Knowledge Graph Question Answering Datasets and Their Generalizability: Are They Enough for Future Research?

Longquan Jiang  
University Hamburg  
Hamburg, Germany  
longquan.jiang@uni-hamburg.de  

Ricardo Usbeck  
University Hamburg  
Hamburg, Germany  
ricardo.usbeck@uni-hamburg.de  

ABSTRACT

Existing approaches on Question Answering over Knowledge Graphs (KGQA) have weak generalizability. That is often due to the standard i.i.d. assumption on the underlying dataset. Recently, three levels of generalization for KGQA were defined, namely i.i.d., compositional, zero-shot. We analyze 25 well-known KGQA datasets for 5 different Knowledge Graphs (KGs). We show that according to this definition many existing and online available KGQA datasets are either not suited to train a generalizable KGQA system or that the datasets are based on discontinued and out-dated KGs. Generating new datasets is a costly process and, thus, is not an alternative to smaller research groups and companies. In this work, we propose a mitigation method for re-splitting available KGQA datasets to enable their applicability to evaluate generalization, without any cost and manual effort. We test our hypothesis on three KGQA datasets, i.e., LC-QuAD, LC-QuAD 2.0 and QALD-9). Experiments on re-split KGQA datasets demonstrate its effectiveness towards generalizability. The code and a unified way to access 18 available datasets is online at https://github.com/semantic-systems/KGQA-datasets as well as https://github.com/semantic-systems/KGQA-datasets-generalization.

CCS CONCEPTS

• Information systems → Question answering. Retrieval effectiveness.

KEYWORDS

KGQA, Question Answering, Generalization, Generalizability, Benchmark, Evaluation

1 INTRODUCTION

Existing approaches on Question Answering over Knowledge Graphs (KGQA) have weak generalizability since they operate with an i.i.d. assumption. The research field is missing well-defined datasets for training and evaluating systems towards generalizability.

Recently, Gu et al. [14] defined three levels of generalization, namely, i.i.d., compositional, zero-shot. First, KGQA systems should be able to handle all elements seen in the training set, such as classes, relations, and query components as well as query structures (i.i.d.). Second, the systems should be able to answer questions using known classes, relations, query components but new compositions of these in the test set. Third, generalizable KGQA systems should also be able to handle unseen classes, relations, and query components and structures, i.e., show zero-shot behavior. Consequently, Gu et al. [14] released a large-scale annotated KGQA dataset – GrailQA – for generalization evaluation. Note, these three levels do build on each other. That is, a test dataset which tests compositionality does comply with i.i.d. behavior. However, GrailQA is limited to Freebase, a discontinued and outdated knowledge graph.

In general, data collection and annotation for a different domain or knowledge graph is costly. An expansion to other KGs such as DBpedia, Wikidata is hard because this requires costly annotation of the logical queries to these KGs [15].

The rise of the transformer-architecture [7, 36] injected a reawakened increase in KGQA research. Our research opens a new evaluation perspective on KGQA systems and thus, fosters and guides the development of new KGQA systems and datasets.

To cost-efficiently reuse datasets from nearly a decade of research and thus generate more training and evaluation coverage, we propose a novel method for re-splitting KGQA datasets according to the three generalization levels without any cost.

To answer whether existing datasets are suitable for non-iid generalization evaluation, we analyze 25 well-known datasets. For 18 of these datasets that are online available, we establish a unified way to access them. We analyze the level of generalizability in each of these. To show the importance of well-split datasets, we train three recent KGQA systems (TeBaQA [37], BART [22], and HGNet [8]) on the original and on re-split datasets. Our experiments expose the weaknesses of the original datasets and shine a light on the generalization ability of the KGQA systems.

Our contributions are as follows:

• Analysis of existing KGQA datasets and systems capability to generalize.

• Development of a novel method for cost-effective re-splitting datasets to effectively train the generalization ability of KGQA systems.
A library for loading the KG datasets https://github.com/semantic-systems/KGQA-datasets. The library is based on the Hugging Face datasets library and uses an Apache 2.0 license.

- We also provide the code for generating generalizable versions of these datasets using an Apache 2.0 license.

As partners of the German national research data infrastructure, we will update and maintain the repositories for at least five more years.

2 NOTATION

2.1 Knowledge Graphs

Here, a knowledge graph is a multigraph, denoted by \( KG = (V, E) \), where \( V \) is a set of vertices and \( E \subseteq (E \cup B \cup C) \times R \times (E \cup B \cup C \cup L) \) is a set of edges expressed as RDF triples. With \( R \) being the set of relations, \( E \) being the set of entities, \( B \) a set of blank nodes indicating the existence of a resource with uniquely identifying it, \( C \) being the set of classes and \( L \) being the set of non-unique literal values such as strings and dates. The knowledge graph can be further subdivided into the ontology, commonly referred to the schema or T-Box \( O \subseteq C \times R \times C \) and the set of facts, instances or so-called A-Box \( M \subseteq (E \cup B) \times R \times (E \cup B \cup C \cup L) \) [14].

2.2 Knowledge Graph Question Answering

KGQA aims at retrieving the correct answer or respectively answers denoted as \( A \) of a given natural language question \( Q_{NL} \) from the knowledge graph \( KG \). This can be done by learning a transformation function \( f \) translating the question \( Q_{NL} \) into a semantically equivalent logical form \( Q_{LF} \). i.e., \( f : Q_{NL} \mapsto Q_{LF} \). In our case, we use SPARQL as the logical form. A SPARQL Abstract Query is a tuple \( (A, DS, F) \), where \( A \) is a SPARQL algebra expression, \( DS \) is an RDF knowledge graph and \( F \) is a logical query form with operators \( \circ \) such as SPARQL solution modifiers, FILTER expressions or operators.

Note, there are also graph traversal-based KGQA systems which are out-of-scope of this work.

2.3 Generalizability in KGQA

The definition of the three levels of generalization in KGQA stems from the work by Gu et al. [14]. We clarify it as follows for better contextualization.

Let \( S = R \cup C \cup \circ \) be the set of schema items. Let \( S_{train} \subseteq S \) be the set of schema items in the training split of a dataset. Let \( S_q \subseteq S \) be the set of schema items in the question at hand. The test set \( Q \) is defined as follows:

- **I.I.D. generalization**: \( \forall q \in Q : S_q \subseteq S_{train} \) and \( q \) follows the training distribution. That is, all relations \( R \), classes \( C \) and logical form constructs \( \circ \) have been seen while training but not the actual entities \( E \) and literals \( L \). In particular, the specific logical form constructs are known.

- **Compositional generalization**: This level differs from I.I.D. generalization in that the specific logical form construct and its operators \( \circ \) are not part of \( S_{train} \) but relations and classes are all known.

1https://www.w3.org/TR/rdf-sparql-query/#sparqlDefinition

- Zero-shot generalization: \( q \in Q : \exists s \in S_q, s \in S \setminus S_{train} \)

That is, for all test questions, there is at least one schema item \( R \cup C \cup \circ \) per question which is not in the training dataset.

To reiterate, the levels do build on each other. However, it is important to build not only one but three different test sets for evaluation.

3 RELATED WORK

Our work is in line with research on KGQA benchmarking and system evaluation. A wide range of KGQA benchmarks and datasets as well as analyses thereof have been created to evaluate KGQA systems for simple and complex questions over different publicly available Knowledge Graphs (KGs). This includes datasets such as WebQuestions [4] and GrailQA [14] for Freebase, LC-QuAD [32] and QALD-9 [34] for DBpedia, RuBQ [21] and CronQuestions [25] for Wikidata. However, none of them are geared towards generalization in KGQA. GrailQA is an exception, but its process of large-scale manual annotation is highly expensive and it is targeted at the not-maintained Freebase. QALD-9 is the only other dataset with similar distributional characteristics as GrailQA, but it is very small with only 558 question-answer pairs.

Bouziane et al. [6] surveyed a total of 31 different KGQA systems with respect to specific characteristics, e.g., interfaces to databases, ontologies, etc. They presented an analysis of the quality of these KGQA systems using multiple indicators, e.g., success rate, correct answers. Saleem et al. [24] conducted a fine-grained analysis of the state-of-the-art KGQA systems on QALD-6 [33], and demonstrated that various natural language phenomena and SPARQL features would affect their final performance. Affolter et al. [1] presented another survey of 24 KGQA systems, but they focused on general KGQA systems, not on KGQA systems specifically. Steinmetz and Sattler [29] analysed into 26 popular KGQA datasets (e.g., QALD, LC-QuAD), their structure and characteristics like ambiguity, lexical gap, complex queries, etc. Their experimental results revealed the specific challenges existing KGQA systems are facing and are required to overcome. Although there are a lot of overview articles looking into KGQA systems and their performance, none of these looks deeply into the generalization aspect of datasets and, thus, their ability to help train generalizable KGQA systems.

To enhance evaluation insights, there are approaches modifying one particular dataset to evaluate specific characteristics of KGQA systems. Siciliani et al. [27] presented the MQALD dataset to evaluate how well KGQA systems can work with certain SPARQL modifiers. They extracted queries with modifiers from QALD and added manually created queries to form a new dataset. To bridge the gap between a user and semantic data in the e-commerce industry, Kaffee et al. [19] extended the base dataset (QALD-9 [34]) by adding SPARQL queries over different KGs. The new dataset is dubbed QALM\(^2\). Singh et al. [28] as well as Ding et al. [12] re-split LC-QuAD [32] to use only queries answerable over an updated KG, i.e., DBpedia 2018 instead of 2016 to overcome outdated queries resulting in 3,253 question-answer pairs instead of 5,000.

We are the first to systematically offer a re-splitting mechanism to enable better training for generalizability in KGQA.

\(^2\)QALM is available at https://github.com/luciekaffee/QALM
4 GENERATION OF SPLITS

4.1 Generalizability

Existing KGQA systems have weak generalizability due to the i.i.d assumption they operate with. That is, they assume that any test set follows the distribution of seen characteristics in a given training dataset. In addition, the shortage of evaluation sets for the ability to generalize hinders the further development of KGQA systems.

We show that the definition of generalization levels is not clear enough and contradicting. That is, the paper [14] suggests that the level of generalization depend on each other - compare Fig 1 [14] - but on the contrary, they do not depend on each other. We measure whether the provided dataset GraidQA follows its definition. We demonstrate that it does not. Thus, we clarify the definition of [14] and provide methods to check for the different levels of generalization. We also establish a method to re-split any dataset to follow the provided definition. These novel datasets allows for more challenging KGQA research.

4.2 Test for three levels of generalization

Given a KGQA dataset, to know whether and how much it is suitable for generalization evaluation of KGQA systems, we developed a simple algorithm to test which of the definitions holds for its test split. Algorithm 1 describes the way of determining the level of generalization of a given question q w.r.t. a given training set. Note, $d_3$ is a dictionary of unique, sorted tuples of schema terms, which allows us to distinguish whether this composition has already been seen in the training dataset. Each of the sets of sorted schema terms becomes a key in the dictionary in the form of a tuple. The value is the index or id of this tuple of schema terms, set to the current length of the dictionary $d_3$. For example:

*How many countries are on the continent of North America?*

which has the corresponding SPARQL query over Wikidata KG:

```
SELECT (COUNT(DISTINCT ?country) AS ?result) WHERE
{ ?country wdt:P31 wd:Q6256.
  ?country wdt:P30 wd:Q49.
} which has results in the following dictionary entry:
```

```
Dict({("COUNT", "wdt:P30", "wdt:P31"),: 0})
```

4.3 Generation Method

To overcome the limited number of available training datasets towards generalization, we implemented a method to generate new splits for a given dataset. That is, we can generate three different subsets for i.i.d., compositional, and zero-shot generalization. We maintain the same proportions of train/test splits as the original dataset as far as possible.

Assume that we have a KGQA dataset $D = \{(q, l, s_q) | q \in Q, l \in LQ, s_q \subset S\}$. We define $q$ as a given natural language question, $Q$ is the set of all questions, $l$ is the logic form, e.g., SPARQL of the question $q$, $LQ$ is the set of all possible logic forms, $s_q$ is the set of schema terms extracted from $l$ using pattern matching, $S$ is the set of unique schema terms of all questions $Q$. For schema terms, we only consider relations, classes, COUNT, and comparative functions (e.g., "<", ">", "<=", ">=", and "!="), similar to [14]. In the following, we use the complete dataset to produce the new train and test splits.

In our approach, we have two variables (see Algorithm 1): (1) $d_q$, a set of unique schema terms for all questions $Q$ in the dataset $D$, and (2) $d_3$, a dictionary to store unique tuples of schema terms for all the questions $Q$ in the dataset $D$. Each of the sets of sorted schema terms becomes a key in the dictionary in the form of a tuple (see Section 4.2). The value is the index or id of this tuple of schema terms. First, we assign a group id gid to each question $q$ in the dataset $D$ by getting the key of the set of schema terms of $q$ in $d_q$. Second, we start to re-split the given dataset. The goal is to re-split a given dataset $D$ into two subsets $D_{\text{train}}$ and $D_{\text{test}}$, where $D_{\text{test}}$ have further three subsets, i.e., $D_{\text{zero}}, D_{\text{compo}}$, and $D_{\text{iid}}$ w.r.t. three levels of generalization respectively. See Algorithm 3 and Algorithm 4 in the appendix for more details. In the end, the generalizability of a KGQA model can be evaluated on these new splits.

Note, our algorithm allows to set the size of each of these subsets ($D_{\text{zero}}, D_{\text{compo}}, D_{\text{iid}}$), i.e., one can emphasize one particular generalization level over the others.

Algorithm 2 describes our overall procedure for re-splitting a given KGQA dataset. First, we sample candidate questions for the corresponding level of generalization in a descending way, from the highest level (zero-shot) to the lowest level (i.i.d.). The candidate question set previously sampled for a higher level will not affect lower-level sets. Note that, in each sampling round, an additional filtering process, i.e., do splitting and concatenation, is required, so that in the next round we ensure that the candidate question set drawn from the remaining full set excludes the previously sampled questions. In Algorithm 2, $D_i$, $i \in \{1, 2\}$ represents the resultant data set as the full training set for the next sampling step. After the final sampling step, i.e., Step 3, the resultant data set $D_{\text{train}}$ would be the final training set for the given KGQA dataset $D_0$ using our approach re-splitting approach. For the two helper functions, i.e., do_GSS and do_splitting, see their implementations in Section A of Appendix. For the train_test_split function, we use the official implementation3 of the sklearn library.

```python
Algorithm 1: Determine the level of generalization of one given question determine_level.

**Input:** A given question $q$, the set of unique schema terms in training set $S_{\text{train}}$, a dictionary of unique sets of schema terms $d_3$

**Output:** The level of generalization of one given question $l_q$

1. $S_{\text{seen}} \leftarrow S_q \setminus S_{\text{train}}$
2. if $S_q \not\subseteq d_3$ then
3. if $S_{\text{seen}} \neq \emptyset$ then $l_q \leftarrow \text{"zero-shot"}$
4. else $l_q \leftarrow \text{"compositional"}$
5. else $l_q \leftarrow \text{"iid"}$
6. return $l_q$
```

3https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
5 ANALYSIS

5.1 Experimental Setup

Our main evaluation goal is to thoroughly evaluate the effectiveness of dataset re-splitting in increasing the generalization ability of KGQA systems. We use the following three KGQA models to verify our hypothesis.

- TeBaQA [37] is a template-based KGQA system, which constructs the correct SPARQL query by first mapping natural language questions into isomorphic basic graph pattern classes and then filling semantic information. Since TeBaQA has very high runtimes, we use it only on smaller datasets.

- BART [22] is a sequence-to-sequence model based on a denoising autoencoder, which is fine-tuned on the machine translation task. We use BART to evaluate performance on LC-QuAD and LC-QuAD 2.0. In particular, on Wikidata-based datasets, due to its speed and original purpose.

- HGNet [8] is a hierarchical autoregressive generation model, which completes the query graph by first outlining its structure and then filling the instances into these structures. HGNet is used to evaluate different DBpedia datasets as originally intended by the authors.

**Evaluation Metrics.** For measuring the performance, we adopt Precision, Recall, F1 score, and Hits@1.

**Implementation Details.** Our experiments are twofold: (1) we perform a preliminary analysis on the original datasets, (2) we evaluate the re-splitted datasets. For both stages, the model parameters are fixed and consistent with the original papers. For the HGNet model, we use the same configurations as in [8]. To be specific, we set the size of hidden layers to 256, the number of BiLSTM layers to 1, the number of the graph transformer layer to 3, the learning rate to $2 \times 10^{-4}$, the size of each beam to 7. For the BART model, we also use the same hyperparameters as in [8]. For the TeBaQA system, we deploy a local instance on our server and test the three KGQA datasets using the default query API.

5.2 Test existing KGQA Datasets

To facilitate the development of KGQA systems in the Semantic Web community, researchers have generated many KGQA datasets over different publicly available Knowledge Graphs for specific purposes. We examined 25 popular KGQA datasets (and their splits) towards their inherent ability to train the generalization ability of KGQA systems, see Table 1. We also calculated the percent of questions w.r.t. three levels for 8 dataset which provide SPARQL queries, see Table 6 in the appendix. Note, a dataset does not automatically mean, the dataset is suitable for the evaluation but indicates the sole existence of queries from this level of generalization. From this evaluation, we have the following observations:

(1) All KGQA datasets targeting Freebase, except for GrauILQA, lack the capability to evaluate zero-shot generalization.

(2) Out of the KGQA datasets targeting DBpedia or Wikidata, QALD-9, LC-QuAD 1.0 and LC-QuAD 2.0 are inherently able to evaluate these three levels of generalization, but they suffer from unbalanced or small-scale issues.

(3) Most KGQA datasets have no capability to evaluate compositional generalization. The exceptions are CFQ and CWQ which are specifically designed for measuring the compositional ability of KGQA systems.

(4) Simple factoid KGQA datasets (e.g., SimpleQuestions, SimpleDBpediaQA, etc.) can only be used to evaluate i.i.d. generalization. This is due to the structure of simple questions involving only one basic graph pattern and thus having no complex structure.

Additionally, we conducted a preliminary performance analysis of the BART and HGNet KGQA systems using LC-QuAD and LC-QuAD 2.0. These systems use neural networks and thus should have a better generalization ability than template-based approaches such as TeBaQA. We calculate the number of questions answered correctly, partially correct, and incorrect w.r.t. different levels of generalization. Our analytical results are shown in Figure 1. We find that i.i.d. and compositional questions contribute a lot to the final performance of BART and HGNet on LC-QuAD, as they account for the largest proportion in LC-QuAD. A similar observation holds for BART on LC-QuAD 2.0. HGNet was developed for DBpedia and Freebase, not Wikidata and thus was not evaluated.
Table 1: Evaluation results of 25 common KGQA datasets analyzed in this work w.r.t. three levels of generalization. ✓ indicates that the test dataset has questions from this level of generalization. ✗ means there is not a single question for this level of generalization. (✓) or (✗) indicates that we deem it suitable or not suitable through literature analysis. The dash symbol '-' indicates that the level of generalization could neither be measured by literature nor by dataset analysis due to missing SPARQL queries (∗) or valid train/test splits (♣).

| Dataset | KG | Year | Generalization |
|---------|----|------|----------------|
|         |    |      | T.I.D. | Compositional | Zero-Shot |
| WebQuestions [4] | Freebase | 2013 | ✓ | (✗) | (✗) |
| SimpleQuestions [5] | Freebase | 2015 | ✓ | (✗) | (✗) |
| ComplexQuestions [3] | Freebase | 2016 | - | - | - |
| GraphQuestions [30] | Freebase | 2016 | ✓ | ✓ | (✗) |
| WebQuestionsSP [38] | Freebase | 2016 | ✓ | (✗) | (✗) |
| The 30M Factoid QA [26] | Freebase | 2016 | ✓ | ✗ | ✗ |
| SimpleQuestionsWikidata [11] | Wikidata | 2017 | ✓ | ✗ | ✗ |
| LC-QuAD 1.0 [32] | DBpedia | 2017 | ✓ | ✓ | ✓ |
| ComplexWebQuestions [31] | Freebase | 2018 | ✓ | (✗) | (✗) |
| QALD-9 [34] | DBpedia | 2018 | ✓ | ✓ | ✓ |
| PathQuestion [40] | Freebase | 2018 | - | - | - |
| MetaQA [39] | WikiMovies | 2018 | - | - | - |
| SimpleDBpediaQA [2] | DBpedia | 2018 | ✓ | ✗ | ✗ |
| TempQuestions [16] | Freebase | 2018 | - | - | - |
| LC-QuAD 2.0 [13] | Wikidata | 2019 | ✓ | ✓ | ✓ |
| FreebaseQA [18] | Freebase | 2019 | - | - | - |
| Compositional Freebase Questions [20] | Freebase | 2020 | ✓ | ✓ | ✗ |
| RuBQ [21] | Wikidata | 2020 | - | - | - |
| GrailQA [14] | Freebase | 2020 | ✓ | (✓) | (✓) |
| Event-QA [9] | EventKG | 2020 | - | - | - |
| RuBQ 2.0 [23] | Wikidata | 2021 | - | - | - |
| MLPQ4 | DBpedia | 2021 | - | - | - |
| Compositional Wikidata Questions [10] | Wikidata | 2021 | ✓ | ✓ | ✗ |
| TimeQuestions [17] | Wikidata | 2021 | - | - | - |
| CronQuestions [25] | Wikidata | 2021 | - | - | - |

The results verify our assumption: the tested KGQA datasets are not sufficient to train KGQA systems for higher levels of generalizability. Note, we directly use the original splits to train and test KGQA systems. One solution would be to construct a brand-new specific KGQA dataset from scratch for generalization evaluation. However, the process of data collection and annotation would be prohibitively expensive and inflexible. Moreover, we think existing KGQA datasets can and should be recycled if we can design an algorithm to re-split them.

Figure 1: The preliminary analytical results of the re-splitted datasets.
5.3 Re-splitting existing Datasets

We hypothesize that existing KGQA datasets could be also applicable to evaluate generalization if we re-split them. We test our hypothesis on three popular KGQA datasets, i.e., LC-QuAD, LC-QuAD 2.0, and QALD. LC-QuAD*, LC-QuAD 2.0*, and QALD* are derived from their original versions using our proposed re-splitting methodology.

QALD. In this work, we use QALD-9[34], the most recent QALD challenge at the time of writing. It consists of 508 multilingual questions in total which are compiled and curated from the previous challenges and expert annotators. In particular, the test set is always created from scratch for each challenge. Note, its small size hinders adoption in end-to-end large-scale KGQA systems.

LC-QuAD. This KGQA dataset is the first large-scale complex question dataset over DBpedia, containing 4000 training and 1000 test question-answer pairs. The answers are provided as SPARQL queries. Note, the questions and queries are based on templates [32].

LC-QuAD 2.0. In comparison to LC-QuAD, the updated version - LC-QuAD 2.0 [13] - has 30,000 unique pairs of questions and SPARQL queries, with a higher language variation and more question types. LC-QuAD 2.0 provides DBpedia and Wikidata SPARQL queries. However, these are automatically generated so their quality cannot be guaranteed. Thus, here we only use the provided Wikidata SPARQL queries, from which schema terms are extracted.

Table 2: Percent of questions with different levels of generalization in the given test sets of QALD, LC-QuAD, and LC-QuAD 2.0.

| Dataset     | I.I.D | Compositional | Zero-Shot |
|-------------|------|---------------|-----------|
| QALD        | 30.67% | 35.33%         | 34.00%    |
| LC-QuAD     | 43.40% | 55.90%         | 0.70%     |
| LC-QuAD 2.0 | 76.51% | 15.68%         | 7.81%     |

Table 3: Percent of questions with different levels of generalization in the given test sets of QALD*, LC-QuAD*, and LC-QuAD 2.0*.

| Dataset     | I.I.D | Compositional | Zero-Shot |
|-------------|------|---------------|-----------|
| QALD*       | 8.09% | 23.70%        | 68.21%    |
| LC-QuAD*    | 20.95% | 64.62%        | 14.43%    |
| LC-QuAD 2.0* | 40.55% | 32.68%        | 26.78%    |

Table 4: Dataset Statistics.

| Dataset     | Total | Train | Dev | Test | Total | I.I.D | Compositional | Zero-Shot |
|-------------|-------|------|-----|------|-------|------|---------------|-----------|
| QALD-9      | 558   | 408  | -   | 150  | 53    | 51   |
| QALD-9*     | 558   | 385  | -   | 173  | 14    | 41   | 118          |
| LC-QuAD     | 5000  | 4000 | -   | 1000 | 344   | 559  | 7            |
| LC-QuAD*    | 5000  | 4000 | -   | 1000 | 344   | 559  | 7            |
| LC-QuAD 2.0 | 30221 | 24177| 521 | 6633 | 4624  | 948  | 472          |
| LC-QuAD 2.0*| 30221 | 24177| 521 | 6633 | 4624  | 948  | 472          |

Figure 2 shows the statistics of these three datasets w.r.t. different levels of generalization without re-splitting. We find that questions for stronger generalization levels account for a smaller percentage of the test set, except for QALD-9, which has a balanced distribution. Note, since the original QALD-9 dataset is suited for i.i.d evaluation, we increased the zero-shot percentage disproportionately on purpose. LC-QuAD has the most unbalanced distribution. There are only 7 questions for measuring zero-shot generalization, accounting for 0.7% - much smaller compared to the other two levels. This unbalanced distribution constraints the applicability of LC-QuAD to evaluate stronger generalization of KGQA systems trained with it despite its scale. Note, LC-QuAD 2.0 is more balanced.

Validation of KGQA systems for generalizability depends on the splits on which they are trained and tested. Thus, we employ our re-splitting algorithm. The statistics of re-splitted datasets are shown in Figure 3. Compared to the original splits of these three datasets, the new datasets emphasize balancing question distributions w.r.t. three levels of generalization, especially for the zero-shot level. Table 4 shows the details of original and re-splitted versions of these three KGQA datasets.

5.4 Evaluating the Re-split

To assess the performance boost due to the use of re-splitted datasets in training, we run experiments on re-splitted versions of QALD, LC-QuAD, and LC-QuAD 2.0 datasets. The evaluation results are shown in Figure 2.

In detail, Figure 2a and Figure 2b show the F1 scores obtained by testing TeBaQA, BART and HGNet on LC-QuAD and LC-QuAD* respectively.

We find that the F1 scores of TeBaQA on the re-splitted version of LC-QuAD w.r.t. different levels decrease to 0.0, indicating that the TeBaQA system does not have the capability of being generalizable within LC-QuAD. The main reason is that in the original split some test question features leaked into the train dataset due to the template-based design of the dataset. Thus, this result does not indicate generalizability. That is, due to the complex and diverse structure of questions, the SPARQL templates which TeBaQa learns to map a question cannot be learned with a proper split. Figure 2c shows the F1 scores using TeBaQA on QALD and its re-splitted version. Compared with the original splits, QALD* poses a significant challenge to the final performance of TeBaQA.

To show the scalability of our approach, we also tested the BART system on LC-QuAD 2.0 which is based on the large-scale Wikidata KG. The results indicate, our approach can also ensure the good quality and difficulty of the new splits on large datasets and on a different KG, namely, Wikidata while increasing the proportions of compositional and zero-shot questions (see Figure 2d). BART performance worsens on LC-QuAD 2.0* as opposed to its performance on LC-QuAD*. This is due to the new balance, where before LC-QuAD was biased towards i.i.d. and compositional questions, and its re-split is more evenly distributed. Thus, the evaluation shows an artifact. LC-QuAD 2.0 was less biased and is now overall more balanced, demanding more generalizability and, thus, the system performs worse.
Overall, one can observe a more realistic evaluation of KGQA systems towards their capability to generalize to unseen schema items and logical forms.

6 SUMMARY AND FUTURE WORK
Research on the ability to generalize for any artificial intelligence system, such as KGQA systems, is important and requires high-quality datasets. We introduced a test for three levels of generalization capability evaluation over KGQA datasets. We developed a method to re-split existing KGQA datasets to avoid unfair evaluations as well as costly generation of new datasets. For the first time, our evaluation spots a gap in the KGQA evaluation landscape pointing towards test leakage. Using re-split datasets, we show that the investigated systems are not able to generalize well on re-split datasets (QALD*, LC-QuAD 2.0*) which do not have test leakage. Additionally, for newly balanced datasets (LC-QuAD), we show that performance can go up.

In the future, we will extend our generation methods for more detailed evaluations which in turn can be used by agent-based KGQA systems to train themselves on weak points. We will also integrate this evaluation paradigm in our benchmarking platform [35] to sustain its use.

We hope to advance the understanding of the generalization ability of different KGQA systems and to support the KGQA community.

ACKNOWLEDGMENTS
The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Climate Action of Germany in the project CoyPu (project number 01MK21007G).

REFERENCES
[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. VLDB J. 28, 5 (2019), 793–819. https://doi.org/10.1007/s00778-019-00567-8
[2] Michael Azmy, Peng Shi, Jimmy Lin, and Ihab F. Ilyas. 2018. Farewell Freebase: Migrating the SimpleQuestions Dataset to DBpedia. In Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018, Emily M. Bender, Leon Derczynski, and Pierre Isabelle (Eds.). Association for Computational Linguistics, 2093–2103. https://aclanthology.org/C18-1178/
[3] Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-Based Question Answering with Knowledge Graph. In COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan, Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad (Eds.). ACL, 2503–2514. https://aclanthology.org/C16-1236/
[4] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EACL 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL. ACL. 1533–1544. https://aclanthology.org/D13-1160/
[5] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale Simple Question Answering with Memory Networks. CoRR abs/1506.02075 (2015). arXiv:1506.02075 http://arxiv.org/abs/1506.02075
[6] Abdelghani Bouriane, Djelloul Bouchka, Noureddine Doumi, and Mimoun Malki. 2015. Question Answering Systems: Survey and Trends. Procedia Computer Science 73 (2015), 366–375. https://doi.org/10.1016/j.procs.2015.12.005 International
To implement our re-splitting algorithm, we also designed the following two helper functions. Algorithm 3 aims to offer randomized train/test indices to split a given dataset according to a third-party provided group. We used the official GroupShuffleSplit implementation\(^\text{8}\) of the scikit-learn library\(^\text{9}\). For each sampling step of three levels, Algorithm 3 return train/test splits from the remaining dataset (e.g., \(D_0, D_1, D_2\)) in the previous step. The test split would be the candidate subset for the corresponding sampling step, while the train split would be the full set for the next sampling step. Algorithm 4 aims to split a test data \(D_{test}\) into three subsets w.r.t. three levels of generalization, i.e., \(D_{zero}, D_{compo}, \) and \(D_{iid}\), according to the training set \(D_{train}\).

### A HELPER FUNCTIONS

To implement our re-splitting algorithm, we also designed the following two helper functions. Algorithm 3 aims to offer randomized train/test indices to split a given dataset according to a third-party

---

\(^8\)https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupShuffleSplit.html

\(^9\)https://scikit-learn.org
Table 5: Evaluation Results on LC-QuAD, QALD, LC-QuAD 2.0 and their corresponding re-splits.

| Model       | Overall | I.I.D. | Compositional | Zero-Shot |
|-------------|---------|--------|---------------|-----------|
|             | P       | R      | F1            | Hits@1    | P       | R      | F1    | Hits@1 |
| LC-QuAD *   |         |        |               |           |         |        |       |        |
| HGNet       | 49.48   | 49.48  | 49.48         | 49.35     | 50      | 50     | 50    | 49.24 |
| BART        | 26.58   | 27.36  | 26.55         | 25.17     | 31.21   | 30.99  | 31.35 | 26.19 |
| TeBaQA      | 0       | 0      | 0             | 0         | 0       | 0      | 0     | 0      |
| LC-QuAD 1.0 |         |        |               |           |         |        |       |        |
| HGNet       | 52.14   | 52.14  | 52.14         | 47.24     | 58.42   | 58.42  | 58.42 | 47.42 |
| BART        | 22.45   | 22.94  | 22.38         | 17.86     | 28.5    | 28.85  | 28.54 | 17.89 |
| TeBaQA      | 2.72    | 2.61   | 2.54          | 1.63      | 4.16    | 4.1    | 4.14  | 1.63  |
| QALD-9*     |         |        |               |           |         |        |       |        |
| TeBaQA      | 10.8    | 12.54  | 9.98          | 4.24      | 20.86   | 17.05  | 18.17 | 21.43 |
| LC-QuAD 2.0 |         |        |               |           |         |        |       |        |
| BART        | 13.87   | 12.34  | 12.08         | 8.63      | 21.19   | 21.16  | 20.15 | 19.57 |
| QALD-9      | 2.91    | 2.92   | 2.91          | 0.75      | 4.39    | 4.39   | 4.39  | 4.39  |
| LC-QuAD 2.0 |         |        |               |           |         |        |       |        |
| BART        | 4.11    | 4.17   | 4.11          | 3.9       | 4.56    | 4.57   | 4.55  | 4.56  |

Table 6: Analytical results of 8 KGQA datasets about the percent of questions w.r.t. three levels of generalization.

| Dataset                          | Complex Questions? | Generalization |
|----------------------------------|--------------------|----------------|
|                                 |                    | I.I.D. | Compositional | Zero-Shot |
| The 30M Factoid QA [26]          | No                 | 100.00%| 0.00%         | 0.00%     |
| SimpleQuestionsWikidata [11]     | No                 | 100.00%| 0.00%         | 0.00%     |
| LC-QuAD 1.0 [32]                 | Yes                | 43.40% | 55.90%        | 0.70%     |
| QALD-9 [34]                      | Yes                | 30.67% | 35.33%        | 34.00%    |
| SimpleDBpediaQA [2]              | No                 | 100.00%| 0.00%         | 0.00%     |
| LC-QuAD 2.0 [13]                 | Yes                | 76.51% | 15.68%        | 7.81%     |
| GrailQA [14]                     | Yes                | 25.00% | 25.00%        | 50.00%    |
| Compositional Wikidata Questions [10] | Yes          | 49.15% | 50.85%        | 0.00%     |

C  DATASET ANALYSIS

Table 6 shows the analytical results of 8 KGQA datasets about the percent of questions w.r.t. the three levels of generalization [14] using our open-source testing method. The result of GrailQA was copied from the original paper [14].