). The answer is produced by a query candidate $L_{i,q,1}$. If an unexpected empty result is shown, then the second case is triggered. The proposed metric may show a clear improvement regarding unanswerable questions. We propose to the scientific community to adopt this metric to ensure reasonable QA quality reflection. In addition, we also analyze such statistics as average correct ($\overline{j^+}$) and incorrect ($\overline{j^-}$) query candidate position, average number of correct ($\overline{L^+_{D_i,q}}$) and incorrect ($\overline{L^-_{D_i,q}}$) query candidates in a list. In Table 4 on page 14 we present the metrics for QA quality introduced in Sect. 3.3. The results demonstrate a significant difference between the values before and after QV. The values of the statistics $\overline{j^+}$ and $\overline{j^-}$ demonstrate that the positions of the correct and incorrect query candidates were shifted to the top of the list after QV. The other values of the $\overline{L^+_{i,d}}$ and $\overline{L^-_{i,d}}$ indicate that the numbers of the correct and incorrect query candidates were decreased after QV. These results are ambiguous, however, the metric proposed

Table 4. Question answering metrics before and after answer filtering.

metric	state	RuBQ 2.0	LC-QuAD 2.0
$\overline{j^+}$	Before QV	16.32	19.63
	After QV	9.16	12.93
$\overline{j^-}$	Before QV	21.98	29.74
	After QV	10.28	19.34
$\overline{L^+_{D_i,q}}$	Before QV	3.12	4.27
	After QV	2.55	3.88
$\overline{L^-_{D_i,q}}$	Before QV	43.49	57.74
	After QV	11.95	27.60
$ATS(D_i)$	Before QV	-0.31	-0.75
	After QV	-0.12	-0.71

Table 5. Evaluation of unanswerable questions from RuBQ 2.0 dataset before and after QV

	state	# questions
$L_{\text{RuBQ},q} = \emptyset$ (correct)	Before QV	0
	After QV	50
$L_{\text{RuBQ},q} \neq \emptyset$ (incorrect)	Before QV	102
	After QV	52

by us is able to disambiguate them. Given our novel metric ($ATS(D_i)$), all the values are negative. The metric results after QV were improved from -0.75 to -0.71 for LC-QuAD 2.0 and from -0.31 to -0.12 for RuBQ 2.0 dataset. Negative values signify that a QA system gives more incorrect answers rather than correct ones. Thus, the QV process decreased the number of incorrect and increased the number of correct query candidates, respectively. Hence, the trustworthiness of the considered system w.r.t. the two analyzed datasets is not good, i.e., users need to evaluate the results carefully.

Improvement w.r.t. Unanswerable Questions. The *unanswerable questions were intentionally integrated by authors of the RuBQ 2.0 dataset*. In this paragraph, we utilize this subset of questions to see if the proposed approach improves the performance of the KGQA system regarding unanswerable questions. *The correct response to an unanswerable question* $q \in D_i$ is $L_{D_i,q} = \emptyset$, *otherwise, the response is incorrect*. Such evaluation strategy is also supported in the original paper [24] of the RuBQ 2.0 authors. Our evaluation results of the RuBQ 2.0 dataset regarding the contained 102 unanswerable questions are shown in Table 5. As the QAnswer system's strategy is to generate a list of query candidates for any kind of question, none of the results were considered as correct before QV. After QV, all the query candidates from the respective 50

lists $L_{\text{RuBQ},q}$ were completely eliminated, and hence 50 unanswerable questions would have been answered correctly (with an empty answer).

6.3 Discussion and Limitations

We raise several questions for the discussion. Firstly, the validity of our approach is strongly depending on the labels provided by the considered KG. For the given datasets, it works surprisingly well, however, for a real-world scenario, additional methods would be required (e.g., integrating synonyms from [28]). Futhermore, this raises the question if current KGQA benchmarks already represent the variety of NL questions well enough or would require additional extensions. Secondly, in this work, we consider a query candidate as correct if the F1 score of the expected and computed results is 1. In this regard, the other options would be to consider a different threshold instead of 1 (e.g., 0.75 or 0.5). The major point of concern of the strict threshold for the real-valued measure (F1 score) for determining whether a query candidate is correct is that some gold-standard correct answers sets contain more than one item (e.g., question: "Give me all post-punk artists"). In this regard, if a query candidate could produce a query equivalent to "Give me all *German* post-punk artists", the result would be *partially correct*. Hence, the *"is correct" threshold should be adapted according to the tolerance to the partially correct answers* of the evaluation context (i.e., in our case, we have zero tolerance to incorrect answers). Thirdly, this evaluation provided in the work may be biased due to the only one KGQA system was used. Although QAnswer is well-known and used for much research, further work should cover different KGQA systems as their approaches and capabilities may vary significantly. Finally, more training techniques of the query validator should be explored, s.t., the final impact on the QA quality can be increased. At the moment, there are still additional opportunities for the QV process improvement, considering not only answerable but also unanswerable questions.

7 Conclusions

In this paper, we have proven the impact of our query candidate validation approach. It uses a NL representation of the SPARQL query that is compared by a trained model with the given question. Our approach takes into account the verbalized information of concepts, predicates, and instances that are already defined in a SPARQL query candidate. Hence, we did not create a complete nor well-formed NL representation of the SPARQL queries. However, the results of our research show significant QA quality improvements w.r.t. different aspects that are important for QA systems. In particular, our evaluation includes answerable and unanswerable questions as well as it shows that the quality of the query candidate list can be improved.

As we have shown, our method is capable of improving the quality of QA systems without knowledge about the implemented approach of a QA system. Consequently, it might be integrated in the query builder component of QA

systems, or as a reusable component via the QA framework (e.g., the Qanary framework [8]) to improve the quality of answers or intermediate candidate lists. Hence, our main contribution is providing a domain-agnostic method that can be applied to any knowledge base that provides verbalization (typically available as predicate `rdfs:label`), s.t., corresponding KGQA systems increase their quality.

Looking forward, we plan to use different models, verify the presented approach on different systems and benchmarks and check the applicability of the approach to other languages.

References

1. Abdiansah, A., Azhari, A., Sari, A.K.: Survey on answer validation for Indonesian question answering system (IQAS). Int. J. Intell. Syst. Appl. **10**, 68–78 (2018). https://doi.org/10.5815/ijisa.2018.04.08
2. Asai, A., Choi, E.: Challenges in information seeking QA: unanswerable questions and paragraph retrieval. arXiv preprint arXiv:2010.11915 (2020)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC-2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
4. Azmy, M., Shi, P., Lin, J., Ilyas, I.: Farewell freebase: migrating the simplequestions dataset to DBpedia. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 2093–2103 (2018)
5. Babych, S., Henn, A., Pawellek, J., Padó, S.: Dependency-based answer validation for German. In: Petras, V., Forner, P., Clough, P.D. (eds.) CLEF 2011 Labs and Workshop, Notebook Papers, 19–22 September 2011, Amsterdam, The Netherlands. CEUR Workshop Proceedings, vol. 1177. CEUR-WS.org (2011)
6. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Sci. Am. **284**(5), 34–43 (2001)
7. Biswas, D., Dubey, M., Rony, M.R.A.H., Lehmann, J.: VANiLLa: verbalized answers in natural language at large scale. CoRR abs/2105.11407 (2021)
8. Both, A., Diefenbach, D., Singh, K., Shekarpour, S., Cherix, D., Lange, C.: Qanary – a methodology for vocabulary-driven open question answering systems. In: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 625–641. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-34129-3_38
9. Both, A., Gashkov, A., Eltsova, M.: Similarity detection of natural-language questions and answers using the VANiLLa dataset. J. Phys: Conf. Ser. **1886**(1), 012017 (2021). https://doi.org/10.1088/1742-6596/1886/1/012017
10. Burtsev, M., et al.: DeepPavlov: open-source library for dialogue systems. In: Proceedings of ACL 2018, System Demonstrations, pp. 122–127. Association for Computational Linguistics, Melbourne (2018)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long and Short Papers), vol. 1, pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019). https://doi.org/10.18653/v1/N19-1423

12. Diefenbach, D., Both, A., Singh, K., Maret, P.: Towards a question answering system over the semantic web. Semantic Web **11**, 421–439 (2020)
13. Diefenbach, D., Giménez-García, J., Both, A., Singh, K., Maret, P.: QAnswer KG: designing a portable question answering system over RDF data. In: Harth, A., Kirrane, S., Ngonga Ngomo, A.-C., Paulheim, H., Rula, A., Gentile, A.L., Haase, P., Cochez, M. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 429–445. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_25
14. Diefenbach, D., Lopez, V., Singh, K., Maret, P.: Core techniques of question answering systems over knowledge bases: a survey. Knowl. Inf. Syst. **55**(3), 529–569 (2017). https://doi.org/10.1007/s10115-017-1100-y
15. 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: Liu, L., et al. (eds.) The World Wide Web Conference, WWW 2019, San Francisco, May 13–17, 2019, pp. 3507–3510. ACM (2019). https://doi.org/10.1145/3308558.3314124
16. Dimitrakis, E., Sgontzos, K., Tzitzikas, Y.: A survey on question answering systems over linked data and documents. J. Intell. Inf. Syst. **55**(2), 233–259 (2019). https://doi.org/10.1007/s10844-019-00584-7
17. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: LC-QuAD 2.0: a large dataset for complex question answering over Wikidata and DBpedia. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) ISWC 2019. LNCS, vol. 11779, pp. 69–78. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_5
18. Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., Vrandečić, D.: Introducing Wikidata to the linked data web. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 50–65. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_4
19. Gashkov, A., Perevalov, A., Eltsova, M., Both, A.: Improving the question answering quality using answer candidate filtering based on natural-language features. In: 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2021) (2021)
20. Godin, F., Kumar, A., Mittal, A.: Learning when not to answer: a ternary reward structure for reinforcement learning based question answering. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Industry Papers), Vol. 2, pp. 122–129. Association for Computational Linguistics, Minneapolis (2019). https://doi.org/10.18653/v1/N19-2016
21. Gómez-Adorno, H., Pinto, D., Vilariño, D.: A question answering system for reading comprehension tests. In: Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Rodríguez, J.S., di Baja, G.S. (eds.) MCPR 2013. LNCS, vol. 7914, pp. 354–363. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38989-4_36
22. Grappy, A., Grau, B., Falco, M., Ligozat, A., Robba, I., Vilnat, A.: Selecting answers to questions from web documents by a robust validation process. In: 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, vol. 1, pp. 55–62 (2011). https://doi.org/10.1109/WI-IAT.2011.210
23. Hu, M., Wei, F., Peng, Y., Huang, Z., Yang, N., Li, D.: Read+verify: machine reading comprehension with unanswerable questions. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6529–6537 (2019)

24. Korablinov, V., Braslavski, P.: RuBQ: a Russian dataset for question answering over Wikidata. In: Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (eds.) ISWC 2020. LNCS, vol. 12507, pp. 97–110. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62466-8_7
25. Lin, Y., Zhang, M., Zhang, R., Zou, L.: Deep-gAnswer: a knowledge based question answering system. In: U, L.H., Spaniol, M., Sakurai, Y., Chen, J. (eds.) APWeb-WAIM 2021. LNCS, vol. 12859, pp. 434–439. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85899-5_33
26. Magnini, B., Negri, M., Prevete, R., Tanev, H.: Is it the right answer? Exploiting web redundancy for answer validation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 425–432. Association for Computational Linguistics, Philadelphia (2002). https://doi.org/10.3115/1073083.1073154
27. Maheshwari, G., Trivedi, P., Lukovnikov, D., Chakraborty, N., Fischer, A., Lehmann, J.: Learning to rank query graphs for complex question answering over knowledge graphs. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) ISWC 2019. LNCS, vol. 11778, pp. 487–504. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30793-6_28
28. Miller, G.A.: WordNet: An Electronic Lexical Database. MIT Press (1998)
29. Napolitano, G., Usbeck, R., Ngomo, A.-C.N.: The scalable question answering over linked data (SQA) challenge 2018. In: Buscaldi, D., Gangemi, A., Reforgiato Recupero, D. (eds.) SemWebEval 2018. CCIS, vol. 927, pp. 69–75. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00072-1_6
30. Pakray, P., Barman, U., Bandyopadhyay, S., Gelbukh, A.: Semantic answer validation using universal networking language. Int. J. Comput. Sci. Inf. Technol. **3**(4), 4927–4932 (2012)
31. Parikh, A.P., Täckström, O., Das, D., Uszkoreit, J.: A decomposable attention model for natural language inference. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 2249–2255. Association for Computational Linguistics (2016)
32. Pellissier Tanon, T., de Assunção, M.D., Caron, E., Suchanek, F.M.: Demoing Platypus – a multilingual question answering platform for Wikidata. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 11155, pp. 111–116. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98192-5_21
33. Powers, D.M.W.: Evaluation: from precision, recall and F-factor to ROC, informedness, markedness & correlation. J. Mach. Learn. Technol. **2**(1), 37–63 (2011)
34. Rodrigo, A., Pérez-Iglesias, J., Peñas, A., Garrido, G., Araujo, L.: A question answering system based on information retrieval and validation. In: CLEF 2010 LABs and Workshops, Notebook Papers (2010)
35. Rybin, I., Korablinov, V., Efimov, P., Braslavski, P.: RuBQ 2.0: an innovated Russian question answering dataset. In: Verborgh, R., Hose, K., Paulheim, H., Champin, P.-A., Maleshkova, M., Corcho, O., Ristoski, P., Alam, M. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 532–547. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_32
36. Saha, A., Pahuja, V., Khapra, M.M., Sankaranarayanan, K., Chandar, S.: Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
37. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)

38. Singh, K., et al.: Why reinvent the wheel: let's build question answering systems together. In: Proceedings of the 2018 World Wide Web Conference, pp. 1247–1256 (2018)
39. Solovyev, A.: Dependency-based algorithms for answer validation task in Russian question answering. In: Gurevych, I., Biemann, C., Zesch, T. (eds.) GSCL 2013. LNCS (LNAI), vol. 8105, pp. 199–212. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40722-2_20
40. Tan, C., Wei, F., Zhou, Q., Yang, N., Lv, W., Zhou, M.: I know there is no answer: modeling answer validation for machine reading comprehension. In: Zhang, M., Ng, V., Zhao, D., Li, S., Zan, H. (eds.) NLPCC 2018. LNCS (LNAI), vol. 11108, pp. 85–97. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99495-6_8
41. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: LC-QuAD: a corpus for complex question answering over knowledge graphs. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 210–218. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_22
42. Usbeck, R., Gusmita, R.H., Ngomo, A.N., Saleem, M.: 9th challenge on question answering over linked data (QALD-9). In: Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, 8th–9th October 2018, pp. 58–64 (2018)
43. Usbeck, R., et al.: GERBIL - general entity annotation benchmark framework. In: 24th WWW Conference (2015)
44. Yen, A.Z., Huang, H.H., Chen, H.H.: Unanswerable question correction in question answering over personal knowledge base. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 14266–14275 (2021)
45. Yih, S.W., Chang, M.W., He, X., Gao, J.: Semantic parsing via staged query graph generation: question answering with knowledge base. In: Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP (2015)
46. Yu, M., Yin, W., Hasan, K.S., Santos, C.D., Xiang, B., Zhou, B.: Improved neural relation detection for knowledge base question answering. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Long Papers), Vol. 1, pp. 1321–1331. Association for Computational Linguistics (2017)
47. Zamanov, I., Kraeva, M., Hateva, N., Yovcheva, I., Nikolova, I., Angelova, G.: Voltron: a hybrid system for answer validation based on lexical and distance features. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). pp. 242–246. Association for Computational Linguistics, Denver (2015). https://doi.org/10.18653/v1/S15-2043