Improving the Question Answering Quality using Answer Candidate Filtering based on Natural-Language Features

1st Aleksandr Gashkov*
Humanitarian Faculty
Perm National Research Polytechnic University
Perm, Russia
gashkov@dom.raid.ru

2nd Aleksandr Perevalov*
Computer Science and Languages
Anhalt University of Applied Sciences
Köthen (Anhalt), Germany
aleksandr.perevalov@hs-anhalt.de

3rd Maria Eltsova
Humanitarian Faculty
Perm National Research Polytechnic University
Perm, Russia
maria_eltsova@mail.ru

4th Andreas Both
Computer Science and Languages
Anhalt University of Applied Sciences
Köthen (Anhalt), Germany
andreas.both@hs-anhalt.de

Technology Innovation Unit
DATEV eG
Nuremberg, Germany
andreas.both@datev.de

* corresponding authors

***Abstract*—Software with natural-language user interfaces has an ever-increasing importance. However, the quality of the included Question Answering (QA) functionality is still not sufficient regarding the number of questions that are answered correctly.**

In our work, we address the research problem of how the QA quality of a given system can be improved just by evaluating the natural-language input (i.e., the user's question) and output (i.e., the system's answer).

Our main contribution is an approach capable of identifying wrong answers provided by a QA system. Hence, filtering incorrect answers from a list of answer candidates is leading to a highly improved QA quality. In particular, our approach has shown its potential while removing in many cases the majority of incorrect answers, which increases the QA quality significantly in comparison to the non-filtered output of a system.

***Index Terms*—question answering, answer validation, answer filtering, answer ranking, improving question answering quality, natural language processing, English language**

I. Introduction

Question Answering (QA) aims to provide precise answers to questions formulated in a natural language (NL). There are many different approaches for implementing QA systems (e.g., AskNow [21], QAnswer [22], DrQA [23]) focusing on specific paradigms and technologies. The two main directions of QA are Open Domain Question Answering (ODQA) and Knowledge Graph Question Answering (KGQA).

The common data flow in QA systems is from a given input (question) to an output (answer). Typically, both are given using NL, s.t., the accessibility of such a system for humans is high. In this work, we tackle the problem of answer validation (AV) considering a KGQA system as a black box. Thus, the concrete approach for computing the answer is hidden (i.e., not visible and changeable) and only textual representation of question and answer are provided for further analyses.

Here, we propose an approach for improving QA quality by filtering out incorrect answer candidates. The approach is built on the assumption that a considered KGQA system provides a list of query candidates (e.g., written in SPARQL) that will be used to retrieve the answers from a knowledge graph (KG). In this scenario, the provided query candidates are executed and, therefore, it is decided whether they will retrieve the correct answer or not (i.e., the incorrect answer candidates should be eliminated) – we call this process answer validation (AV) or filtering. Consequently, each query candidate must be transformed to an answer candidate in NL (i.e., verbalized), in this regard, our AV experiments are performed on 3 levels (cf., Figure 1) where the verbalizations of query candidates (i.e., NL form of a query) are (A_1) provided using well-formed NL (written by a human as a baseline), (A_2) computed using Natural Language Generation (NLG) considering the contained facts, and (A_3) computed using a bag-of-labels approach of available entities.

In this paper, we follow our long-term research agenda of improving the overall quality of KGQA systems following a domain-agnostic approach that is not limited to just a single class of KGQA systems. Therefore, while having limited access to internal data structures of a KGQA system, the NL form of the questions and answers gains importance. To show the significance of our approach, we not only consider AV module quality (i.e., F1 Score) but also its impact on the end-to-end QA quality (i.e., Precision@k, NDCG@k).

In this paper, we address the following research questions considering the task of filtering NL answer candidates:

RQ$_1$ Is it possible to improve the QA quality while filtering answers just by their NL representation?

RQ$_2$ What QA quality is achievable while filtering the well-formed NL representations of the answer candidates

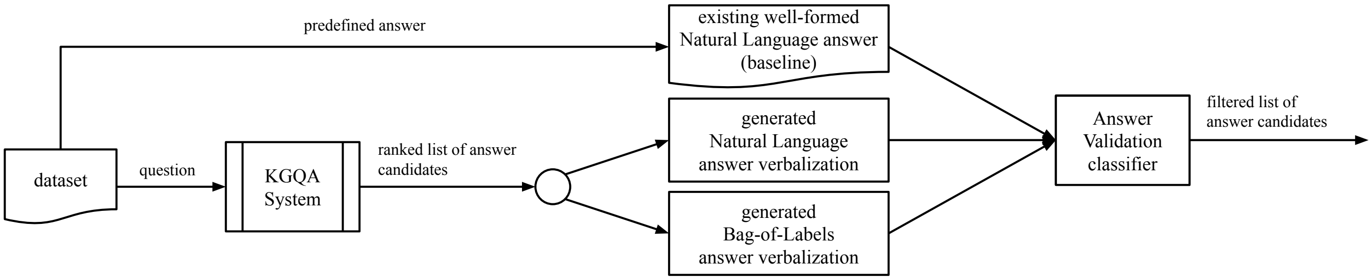

Fig. 1. Overview of the general research idea

TABLE I
ANSWER VALIDATION PUBLICATIONS OF THE LAST DECADE [1]

Ref.	Year	Languages	Datasets	Methods	Evaluation score
[2]	2010	English, Spanish	ResPubliQA [3]	EAT, NER, Acronym Checking	65% Accuracy (English) 57% Accuracy (Spanish)
[4]	2010	Spanish	CLEF 06 [5]	RTE	53% Accuracy
[6]	2011	German	CLEF 11 [7]	Rule-set	44% Accuracy
[8]	2011	French	Web	Decision Tree Combination	53% MRR
[9]	2012	German	CLEF-QA [10]	LogAnswer Framework	61% Correct Top Rank
[11]	2013	English	AVE 08 [12]	RTE	58% Precision 22% F-Score
[13]	2013	English	CLEF 11 [7]	RTE	45% Precision
[14]	2013	Russian	ROMIP [15]	RTE	70.4% F-Score
[16]	2015	English	Sem-Eval 2015 [17]	Word vectors + Classifier	62.35% Accuracy 46.07% F-Score
[18]	2018	English	SQuAD-T [18]	GloVe + GRU	71.24% F-score
[19]	2019	English	SQuAD 2.0 [20]	RTE + ELMo + Verifier	74.2% F-score

(cf., Approach A_1)?

RQ$_3$ What QA quality is achievable while having automatically generated NL answer candidates (cf., Approach A_2 and Approach A_3)?

We conclude our main contributions as follows:

- We demonstrate and evaluate two answer verbalization techniques (A_2 and A_3) based on provided SPARQL query candidates.
- We propose and validate a system-agnostic approach to filter a set of answer candidates using just its NL representation.
- The experimental results show that the proposed approach significantly improves the QA quality.

This paper is structured as follows: after presenting the related work (Section II), we give an overview of our approach in Section III. Section IV presents the used components and datasets. Our experiments are described in Section V followed by their evaluation (Section VI). Section VII concludes and outlines future work.

II. Related Work

Techniques that tackle the task of validating the answer were applied mainly in ODQA where systems are often required to rank huge amounts of candidate answers [24], e.g., the incorrect answer candidates in form of textual paragraphs have to be eliminated by the AV module. In our work, we also use a textual representation to validate answer candidates while using KGQA systems. Thus, the ODQA field is related to our research questions.

The comprehensive list of the AV approaches was presented in [1]. Based on this we show the most related work in the Answer Validation field of the past decade in Table I.

In [2] Answer Validation is applied to an Information Retrieval system. The validation 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.

Other publications (e.g., [4], [11], [13], [14]) are based on the idea of recognizing the textual entailment (RTE) using the output of several QA systems, universal networking language, lexical similarity, and dependency parsing algorithms accordingly. All mentioned papers used textual datasets (containing a pair of one question and one answer).

Babych et al. [6] utilized a rule-based decision algorithm that includes: syntactic analysis, predicate-argument relation analysis, and semantic relation analysis. In [8] Decision Tree Combinations are used that are based on the extracted textual features. Additionally, the system filters answer candidates (i.e., potential answers to a given question) using Named Entities. The authors of [9] applied the so-called LogAnswer Framework which incorporates case-based reasoning.

[16], [18] and [19] applied supervised learning methods. The first utilized the following features: lexical, n-grams, bad-answer specific, named entities, term-frequency vectors, and word2vec [25]. In contrast, [18], [19] used a deep neural network (DNN) model and only GloVe [26] and ELMo [27] features correspondingly.

To conclude we see how AV approaches changed in time from rule-based to DNNs. Additionally, we recognize that Answer Validation was used mainly in ODQA systems while – to the best of our knowledge – no work related to KGQA exists.

III. Approach

We assume here that a KGQA system provides a ranked list of potential answers; we call them *answer candidates*. The first element of the list is reflecting the answer the QA system would show to the user. However, any answer in the list might be correct (i.e., it is considered to correctly answering the given question), we denote such an answer candidate as *correct answer*). Note that there may be multiple candidate answers in the ranked list that are considered to represent the correct answer. The other answer candidates are representing potential answers to the given question where the system has decided that they are not correctly answering the given question (i.e., they are considered to be *incorrect answers* w.r.t. the given question).

To evaluate our research idea, we utilize a text classification model trained on NL representations of correct and incorrect question-answer pairs, i.e., the goal is to predict if a corresponding answer is correct for a given question using their NL representations (cf., Figure 1).

We use the following three settings to establish experiments where a verbalization of an answer is compared to the corresponding NL question: A_1 a well-formed answer text, A_2 an NL representation of each answer candidate, and A_3 a bag-of-labels representation of each answer candidate. These three approaches allow a distinct validation of the capabilities of our approach while validating it from a "perfect" to a "clumsy" textual answer representation. The AV classifier is trained w.r.t. each verbalization technique, and then the results are compared. The AV classifier is used to filter incorrect answer candidates, s.t., the QA quality before and after the filtering process is measured.

For A_1 a dataset is available (cf., Section IV). For A_2 and A_3 we require an automatically executable process to generate the textual representations.

In the following, we will describe the different approaches.

A. Approach 1 – Well-formed NL Answers

The first approach (A_1) should use well-formed NL answers to reflect a high-quality answer verbalization. For example, a well-formed answer to the question "What was the cause of death of John Kennedy?" can be "John Kennedy was assassinated.".

Therefore, it has high demands w.r.t. the textual answer quality and is hard to automate. For this reason, we will use the existing dataset VANiLLa (cf., Section IV-A) containing well-formed NL answers to establish a baseline for our evaluations.

B. Approach 2 – Automatically generated NL answers

The second approach (A_2) has less strict requirements rather than the first one to enable an automatic approach. We will use an NLG method to generate the answer verbalization. We assume here that a KGQA system (like QAnswer, cf., Section IV-B) provides a ranked list of potential answers in the form of SPARQL queries (i.e., a ranked list of *query candidates*). Hence, the SPARQL queries are usable to generate the verbalizations with existing NLG tools (e.g., Triple2NL, cf., Section IV-C). Regarding the example question (see above), the following SPARQL can be proposed (see Listing 1)[1].

```
# question:
#     What was the cause of death of John Kennedy?
# query:
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?answer WHERE {
    dbr:John_F._Kennedy dbo:deathCause ?answer .
}
# the result is dbr:Assassination_of_John_F._Kennedy
```

Listing 1. Retrieving answer URI via SPARQL from the DBpedia KG.

Providing the query and the result (RDF resource), we expect to receive the textual output from an NLG tool. Given the example, a possible verbalization might be "The John F. Kennedy's death cause is Assassination of John F. Kennedy"[2].

C. Approach 3 – Answer verbalization based on bag-of-labels

The third approach (A_3) manifests a number of labels to represent a simple verbalization of each answer candidate. The labels are retrieved from the resources, properties, and results mentioned in a SPARQL query candidate and its results (cf., the previous subsection). The representation of such an answer is readable, however, it can barely be described as an NL sentence.

For example, considering the example in Listing 1, the following resources are present in the SPARQL query and it's result: `dbr:John_F._Kennedy`, `dbo:deathCause`, and `dbr:Assassination_of_John_F._Kennedy`. After that, all the English labels of each resource are retrieved and concatenated (i.e., a single string for each answer candidate). In this example, the following verbalization of the answer candidate is obtained "John F. Kennedy death cause Assassination of John F. Kennedy".

[1] In this example, the SPARQL query retrieves the correct answer regarding the given question.

[2] The title of `dbr:dbr:Assassination_of_John_F._Kennedy` is "Assassination of John F. Kennedy"

IV. Materials and Methods

In this section, we describe the components used to manifest the experimental environment.

A. The VANiLLa Dataset

Based on our preliminary research, it was decided to use the VANiLLa dataset [28] that contains NL representation for both questions and answers in contrast to many other known datasets (cf., [29][3]). It contains 107,166 examples with an 80% (train) – 20% (test) split of questions and answers in a full textual form. The dataset was built on top of the Wikidata knowledge graph. There are six fields for each instance of the VANiLLa dataset: `question_id`, `question`, `answer`, `answer_sentence`, `question_entity_label`, `question_relation`. Obviously, the dataset represents only the *correct question-answer pairs*, which does not allow us to train the AV classifier properly. To generate the data for the *incorrect question-answer pairs* (i.e., negative sampling), we created a set of the corresponding question-answer pairs by having randomly paired a question and an answer from different records of the VANiLLa dataset. The method is precisely introduced in [30].

B. The KGQA system QAnswer

Our study requires a KGQA system to be utilized in order to prove the approach validity with actual (correct and incorrect) answer candidates. Therefore, we selected the well-known QAnswer system [22] which represents the current state-of-the-art in KGQA and satisfies the requirements for A_2 and A_3. QAnswer provides an API to ask a question and receive the corresponding ranked query candidate list. It supports questions over several knowledge bases including Wikidata. The SPARQL queries are constructed by traversing the KG and discovering how the concepts and relations that are mentioned in the question are arranged. To the best of our knowledge, there is no other KGQA system available that satisfies our requirements and provides a state-of-the-art QA quality. [31] compares the results of DeepPavlov's and QAnswer's top-1 results on RuBQ dataset where QAnswer outperforms DeepPavlov in terms of most metrics.

C. Triple-based NLG using Triple2NL

While following our approach A_2, we require automatically generated NL representations for query candidates. However, there are quite a few state-of-the-art open-source toolkits that introduce an API or a module for programming language.

As the used QA system (QAnswer, cf., the previous subsection) responds with a list of query candidates, a tool matching most of our demands is Triple2NL. It represents an integrated system[4] generating a complete NL representation for RDF and SPARQL. The results (cf., [32], [33]) demonstrate that Triple2NL generates comprehensible texts. Triple2NL's approach is based on a bottom-up process to verbalization and exploits some rules. To create a textual representation, Triple2NL tries to get the label or the name of all triple elements from DBpedia. If no label nor name is found, the last part of URI is taken as an NL representation. This allows us to generate text for any KG by submitting labels instead of URIs.

D. BERT Model as Answer Validation Classifier

Modern text classification methods utilize large sets of unstructured data to pre-train a model. In recent years, transformer-based models are holding a significant part in the whole NLP industry and research community. Such models as ELMo [27], ULM-FIT [34], XLNet [35], and BERT [34] represent the current state-of-the-art. The well-known BERT still stays as the landmark model showing one of the best results in many downstream tasks (cf., [36]).

In this work, we used the `bert-base-cased` model[5] the next sentence prediction (NSP) setting as the input consists of a textual tuple question-answer.

V. Experimental Setup

A. Answer Validation Classifier

The AV classifier model (see Section IV-D) was trained and evaluated according to each answer verbalization approach separately (see Section III). Each experiment was executed ten times using different random seed values for the dataset's shuffle to ensure validity. We used such well-known metrics for classification as Precision, Recall, and F1 Score. Given the obtained metric results, we calculated standard deviation (std) which is denoted as "±" in our results.

The generated dataset consists of the mixed correct and incorrect subsets and contains 66,632 records from the original training subset of VANiLLa (the original test subset was used in the next experiment described in Section V-B). The train/test split was done randomly in every experiment with the corresponding ratio 67%/33%. We evaluated both balanced and unbalanced (50 incorrect to 1 correct tuple) dataset's settings to see the corresponding effect.

B. Evaluation of Answer Validation's impact on Question Answering Quality

To demonstrate the impact of AV on a KGQA system's quality, we used the following approach (see Figure 2). The KGQA system was evaluated on the original test subset of VANiLLa containing 21,360 records, but only 8,955 of them have unique questions. We decided to send each question to KGQA only once as the response is stable (i.e., not changing inside one session). A considered KGQA system (here: QAnswer) processed the test dataset and the obtained ordered lists of query candidates were cached. As the query candidate was presented in a form of SPARQL, we were able

[3] The comparison [29] reveals the lack of datasets with verbalized answers; however, the existing datasets (e.g., VQuAnDa and ParaQA) with NL representation of an answer in the full form are relatively small: only 5000 question-answer pairs.

[4] It provides a REST API, which can be deployed locally, but it is not able to generate text from more than one triple.

[5] Available online at https://huggingface.co/bert-base-cased.

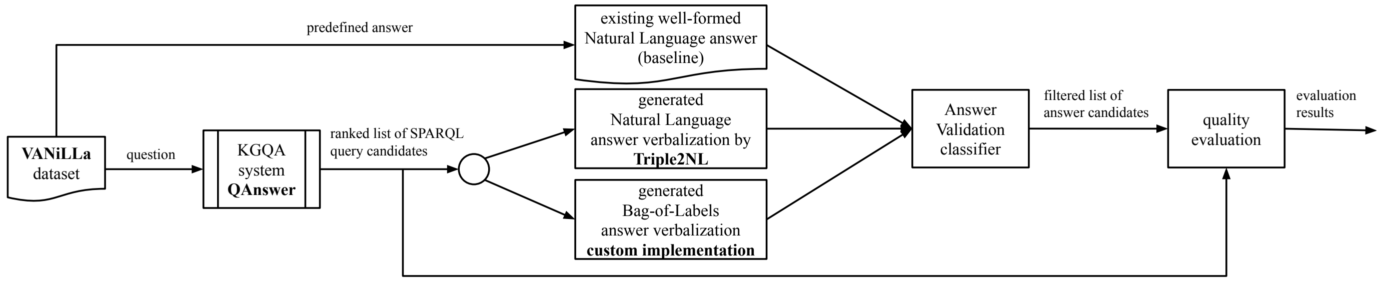
predefined answer
existing well-formed Natural Language answer (baseline)
VANiLLa dataset
question
KGQA system QAnswer
ranked list of SPARQL query candidates
generated Natural Language answer verbalization by Triple2NL
generated Bag-of-Labels answer verbalization custom implementation
Answer Validation classifier
filtered list of answer candidates
quality evaluation
evaluation results

Fig. 2. Flowchart of the QA quality comparison process. The quality is compared between filtered and non-filtered answer candidate sets

to apply A_2 and A_3 to validate the candidates. Using one of the selected verbalization approaches, the answer candidates were being filtered by the AV classifier while the order of candidates stayed the same (i.e., we removed such answer candidates from the list). Finally, the QA quality was measured and compared based on the non-filtered and filtered answer candidate sets. We used well-known metrics such as Precision@k (P@k), Normalized Discounted Cumulative Gain@k (NDCG@k) with $k \in \{1, 5\}$ as QA quality measures [37].

VI. Results and Discussion

A. Experiment 1: High quality NL questions and answers (A_1)

The first experiment is intended to evaluate the capabilities of the AV component on well-formed answer representations which are retrieved from the VANiLLa dataset. The results of the experiment are presented in Table II.

TABLE II
Detailed results for Experiment 1

F1 Score	**Precision**	**Recall**
balanced data (1 to 1)		
0.9968 ± 0.0089	**0.9968** ± 0.0100	**0.9968** ± 0.0098
unbalanced data (50 to 1)		
0.9838 ± 0.0276	0.9918 ± 0.0251	0.9761 ± 0.0454

The first experiment's results are strong and fulfill the quality requirements. They demonstrate that detecting incorrect question-answer pairs using only text is possible, and high quality is achievable. Although, using the unbalanced setting for training negatively affects mostly the recall which is the most important metric for the intended goal. Hence, assuming NL answers of very high quality are provided by a QA system, then our approach should be capable of identifying (and therefore, filtering) incorrect answers.

B. Experiment 2: NLG of limited quality

In this setting, we are evaluating artificially generated NL answers. There are computed automatically from SPARQL queries and the corresponding results.

To our best knowledge, there is no KGQA system available that is providing an API to produce full-fledged NL answers. Consequently, it is required to generate answer verbalization from the available information (i.e., SPARQL query candidates).

Generating artificial answers in a three-step process includes (1) providing a question in textual form to the KGQA system, (2) sending the computed SPARQL query answer candidates to Wikidata, and (3) generating NL representation from the obtained list of the query candidates and Wikidata response.

The first step is straightforward: the question provided to the KGQA system. The system returns a set of query candidates. A typical SPARQL query consists of one or more triples including at least one variable (cf., the following example).

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT DISTINCT ?o2 WHERE {
    ?s1  ?p1  wd:Q57747377 .
    ?s1  wdt:P21 ?o2 .
}  LIMIT 1000
```

Listing 2. An example SPARQL query over Wikidata.

The second step extracts all variables' values from the query's result set as all triples' positions must be filled to generate a text. Hence, we change `SELECT DISTINCT ?o2 WHERE` to `SELECT DISTINCT * WHERE` and execute a query on Wikidata. For the given example it returns:

```
[
 {
  "s1": "wd:Q16027703",
  "p1": "wdt:P735",
  "o2": "wd:Q6581097"
 },
 {
  "s1": "wd:Q2976815",
  "p1": "wdt:P735",
  "o2": "wd:Q6581097"
 }
]
```

Listing 3. Result set of the SPARQL query of Listing 2 with `SELECT DISTINCT * WHERE`. The RDF vocabulary prefixes are ommited.

The third step is generating the NL representation. The SPARQL request is split into triples and all variables are substituted with their values. Each triple is converted into a text individually and then, if there is more than one triple, joined with the word "and". For example, the first set of values gives a sentence: "Claude-Nicolas Le Cat is given name Claude-Nicolas and Claude-Nicolas Le Cat's sex or gender is male." Although the sentence looks not fluent, it serves our goal well, which is clear from experimental results.

The experiment was carried out on balanced and unbalanced datasets. The results are shown in Table III.

TABLE III
DETAILED RESULTS FOR EXPERIMENT 2

F1 Score	Precision	Recall
balanced data (1 to 1)		
0.9982 ± 0.0011	**0.9980** ± 0.0020	**0.9983** ± 0.0008
unbalanced data (50 to 1)		
0.9631 ± 0.0170	0.9321 ± 0.0317	0.9968 ± 0.0053

The obtained results demonstrate that the difference between the well-formed textual answers and the generated answers quality metrics is not significant. However, in this experiment, the precision drops significantly for the unbalanced set.

C. Experiment 3: Bag-of-labels representation

In this experimental setting, we evaluate bag-of-labels answer verbalizations computed from entities and relations that are available in SPARQL query and its result set. The evaluation results of the AV classifier according to A_3 are shown in Table IV.

TABLE IV
DETAILED RESULTS FOR EXPERIMENT 3

F1 Score	Precision	Recall
balanced data (1 to 1)		
0.9613 ± 0.0029	**0.9355** ± 0.0109	**0.9886** ± 0.0089
unbalanced data (50 to 1)		
0.9205 ± 0.0570	0.9289 ± 0.0520	0.9170 ± 0.0870

In comparison to the balanced dataset setting, usage of unbalanced data (50 incorrect to 1 correct) leads to a quality drop. Additionally, the classification results demonstrate less robustness of the model while paying attention to standard deviation (as in the previous experiment). These results show that it is still possible to filter incorrect answers just by having the entity labels of the answer candidate set.

D. Filtering Answers for a Question Answering Process

In this subsection, the impact of AV on QA quality is demonstrated. We conducted the corresponding experiments using A_2 and A_3, the results are shown in Table V.

The results provide strong evidence that the QA quality was significantly increased while using both approaches in comparison to not-validated answer candidate sets. While considering P@1 (NDCG@1), the results were improved from 0.2476 to 0.4251 (i.e., by 71.7%) while using A_2 and to 0.2948 (i.e., by 19.1%) while using A_3. The corresponding results w.r.t. the P@5 are as follows: from 0.1036 to 0.1368 (i.e., by 32.0%) while using A_2 and to 0.1183 (i.e., by 14.2%) while using A_3. Finally, the results for NDCG@5 are: from 0.3249 to 0.4698 (i.e., by 44.6%) while using A_2 and to 0.3787 (i.e., by 16.5%) while using A_3.

The A_2 outperforms A_3 which intuitively comes from the fact that the A_2 produces more fluent answer verbalization rather than A_3. Hence, the AV classifier is able to capture more semantics and therefore, distinguish between correct and incorrect question-answer tuples. The A_1 obviously could not be evaluated as it would require a huge amount of manual work or a fine-tuned NLG module. However, we assume that it would produce even higher quality.

It is worth emphasizing that the AV filtering removed the majority of the answer candidates. For example, in the case of A_2 57 out of 60 candidates were removed on average, while in the case of A_3 – 44 out of 60. Consequently, the position of correct candidates in the list goes up because of filtering out the incorrect candidates.

E. Limitations

Notwithstanding the promising results described in the previous subsection, our work has several limitations. The generalizability of the study is limited by the used dataset (VANiLLa), KGQA system (QAnswer), NLG tool (Triple2NL), and model for AV classifier (BERT). The source data for A_2 (SPARQL queries) was constrained by removing `COUNT`, `MAX`, `LIMIT`, `ORDER` and `FILTER` clauses as Triple2NL can only verbalize simple RDF sequences.

In the subject-predicate-object structure, VANiLLa defines resource URIs solely for the predicate, in contrast, only labels are given for the subjects and the objects of the sentences. Hence, it becomes difficult to determine whether an answer candidate is correct or not without any ambiguity w.r.t. to the uniqueness of labels in Wikidata. For example, there are 6172 resources with the label "Correction" from the VANiLLa test subset. While it is hard to distinguish automatically entities with the same label for a given question, we considered any entity with a matching label as relevant for the current question.

There is another limitation in the VANiLLa dataset's structure. Some questions imply many correct answers, but only one of them is given as a reference in the dataset. In this case, we believe that if an object's (or subject's) label and predicate correspond to the VANiLLa record, then the answer is correct, regardless of the number of entities.

Obviously, these compromises might influence the quality (i.e., there is an opportunity to improve our approach).

F. Summary

Despite several limitations described in the previous subsection, the answer validation approach demonstrated its effectiveness by eliminating incorrect answer candidates based only on NL representations of questions and possible answers.

The AV classifier on well-defined answer verbalizations (A_1) as well as on the automatically generated answer verbalizations (A_2, A_3) demonstrated a reasonable F1 Score – 0.9968, 0.9982, and 0.9613 respectively for balanced data setup. The unbalanced data setup leads to the significant decrease in the classification quality. Therefore, the AV classifier should be trained on a balanced dataset to avoid a high false-negative rate while filtering answer candidates.

Given the comparison results before and after answer filtering with AV classifier, we were enabled to significantly improve the KGQA system's quality. The maximal improvement was achieved using A_2: +71.7% (P@1/NDCG@1), +32.0%

TABLE V
COMPARISON OF QA QUALITY BEFORE AND AFTER FILTERING BY OUR AV CLASSIFIERS

P@1 = NDCG@1		**P@5**		**NDCG@5**	
Before AV	**After AV**	**Before AV**	**After AV**	**Before AV**	**After AV**
0.2476	A_2	0.1036	A_2	0.3249	A_2
	0.4251		**0.1368**		**0.4698**
	A_3		A_3		A_3
	0.2948		0.1183		0.3787

(P@5), and +44.6% (NDCG@5). Although, A_3 also demonstrated reasonable quality improvement.

While answering **RQ**$_1$, the AV classifier is capable of eliminating wrong answers and, therefore, moving correct answers to the top of the result list. The AV classifier is capable of distinguishing between correct and incorrect answers based just on automatically generated verbalizations obtained within A_2 and A_3 (**RQ**$_3$). We could not fully answer **RQ**$_2$ as the manual NL generation for each query candidate requires huge labor costs. Therefore, extrapolating classification results of A_1 (Table II) on QA quality comparison experiments (Table V), we assume that the possible outcome of A_1 may even outperform A_2 and A_3.

VII. Conclusion

In this paper, we presented an approach to filter computed NL answer candidates within a KGQA process. We have purposely limited ourselves to use only NL features, in particular, labels of the entities, properties, and concepts, s.t., our approach is not tightly integrated into existing KGQA systems. Hence, our approach is a *generalized method for optimizing KGQA systems*. For proving this, we defined 3 research questions that were answered by our experiments. As we have used the state-of-the-art KGQA system QAnswer, we consider our results to be very solid, as we were capable of improving the quality even for this well-established system. Hence, we can conclude that our approach should be applicable to any QA system that provides additional information about the answer candidates.

Our results show a huge impact w.r.t. the VANiLLa dataset. In the executed experiments, we were capable of removing incorrect answer candidates (from the ranked list of SPARQL queries), s.t., the result improved by 71.7% and 44.6% (w.r.t. P@1 and NDCG@5).

In the future, we will integrate our approach into existing QA systems that are using NL input and output, s.t., an additional QA quality improvement is achieved. It would also be possible to use the results of our classification as an additional feature for the core of a QA engine, s.t., the quality of answer candidates might be improved on a lower system level. In particular, there are QA systems that provide high performance while at the same time not interpreting the question's semantics precisely. Such systems might benefit heavily from our approach.

References

[1] A. Ns, A. Azhari, and A. Sari, "Survey on answer validation for Indonesian question answering system (IQAS)," *International Journal of Intelligent Systems and Applications*, vol. 10, pp. 68–78, 04 2018.

[2] A. Rodrigo, J. Pérez-Iglesias, A. Peñas, G. Garrido, and L. Araujo, "A question answering system based on information retrieval and validation," in *CLEF 2010 LABs and Workshops, Notebook Papers*, 2010.

[3] J. Pérez-Iglesias, G. Garrido, Á. Rodrigo, L. Araujo, and A. Peñas, "Information retrieval baselines for the respubliqa task," in *CLEF*, 2009.

[4] A. Téllez-Valero, M. M. y Gómez, L. Pineda, and A. Peñas, "Towards multi-stream question answering using answer validation," *Informatica (Slovenia)*, vol. 34, pp. 45–54, 2010.

[5] B. Magnini, D. Giampiccolo, P. Forner, C. Ayache, V. Jijkoun, P. Osenova, A. Peñas, P. Rocha, B. Sacaleanu, and R. Sutcliffe, "Overview of the clef 2006 multilingual question answering track," in *Evaluation of Multilingual and Multi-modal Information Retrieval*, C. Peters, P. Clough, F. C. Gey, J. Karlgren, B. Magnini, D. W. Oard, M. de Rijke, and M. Stempfhuber, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 223–256.

[6] S. Babych, A. Henn, J. Pawellek, and S. Padó, "Dependency-based answer validation for German," in *CLEF 2011 Labs and Workshop, Notebook Papers, 19-22 September 2011, Amsterdam, The Netherlands*, ser. CEUR Workshop Proceedings, V. Petras, P. Forner, and P. D. Clough, Eds., vol. 1177. CEUR-WS.org, 2011.

[7] L. Vanderwende, "Answering and questioning for machine reading," in *Machine Reading, Papers from the 2007 AAAI Spring Symposium*, 2007, pp. 91–96.

[8] A. Grappy, B. Grau, M. Falco, A. Ligozat, I. Robba, and A. Vilnat, "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, 2011, pp. 55–62.

[9] I. Glöckner and K. Weis, "An integrated machine learning and case-based reasoning approach to answer validation," in *2012 11th International Conference on Machine Learning and Applications*, vol. 1, 2012, pp. 494–499.

[10] I. Glöckner and B. Pelzer, "The LogAnswer Project at CLEF 2009," in *Working Notes for CLEF 2009 Workshop co-located with the 13th European Conference on Digital Libraries (ECDL 2009) , Corfù, Greece, September 30 - October 2, 2009*, ser. CEUR Workshop Proceedings, C. Peters and N. Ferro, Eds., vol. 1175. CEUR-WS.org, 2009.

[11] P. Pakray, U. Barman, S. Bandyopadhyay, and A. Gelbukh, "Semantic answer validation using universal networking language," in *International Journal of Computer Science and Information Technologies*, vol. 3 (4), 2012.

[12] Á. Rodrigo, A. Peñas, and F. Verdejo, "Overview of the answer validation exercise 2008," in *Evaluating Systems for Multilingual and Multimodal Information Access*, C. Peters, T. Deselaers, N. Ferro, J. Gonzalo, G. J. F. Jones, M. Kurimo, T. Mandl, A. Peñas, and V. Petras, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 296–313.

[13] H. Gómez-Adorno, D. Pinto, and D. Vilariño, "A question answering system for reading comprehension tests," in *Pattern Recognition*, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. S. Rodríguez, and G. S. di Baja, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 354–363.

[14] A. Solovyev, *Dependency-Based Algorithms for Answer Validation Task in Russian Question Answering*. Springer, Berlin, Heidelberg, 01 2013, vol. 8105, pp. 199–212.

[15] ——, “Who is to blame and where the dog is buried? method of answers validations based on fuzzy matching of semantic graphs in question answering system.” in *ROMIP 2011*, Kazan, Russia, 2011, pp. 125–141.

[16] I. Zamanov, M. Kraeva, N. Hateva, I. Yovcheva, I. Nikolova, and G. Angelova, “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)*. Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 242–246.

[17] P. Nakov, L. Màrquez, W. Magdy, A. Moschitti, J. Glass, and B. Randeree, “SemEval-2015 task 3: Answer selection in community question answering,” in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 269–281.

[18] C. Tan, F. Wei, Q. Zhou, N. Yang, W. Lv, and M. Zhou, “I know there is no answer: Modeling answer validation for machine reading comprehension,” in *Natural Language Processing and Chinese Computing*, M. Zhang, V. Ng, D. Zhao, S. Li, and H. Zan, Eds. Cham: Springer International Publishing, 2018, pp. 85–97.

[19] M. Hu, F. Wei, Y. xing Peng, Z. Huang, N. Yang, and M. Zhou, “Read + verify: Machine reading comprehension with unanswerable questions,” *ArXiv*, vol. abs/1808.05759, 2019.

[20] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for SQuAD,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 784–789.

[21] M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, and J. Lehmann, “AskNow: A framework for natural language query formalization in SPARQL,” in *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, ser. Lecture Notes in Computer Science, H. Sack, E. Blomqvist, M. d’Aquin, C. Ghidini, S. P. Ponzetto, and C. Lange, Eds., vol. 9678. Springer, 2016, pp. 300–316.

[22] D. Diefenbach, A. Both, K. Singh, and P. Maret, “Towards a question answering system over the semantic web,” *Semantic Web*, vol. 11, pp. 421–439, 2020.

[23] D. Chen, A. Fisch, J. Weston, and A. Bordes, “Reading Wikipedia to answer open-domain questions,” in *Association for Computational Linguistics (ACL)*, 2017.

[24] B. Magnini, M. Negri, R. Prevete, and H. Tanev, “Is it the right answer? exploiting web redundancy for answer validation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, 2002, pp. 425–432.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013.

[26] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[27] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[28] D. Biswas, M. Dubey, M. R. A. H. Rony, and J. Lehmann, “VANiLLa: Verbalized answers in natural language at large scale,” *CoRR*, vol. abs/2105.11407, 2021.

[29] E. Kacupaj, B. Banerjee, K. Singh, and J. Lehmann, “ParaQA: A question answering dataset with paraphrase responses for single-turn conversation,” in *European Semantic Web Conference*. Springer, 2021, pp. 598–613.

[30] A. Both, A. Gashkov, and M. Eltsova, “Similarity detection of natural-language questions and answers using the VANiLLa dataset,” *Journal of Physics: Conference Series*, vol. 1886, no. 1, p. 012017, April 2021.

[31] V. Korablinov and P. Braslavski, “RuBQ: A Russian dataset for question answering over Wikidata,” in *International Semantic Web Conference*. Springer, 2020, pp. 97–110.

[32] A.-C. N. Ngomo, D. Moussallem, and L. Bühmann, “A holistic natural language generation framework for the semantic web,” *arXiv preprint arXiv:1911.01248*, 2019.

[33] T. Ferreira, C. Gardent, N. Ilinykh, C. van der Lee, S. Mille, D. Moussallem, and A. Shimorina, “The 2020 bilingual, bi-directional WebNLG+ shared task overview and evaluation results (WebNLG+ 2020),” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, 2020.

[34] R. S. Howard J, “Universal language model fine-tuning for text classification,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.

[35] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” in *33rd Conference on Neural Information Processing Systems NeurIPS*, 2019.

[36] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune BERT for text classification?” *ArXiv*, vol. abs/1905.05583, 2019.

[37] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.