Do I have the Knowledge to Answer? Investigating Answerability of Knowledge Base Questions

Mayur Patidar[†], **Prayushi Faldu**[‡], **Avinash Singh**[†], **Lovekesh Vig**[†],
Indrajit Bhattacharya[†], **Mausam**[‡]
[†]TCS Research, [‡]Indian Institute of Technology, Delhi
{patidar.mayur, singh.avinash9, lovekesh.vig, b.indrajit} @tcs.com
prayushifaldu123@gmail.com, mausam@cse.iitd.ac.in

Abstract

When answering natural language questions over knowledge bases, missing facts, incomplete schema and limited scope naturally lead to many questions being unanswerable. While answerability has been explored in other QA settings, it has not been studied for QA over knowledge bases (KBQA). We create *GrailQAbility*, a new benchmark KBQA dataset with unanswerability, by first identifying various forms of KB incompleteness that make questions unanswerable, and then systematically adapting GrailQA (a popular KBQA dataset with only answerable questions). Experimenting with three state-of-the-art KBQA models, we find that all three models suffer a drop in performance even after suitable adaptation for unanswerable questions. In addition, these often detect unanswerability for wrong reasons and find specific forms of unanswerability particularly difficult to handle. This underscores the need for further research in making KBQA systems robust to unanswerability.

1 Introduction

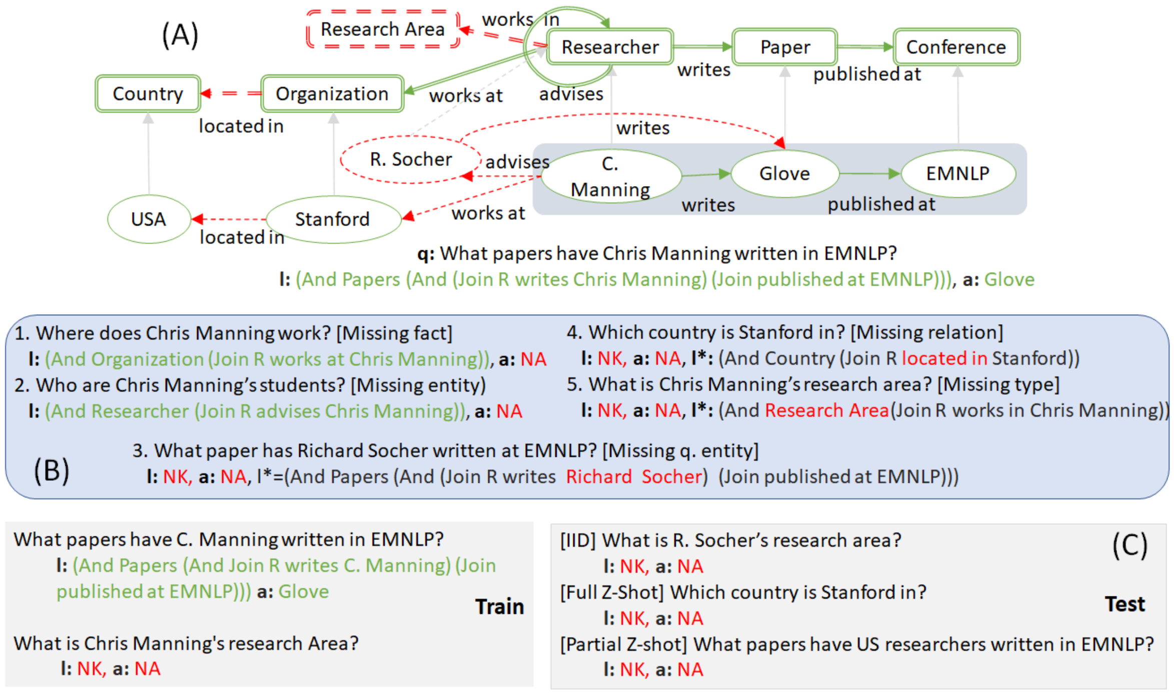
The problem of natural language question answering over knowledge bases (KBQA) has received a lot of interest in recent years (Saxena et al., 2020; Zhang et al., 2022; Mitra et al., 2022; Wang et al., 2022; Das et al., 2022; Cao et al., 2022c; Ye et al., 2022; Chen et al., 2021; Das et al., 2021). An important aspect of this task for real-world deployment is detecting answerability of questions. This problem arises for KBs due to various reasons, including schema-level and data-level incompleteness of KBs (Min et al., 2013), limited KB scope, questions with false premises, etc. In such cases, a robust and trustworthy model should detect and report that a question is unanswerable, instead of outputting some incorrect answer.

Answerability is well studied for QA over unstructured contexts (Rajpurkar et al., 2018; Choi et al., 2018; Reddy et al., 2019; Sulem et al., 2022; Raina and Gales, 2022). However, there is no existing work on answerability for KBQA. Benchmark KBQA datasets (Gu et al., 2021; Yih et al., 2016a; Talmor and Berant, 2018; Cao et al., 2022a) contain only answerable questions.

We first identify how different categories of KB incompleteness (schema and data incompleteness) affect answerability of questions. Then, using GrailQA (Gu et al., 2021), one of the largest KBQA benchmark dataset, we create a new benchmark for KBQA with unanswerable questions, which we call *GrailQAbility*, by deleting various elements from the KB to simulate scope and fact coverage limitations. This involves addressing a host of challenges, arising due to different ways in which KB element deletion affects answerability of questions, dependence between deletion of different types of KB elements, the shared nature of KB elements across questions, and more. We also define and include different generalization scenarios for unanswerable questions in the test set, namely IID and zero-shot, mirroring those for answerable questions.

We then use GrailQAbility to evaluate the robustness of three recent state-of-the-art KBQA models, RnG-KBQA (Ye et al., 2022), ReTraCk (Chen et al., 2021) and TIARA (Shu et al., 2022), against unanswerable KB questions. We find that all three models suffer an overall drop in performance with unanswerable questions, even after appropriate adaptation for unanswerability via retraining and thresholding. More alarmingly, these often detect unanswerability for incorrect reasons, raising concerns about trustworthiness. Additionally, while the strength of these models is that they learn at the schema-level, we find that this also results in significantly poorer ability to detect data-level incompleteness. Using error analysis, we identify important failure points for these models. All of these highlight robustness issues for KBQA models in real applications, raising important questions for future research.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 10341–10357
July 9-14, 2023 ©2023 Association for Computational Linguistics



q: What papers have Chris Manning written in EMNLP?
l: (And Papers (And (Join R writes Chris Manning) (Join published at EMNLP))), **a:** Glove

1. Where does Chris Manning work? [Missing fact]
   l: (And Organization (Join R works at Chris Manning)), **a:** NA
2. Who are Chris Manning's students? [Missing entity]
   l: (And Researcher (Join R advises Chris Manning)), **a:** NA
3. What paper has Richard Socher written at EMNLP? [Missing q. entity]
   l: NK, **a:** NA, l*=(And Papers (And (Join R writes Richard Socher) (Join published at EMNLP)))
4. Which country is Stanford in? [Missing relation]
   l: NK, **a:** NA, **l*:** (And Country (Join R located in Stanford))
5. What is Chris Manning's research area? [Missing type]
   l: NK, **a:** NA, **l*:** (And Research Area(Join R works in Chris Manning))

What papers have C. Manning written in EMNLP?
l: (And Papers (And Join R writes C. Manning) (Join published at EMNLP))) **a:** Glove

What is Chris Manning's research Area?
l: NK, **a:** NA

[IID] What is R. Socher's research area?
l: NK, **a:** NA
[Full Z-Shot] Which country is Stanford in?
l: NK, **a:** NA
[Partial Z-shot] What papers have US researchers written in EMNLP?
l: NK, **a:** NA

Figure 1: (A) KB schema and facts. Elements in red are part of 'ideal' KB but missing in the given KB for QA. An answerable question is shown for this KB with logical form l in s-expression, answer a and path (shaded blue). (B) 5 types of unanswerable questions for provided KB, with actual logical forms l, answers a and ideal logical forms l^* with missing KB elements in red (3-5). (C) Illustration of 3 different types of unanswerability scenario in test.

In summary, our contributions are as follows. (a) We motivate and introduce the task of detecting answerabilty for KBQA. (b) We create GrailQAbility, which is the first benchmark for KBQA with unanswerable questions. (c) Using experiments and analysis on GrailQAbility with three state-of-the-art KBQA models , we identify aspects of unanswerability that these models struggle to identify. We release code and data for further research.[1]

2 KBQA with Answerability Detection

A *Knowledge Base* (KB) (also called Knowledge Graph) G contains a *schema* S (or ontology) with *entity types* (or types) T and *relations* R defined over pairs of types, which we together refer to as *schema elements* of the KB. The types in T are often organized as a hierarchy. It also contains *entities* E as instances of types, and *facts* (or triples) $F \subseteq E \times R \times E$, which we together refer to as *data elements* of the KB. The top layer of Fig. 1(A) shows example schema elements, while the bottom layer shows entities and facts. In *Knowledge Base Question Answering* (KBQA), we are given a *question* q written in natural language which needs to be translated to a *logical form* (or query) l that executes over G to yield a set of *answers* A. Different logical forms, SPARQL (Yih et al., 2016b), s-expressions (Gu et al., 2021), programs (Cao et al., 2022b), etc., have been used in the KBQA literature. We concentrate on *s-expressions* (Gu et al., 2021), which employ set-based semantics and functions with arguments and return values as sets. These can be easily translated to KB query languages such as SPARQL, and provide a balance between readability and compactness (Gu et al., 2021). We call a logical form *valid* for G if it executes over G without an error. On successful execution, a logical form traces a *path* in the KB leading to each answer. Fig. 1(A) shows an example query with a valid logical form (using s-expression) and the path traced by its execution.

We define a question q to be **answerable** for a KB G, if **(a)** q admits a valid logical form l for G, AND **(b)** l returns a *non-empty* answer set A when executed over G. The example question in Fig. 1(A) is answerable for the shown KB. The *standard* KBQA task over a KB G is to output the answer A, and optionally the logical form l, given a question q, *assuming q to be answerable for G*.

Most recent KBQA models (Ye et al., 2022;

[1] https://github.com/dair-iitd/GrailQAbility.git

Chen et al., 2021) are trained with questions and gold logical forms. Other models directly generate the answer (Sun et al., 2019; Saxena et al., 2022). Different train-test settings have been explored and are included in benchmark KBQA datasets (Gu et al., 2021). For a question q, let S_q denote the schema elements in the logical form for q. Given a training set Q_{tr}, a test question q is labelled *iid* if it follows the distribution for questions in Q_{tr}, and contains only schema elements seen in train $S_q \subseteq S_{Q_{tr}}$ (we have overloaded notation to define $S_{Q_{tr}}$). Alternatively, a test question q is labelled *zero shot* if it involves at least one unseen schema element ($S_q \nsubseteq S_{Q_{tr}}$). Finally, test question q involves *compositional generalization* if $S_q \subseteq S_{Q_{tr}}$ but the specific logical form for q does not match that for any $q' \in Q_{tr}$.

By negating the above answerability definition, we define a question q to be **unanswerable** for a KB G if **(a)** q does not admit a valid logical form l for G, or **(b)** the valid l when executed over the G returns an empty answer. Clearly, meaningless and out-of-scope questions for a KB are unanswerable. Even for a meaningful question, unanswerability arises due to incompleteness (in data or schema) in G. Such questions admit an 'ideal KB' G^* for which q has a valid ideal logical form l^* which executes on G^* to generate a non-empty ideal answer a^*. The available KB G lacks one or more schema or data elements making q unanswerable. Fig. 1(A) illustrates an available KB, with missing elements with respect to the ideal KB shown in red. In Fig. 1(B), questions 1-2 yield valid queries for the available KB but missing facts lead to empty answers, while questions 3-5 lack schema elements for valid queries.

The task of **KBQA with answerability detection**, given a question q and an available KB G, is to **(a)** appropriately label the answer A as NA (No Answer) or the logical form l as NK (No Knowledge, i.e., query not possible) when q is unanswerable for G, or **(b)** generate the correct non-empty answer A and valid logical form l when q is answerable for G. The training set may now additionally include unanswerable questions labeled appropriately with $A =$ NA or $l =$ NK. Note that training instances do not contain 'ideal' logical forms for the unanswerable questions that have $l =$ NK.

Mirroring answerable questions, we define different train-test scenarios for unanswerable questions as well. An *iid unanswerable* question in test follows the same distribution as unanswerable questions in train, and all missing KB elements (schema elements in its ideal logical form and missing data elements in its ideal paths) are encountered in train unanswerable questions associated with the same category of incompleteness. For example, the missing schema element *Research Area* for the first test question in Fig. 1(C) is covered by the second train question. In contrast, a *zero-shot unanswerable test question* involves at least one missing KB element (schema element in its ideal logical form or data element in its paths) that is not part of any unanswerable question in train associated with same category of incompleteness. E.g., the missing schema elements (*located in* and *works at*) for the second and third test questions in Fig.1(C) are not covered by any unanswerable question in train. We further define two sub-classes, partial and complete zero-shot, for zero-shot unanswerable questions, but for clarity, discuss these in Sec. 5.

3 GrailQAbility: Extending GrailQA with Answerability Detection

In this section, we describe the creation of a new benchmark dataset for KBQA with unanswerable questions. In a nutshell, we start with a standard KBQA dataset containing only answerable questions for a given KB. We introduce unanswerability in steps, by deleting schema elements (entity types and relations) and data elements (entities and facts) from the given KB. We mark questions that become unanswerable as a result of each deletion with appropriate unanswerability labels. We control the percentage of questions that become unanswerable as a result of each type of deletion.

Many complications arise in this. (a) Deletion of different KB elements affect answerability differently. Some affect logical forms and answers, while others affect answers only. (b) The same KB element potentially appears in paths or logical forms of multiple questions. (c) KB elements cannot be deleted independently – entity types are associated with relations and entities, while relations and entities are associated with facts. (d) Questions with multiple answers remain answerable until the fact paths to *all* of these answers have been broken by deletions. (e) Choosing KB elements to delete uniformly at random does not resemble incompleteness in the real world.

We address these issues as follows. (a-b) We iterate over the 4 categories of KB elements to be

Dataset	#Q	#LF	#D	#R	#T	#E	Q. Type		Test Scenarios	
							A	U	A	U
GrailQA	64,331	4969	86	3720	1534	32,585	✓	✗	I, C, Z	✗
GrailQAbility	50,507	4165	81	2289	1081	22,193	✓	✓	I, C, Z	I, Z

Table 1: Statistics for GrailQA and GrailQAbility. #Q is no. of questions, #LF no. of unique canonical logical forms, #D no. of domains, #R, #T, #E no. of relations, types and entities, A and U denote answerable and unanswerable questions. I, C, and Z denote IID, compositional and zero-shot.

deleted, efficiently identify affected questions for a deleted schema element using an index, tag these with the deleted type, and appropriately relabel their logical forms or answers. We stop when specific percentages of questions are unanswerable for each category. (c) We delete different types of KB elements in an appropriate sequence – entity types, followed by relations, entities and finally facts. (d) We track remaining fact paths for questions and mark a question as unanswerable only when all paths are broken by KB deletions. (e) When sampling KB elements to delete, since "better known" KB elements are less likely to be missing, we incorporate the inverse popularity of an element in the original KB in the sampling distribution. Additionally, we only consider those elements present in still valid logical forms and paths for the questions in the dataset. Next, we describe the specifics for individual KB element categories.

Fact Deletion: Dropping a KB fact can break the path of one or more answers for a question but cannot affect the logical form. Answers whose paths are broken are removed from the answer list of the question. If the answer list becomes empty as a result of a fact drop, we set its answer to NA but leave its logical form unchanged. In Fig.1(B), deleting (*C. Manning, works at, Stanford*) makes Q1 unanswerable.

Entity Deletion: To delete an entity from the KB, we first delete all its associated facts, and then drop the entity itself. Deleting facts affects answerability of questions as above, as for Q2 in Fig.1(B). Deleting an entity additionally affects answerability of questions whose logical form contains that entity as one of the mentioned entities. This happens for Q3 in Fig.1(B) when entity *R. Socher* is deleted. For such questions, the logical form also becomes invalid, and we set it as NK.

Relation Deletion: To delete a relation, we first drop all facts associated with it, and then drop the relation itself from the schema. Deleting facts makes some questions unanswerable as above, and we set their answers to be NA. Deleting the relation additionally affects the logical form of some questions, and we set their logical forms to be NK. This happens for Q4 in Fig. 1(B) on deleting the *located in* relation.

Entity Type Deletion: Entities are often tagged with multiple types in a hierarchy (e.g *C. Manning* may be *Researcher* and *Person*). After deleting an entity type from the KB schema, we also delete all entities e that are associated *only* with that type. We further delete all relations associated with the type. For Q5 in Fig.1(b), the logical form becomes invalid on deleting the *Research Area* entity type. For an affected question, we set its answer as NA and its logical form as NK.

Split	A	U	
		NK	NA
Train	23,933	7110	4240
Dev	3399	1064	595
Test	6808	2162	1196

Table 2: GrailQAbility: Train, Dev and Test Splits

GrailQAbility Dataset: We make use of GrailQA (Gu et al., 2021), which is one of the largest and most diverse KBQA benchmark based on Freebase but contains only answerable questions, and create a new benchmark for KBQA with answerability detection. We call this GrailQAbility (GrailQA with Answerability). We make this dataset public. Aligning with earlier QA datasets with unanswerability (Rajpurkar et al., 2018; Sulem et al., 2022; Choi et al., 2018; Raina and Gales, 2022), we keep the total percentage of unanswerable questions as 33%, splitting this nearly equally (8.25%) between deleted entity types, relations, entities and facts.

Train-Test Split: Since the test questions for GrailQA are unavailable, we use the train and dev questions. We keep aside the compositional and zero shot questions from dev as the compositional and zero shot *answerable* questions in our new dev and test set. We then combine the train and iid dev questions, introduce unanswerability into these by running the 4 categories of deletion algorithms in sequence, and split these to form the new train and iid test+dev (both answerable and unanswerable) and zero shot unanswerable test+dev questions. The unanswerable questions in test and dev contain 47% iid, and 53% zero-shot. Statistics for GrailQAbility and GrailQA are compared in Tab. 1. Sizes of the different splits are shown in Tab. 2. Details on dataset creation are in appendix (A.1).

4 Experimental Setup

KBQA Models: Among state-of-the-art KBQA models, we pick RnG-KBQA (Ye et al., 2022), ReTraCk (Chen et al., 2021) and TIARA (Shu et al., 2022). These report state-of-the-art results on GrailQA as well as on WebQSP (Berant et al., 2013; Yih et al., 2016a; Talmor and Berant, 2018) - the two main benchmarks. On the GrailQA leader board,[2] these are the top three published models with available code (at the time of submission). Since these generate logical forms, we expect these to be more robust to data level incompleteness than purely retrieval-based approaches (Saxena et al., 2020; Das et al., 2021; Zhang et al., 2022; Mitra et al., 2022; Wang et al., 2022).

RnG-KBQA (Ye et al., 2022) first uses a BERT-based (Devlin et al., 2019) ranker to select a set of candidate logical forms for a question by searching the KB, and then a T5-based (Raffel et al., 2020a) model generates the logical form using the question and candidates. **ReTraCk** (Chen et al., 2021) also uses a rank and generate approach, but uses a dense retriever to retrieve schema elements for a question, and grammar-guided decoding using an LSTM (Hochreiter and Schmidhuber, 1997) to generate the logical form using the question and retrieved schema items. **TIARA** (Shu et al., 2022) combines the retrieval mechanisms of the first two models to include both candidate logical forms as well as candidate schema elements from the KB. It then uses constrained decoding like ReTraCk but using T5 (Raffel et al., 2020b). All three models use entity disambiguation to find KB entities mentioned in a question and also check execution for generated logical forms.

Adapting for Answerability: We use existing code bases[3][4] [5] of these models, and adapt these in two ways — thresholding and training with unanswerability. ReTraCk and TIARAreturn empty logical form when execution fails, which we interpret as $l =$ NK prediction. For all models, we additionally introduce thresholds for entity disambiguation and logical form generation, and take the prediction to be NK when the scores for entity linking and logical form are less than their corresponding thresholds. These thresholds are tuned using the validation set. We train the models as in their original setting with only the answerable subset of training questions, leaving out the unanswerable questions (**A training**). We also train by including both the answerable and unanswerable questions in the training data (**A+U training**). More details are in appendix (A.3).

Evaluation Measures: To evaluate a model's performance for detecting unanswerability, we primarily focus on the correctness of the logical form. We compare the predicted logical form with the gold-standard one using exact match (EM) (Ye et al., 2022). As it is ultimately a QA task (and other systems may produce answers without generating logical forms), we also perform direct answer evaluation. Since in general a question may have multiple answers, we evaluate predicted answers using precision, recall and F1. In regular answer evaluation (R), we compare the predicted answer (which could be NA) with the gold answer in the modified KB, as usual. Specifically for unanswerability, we also consider lenient answer evaluation (L), where we account for the gold answer in the original (ideal) KB as well, and also give credit to models which are able to recover this answer, perhaps via inference. As an example, for the second test question in Fig. 1(C), R-evaluation only rewards NA as answer, whereas L-evaluation rewards both NA and *USA* as perfect answers. Details of evaluation measures are in appendix (A.2).

5 Results and Discussion

We structure our discussion of experimental results around four research questions.

[2] https://dki-lab.github.io/GrailQA/

[3] https://github.com/salesforce/rng-kbqa

[4] https://github.com/microsoft/KC/tree/main/papers/ReTraCk

[5] https://github.com/microsoft/KC/tree/main/papers/TIARA

Train	Model	Overall			Answerable			Unanswerable		
		F1(L)	F1(R)	EM	F1(L)	F1(R)	EM	F1(L)	F1(R)	EM
A	RnG-KBQA	67.8	65.6	51.6	78.1	78.1	74.2	46.9	40.1	5.7
	RnG-KBQA+T	67.6	65.8	57.0	71.4	71.3	68.5	59.9	54.5	33.6
	ReTraCk	69.2	67.0	50.7	67.0	66.9	62.4	73.8	67.2	27.1
	ReTraCk+T	69.9	67.9	52.0	65.3	65.3	61.2	79.3	73.2	33.4
	TIARA	77.1	75.0	56.0	**82.9**	**82.8**	**79.2**	65.4	59.0	9.0
	TIARA+T	76.5	74.8	63.4	76.9	76.8	74.1	75.9	70.8	41.8
A+U	RnG-KBQA	80.5	79.4	68.2	75.9	75.9	72.6	89.7	86.4	59.4
	RnG-KBQA+T	77.8	77.1	67.8	70.9	70.8	68.1	92.0	89.8	67.2
	ReTraCk	69.7	68.4	56.5	61.4	61.3	57.3	86.5	82.8	54.7
	ReTraCk+T	70.3	69.1	56.6	61.2	61.1	57.1	88.7	85.1	55.5
	TIARA	**83.9**	**82.9**	69.7	81.0	81.0	78.3	89.9	86.8	52.3
	TIARA+T	81.7	81.1	**72.6**	76.0	76.0	74.0	**93.3**	**91.3**	**69.8**

Table 3: Performance of different models on GrailQAbility over all, answerable and unanswerable questions. EM is exact match on logical forms and F1(L) and F1(R) are lenient and regular evaluations of answers. A and A+U indicate training with only answerable questions and with both answerable and unanswerable questions. Models with suffix +T have additional thresholds for entity disambiguation and logical form fine-tuned on dev set.

Train	Model	Schema Element Missing						Data Element Missing			
		Type		Relation		Mention Entity		Other Entity		Fact	
		F1(R)	EM	F1(R)	EM	F1(R)	EM	F1(R)	EM	F1(R)	EM
A	RnG-KBQA	40.1	0.0	44.2	0.0	27.4	0.0	45.1	13.5	46.0	16.8
	RnG-KBQA+T	55.5	49.5	57.1	46.6	44.7	40.3	56.0	11.5	58.6	13.9
	ReTraCk	71.1	34.8	59.3	18.9	80.7	63.7	72.6	11.2	64.4	11.9
	ReTraCk+T	75.7	47.9	64.9	28.8	83.5	70.3	81.0	10.9	72.3	12.0
	TIARA	57.7	0.0	56.9	0.0	51.9	0.0	65.8	22.4	65.8	26.3
	TIARA+T	68.0	56.5	69.5	48.7	74.4	62.6	70.9	18.5	74.0	20.9
A+U	RnG-KBQA	91.6	75.8	86.4	66.6	87.6	72.0	84.0	37.5	82.4	39.1
	RnG-KBQA+T	**93.4**	**86.8**	89.7	**85.5**	92.1	**89.6**	87.1	30.8	86.0	32.5
	ReTraCk	89.6	82.2	86.4	74.4	90.3	85.9	79.0	9.8	71.7	10.8
	ReTraCk+T	90.6	83.1	87.8	76.0	91.2	86.8	83.2	9.8	76.4	10.8
	TIARA	83.7	50.6	83.6	40.5	88.7	52.5	**91.6**	**62.5**	90.9	**63.7**
	TIARA+T	88.9	80.3	**90.9**	77.1	**94.7**	84.6	91.6	53.2	**92.6**	53.4

Table 4: Performance for different KBQA models for subsets of questions affected by different types of KB incompleteness. Note that missing mention entities result in invalid logical form and other missing entities lead to valid logical form with no answer. Names have the same meanings as in Tab. 3.

Train	Model	IID		Zero-Shot	
		F1(R)	EM	F1(R)	EM
A+U	RnG-KBQA	91.9	73.3	81.7	47.1
	RnG-KBQA+T	94.3	75.9	85.9	59.5
	ReTraCk	88.7	66.5	77.7	44.4
	ReTraCk+T	90.1	66.6	80.7	45.7
	TIARA	90.9	63.4	83.1	42.5
	TIARA+T	**94.5**	**78.7**	**88.5**	**62.0**

Table 5: Performance of different models for unanswerable IID and zero-shot test scenarios in GrailQAbility. Names have the same meanings as in Tab. 3.

Train	Model	Full Z-Shot		Partial Z-Shot	
		F1(R)	EM	F1(R)	EM
A+U	RnG-KBQA	87.2	75.9	78.0	40.0
	RnG-KBQA+T	89.7	**86.7**	**83.1**	**71.0**
	ReTraCk	86.2	65.0	73.6	54.5
	ReTraCk+T	88.2	67.0	75.6	56.7
	TIARA	85.7	41.9	73.7	20.0
	TIARA+T	**90.6**	72.4	82.0	64.0

Table 6: Performance of different models for partial zero-shot and full-zero test scenarios in GrailQAbility. Names have the same meanings as in Tab. 3.

RQ1. How do state-of-the-art KBQA models perform for answerability detection?

Tab. 3 shows high-level performance for the three models on answerable and unanswerable questions. We observe the following.

(A) When training with only answerable questions (A training), all models perform poorly for unanswerable questions in terms of EM, ReTraCk being better than the other two.

(B) Performance improves for unanswerable questions with thresholding and A+U training but remains below the skyline for answerable questions with A training. The gap is ∼7 pct points for RnG-KBQA and ReTraCk and ∼9 for TIARA.

(C) Not surprisingly, improvement for unanswerable questions comes at the expense of answerable question performance. The best overall performance (72.6 EM for TIARA) is ∼6.5 percentage points lower than the best answerable performance (79.2 EM for TIARA). Further, we observed that answerable performance is affected by thresholding (across iid, compositional and zero-shot settings) for all models. This is also the case for the A+U training in the zero-shot setting. More details can be found in Tab.9 in appendix. The reason is that for both forms of adaptation, the models incorrectly predict $l = \text{NK}$ for answerable questions.

(D) Unlike for answerable questions, there is a very large gap between EM and F1(R) for unanswerable questions. This is because correct NA (no answer) predictions are often associated with spurious logical form predictions, for all three models but for different reasons. We discuss this further under RQ4.

(E) Performance is better (by about 2-4 percentage points) with lenient answer evaluation than with the regular counterpart. We found that this is often because the models generate logical forms with schema elements similar to the deleted ones, and return as a result subsets or supersets of the old answer instead of NA. As one example, the question *Which football leagues share the same football league system as Highland Football League?* has 7 answers, but becomes unanswerable when the relation *soccer.football_league_system.leagues* is missing. The model answers a different question - *Which football leagues play the same sport as Highland Football League* - by substituting the missing relation with *sports.sports_league.sport*, and retrieves 152 answers, one of which, *Scottish Premier League*, is also in the original answer.

RQ2. Are different forms of KB incompleteness equally challenging?

In Tab. 4, we break down performance for unanswerable questions according to different forms of KB incompleteness. Note that we have decomposed entity deletions further into deletion of mentioned entities (which affect the logical form) and other entities in the path (which affect only the answer paths). The following are the main takeaways.

(A) Performance (EM) is significantly poorer for all forms of missing data elements than missing schema elements, even after thresholding and retraining. TIARA is an exception and performs better for missing data elements with A+U training.

(B) A+U training significantly boosts performance for missing schema elements but not for missing data elements. This is because RnG-KBQA and ReTraCk learn to generate logical forms involving schema elements. As a result, schema-level patterns are easier to learn for unanswerable questions with missing schema elements than those with missing data elements. Secondly, these two rely on retrieved data paths to generate logical forms. When relevant data elements are missing, the models fail to retrieve any familiar input pattern and predict $l = \text{NK}$. The interesting exception is TIARA. By virtue of generating logical forms conditioned on both retrieved paths and schema elements and removing data path constraints during decoding, it learns to generate correct logical forms for missing data elements. But this also leads to the generation of syntactically valid but incorrect logical forms for missing schema elements. However, these typically have low score and performance for missing schema elements improves with thresholding.

(C) Gap between EM and F1(R) is small for missing schema elements ($l = \text{NK}$) and extremely large for missing data elements ($l \neq \text{NK}$), with the exception of A+U trained TIARA. Also, thresholding hurts performance for missing data elements. This is because questions with missing data elements have valid logical forms, and thresholding and A+U training produce $l = \text{NK}$ predictions which are themselves incorrect but imply $A = \text{NA}$ which is correct. Thus we get correct $A = \text{NA}$ predictions for the wrong reason.

RQ3. How difficult is zero shot generalization compared to iid for unanswerable questions?

Recall that a zero-shot unanswerable test instance involves one or more missing KB elements that are not encountered in any unanswerable train instance with the same category of incompleteness. Note that the definitions of iid and zero-shot make use of unanswerable training instances, so that only A+U training makes sense for this comparison.

(A) The decomposition of unanswerable performance in terms of iid and zero-shot subsets is shown in Tab. 5. As expected, iid performance is better than zero-shot for all models. The best performance is for TIARA+T (EM 78.7 for iid, 62 for zero-shot) which is marginally better than RnG-KBQA+T.

(B) However, more interesting insights arise for unanswerability from a deeper drill-down of zero-shot instances. We define a zero-shot instance to be *full zero-shot* when it does not involve any schema element seen in logical forms of answerable questions in train. The second test question in Fig. 1(C), involving the missing relation *located in* is an example. In contrast, a *partial zero-shot* unanswerable question is part "seen answerable" in addition to being part "unseen unanswerable". Specifically, its ideal logical form also contains at least one schema element seen for answerable questions in train. The third test question in Fig. 1(C) is an example. The *located in* and *works at* relations are "new unseen", while *writes* and *published at* are "seen" in the first train question, which is answerable. In GrailQAbility, zero-shot instances due to schema drop are roughly 75% partial zero-shot and 25% full zero-shot. Tab.6 shows full zero-shot and partial zero-shot performance for unanswerable questions. We see that all models find full-zero-shot to be significantly easier than partial zero-shot. For RnG-KBQA+T, which is the best model, there is a 15.7 percentage point difference in EM. The reason is that partial zero-shot unanswerable questions have some KB elements seen during training (in answerable contexts), and some zero-shot KB elements (that make the question unanswerable) unseen during training. This confuses the models, which often labels these as answerable. The full zero-shot instances do not have any similarity with training answerable questions and are less confusing.

We have not considered compositional generalization for unanswerable questions. We may define a compositional unanswerable question as one that contains more than one missing KB element in its ideal logical form or in its ideal paths, all of which have appeared in unanswerable training instances, but not all in the same instance. We hypothesize that detecting unanswerability in this scenario should only be hard as for IID unanswerability. We plan to validate this experimentally in the future. Additionally, since missing data elements constitute an important aspect of unanswerability for KB questions, we have included missing data elements in our definitions of iid and zero-shot unanswerability. However, distributions at the level of KB data elements cannot realistically be learnt. Therefore alternative definitions for these based only on schema elements may be more practical.

RQ4. How do RnG-KBQA, ReTraCk and TIARA compare for unanswerable questions?

On GrailQA (answerable questions with A-training), RnG-KBQA outperforms ReTraCk (Ye et al., 2022) and TIARA outperforms both (Shu et al., 2022), and we see the same pattern in GrailQAbility. In the context of unanswerable questions, we make the following observations.

(A) RnG-KBQA outperforms ReTraCk with thresholding and retraining by a similar margin as for answerable questions (12 pct points). However, TIARA outperforms RnG-KBQA by a much smaller margin for unanswerable questions (2.6 pct points) compared to answerable ones (5 pct points).

(B) With just A training, ReTraCk performs better than the other two models for unanswerable questions. This is due to the difference in fallback strategies when execution fails for generated logical forms. ReTraCk's fallback acknowledges unanswerability — it returns empty logical form. On the other hand, RnG-KBQA's fallback assumes answerability. It returns logical forms corresponding to top-ranked paths or the nearest neighbor in the training set. In settings with unanswerability, ReTraCk naturally performs better. TIARA also has the ability to return empty logical forms, but this happens rarely — when execution fails for generated logical forms and additionally the ranker output is empty (i.e. no enumerations)).

(C) We find that all models generate spurious logical forms, but for different reasons. RNG-KBQA hallucinates relations that do not exist in the KB. For example, when the relation *cricket_tournament_event.tournament* is deleted, RnG-KBQA substitutes that with the imaginary relationship *cricket_tournament_event.championship*. ReTraCk and TIARA avoid this by virtue of constrained decoding, but incorrectly replace missing relations with other semantically or lexically relevant relations for the same entity. For example, for the question *Which ac power plug standard can handle more than 50 Hz?*, when the *mains_power.ac_frequency* relation is missing, ReTraCk incorrectly replaces that with *power_plug_standard.rated_voltage*.

(D) With A+U training and thresholding, ReTraCk performs almost at par with RnG-KBQA for missing schema elements. But it performs significantly worse for missing data elements, for which its performance is hurt by these adaptations. This is because ReTraCk's constrained decoding forces it to always generate l = NK in the absence of valid answer paths, which cannot be alleviated by additional training. Using decoding with syntactic constraints, TIARA establishes the best balance between missing schema and data elements and outperforms the other two models by a huge margin for missing data elements. However for missing schema elements RnG-KBQA is the best individual model outperforming TIARA by 5-8 pct points.

6 Related Work

KBQA models: There has been extensive research on KBQA in recent years. Retrieval based approaches (Saxena et al., 2020; Zhang et al., 2022; Mitra et al., 2022; Wang et al., 2022; Das et al., 2022) learn to identify paths in the KB starting from entities mentioned in the question, and then score and analyze these paths to directly retrieve the answer. Query generation approaches (Cao et al., 2022c; Ye et al., 2022; Chen et al., 2021; Das et al., 2021) learn to generate a logical form or a query (e.g in SPARQL) based on the question, which is then executed over the KB to obtain the answer. Some of these retrieve KB elements first and then use these in addition to the query to generate the logical form (Ye et al., 2022; Chen et al., 2021). Cao et al. (2022c) first generate a KB independent program sketch and then fill in specific arguments by analyzing the KB. All these models have so far only been evaluated for answerable questions. There is work on improving accuracy of QA over incomplete KBs (Thai et al., 2022; Saxena et al., 2020), but these do not address answerability.

Answerability in QA: Answerability has been explored for extractive QA (Rajpurkar et al., 2018), conversational QA (Choi et al., 2018; Reddy et al., 2019), boolean (Y/N) QA (Sulem et al., 2022) and MCQ (Raina and Gales, 2022). While our work is motivated by these, the nature of unanswerable questions is very different for KBs compared to unstructured contexts. Also, KBQA models work differently than other QA models. These retrieve paths and KB elements to prepare the context for a question. Relevant context is then pieced together to generate a logical query rather than the answer directly. We find that this makes them more prone to mistakes in the face of unanswerability.

QA Datasets and Answerability: Many benchmark datasets exist for KBQA (Gu et al., 2021; Yih et al., 2016a; Talmor and Berant, 2018; Cao et al., 2022a), but only contain answerable questions. QALD (Perevalov et al., 2022) is a multilingual dataset containing "out-of-scope" questions that may be considered unanswerable according to our definition. However, the number of such questions is very small (few tens in different versions of the dataset), which hinders any meaningful bench-marking. It also does not have any finer categorization of such questions.

Unanswerable questions have been incorporated into other QA datasets (Rajpurkar et al., 2018; Sulem et al., 2022; Reddy et al., 2019; Choi et al., 2018; Raina and Gales, 2022). These are typically achieved by pairing one question with the context for another question. Introduction of unanswerability in the dataset in a controlled manner is significantly more challenging in KBQA, since the KB is the single shared context across questions and across train and test.

7 Conclusions and Discussion

We have introduced the task of detecting answerability when answering questions over a KB. We have released GrailQAbility[1] as the first benchmark dataset for KBQA with unanswerable questions, along with extensive experiments on three KBQA models. We find that no model is able to replicate its answerable performance for the unanswerable setting even with appropriate retraining and thresholding, though both these methods of adaptation help in improving performance substantially.

Further, we find that there is a trade-off between robustness to missing schema elements and missing data elements. The models find schema-level incompleteness easier to handle while data-level incompleteness substantially affects the models that enforce data-level constraints while decoding. Another observation is that the models get quite confused for those unanswerable questions that contain a schema element seen in an answerable train question, along with a missing schema element that is not seen at training. Finally, while TIARA turns out to be the best overall model, different models find different categories of unanswerability to be more challenging. This suggests that new KBQA models will need to combine architectural aspects of different existing models to best handle unanswerability. We believe that our dataset and observations will inspire research towards developing more robust and trustworthy KBQA models.

Acknowledgements

Prayushi is supported by a grant from Reliance Foundation. Mausam is supported by grants by TCS, Verisk, and the Jai Gupta chair fellowship by IIT Delhi. He also acknowledges travel support from Google and Yardi School of AI travel grants. We thank the IIT Delhi HPC facility for its computational resources.

Limitations

Our dataset creation process - introducing unanswerability into a dataset of answerable KB questions by deleting KB elements - limits the nature of unanswerable questions. All of these become answerable by completing the provided KB. However, other kinds of unanswerability exists. Questions may involve false premise, for example, *C. Manning works at which European University?*, or may not even be relevant for the given KB. We will explore these in future work.

Complete training and inference for each model with our dataset size takes 50-60 hours. As a result, generating multiple results for the same models in the same setting was not possible and our results are based on single runs. However, using multiple runs with smaller dataset sizes we have seen that the variance is quite small. Also, the dataset creation involves sampling KB elements for deletion and as such the generated dataset is one sample dataset with unanswerability. This is unfortunately unavoidable when creating one benchmark dataset.

Risks

Our work does not have any obvious risks. In fact, addressing answerability reduces the risk of KBQA models confidently generating incorrect answers in spite of lack of knowledge.

References

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*.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022a. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022b. KQA Pro: A large diagnostic dataset for complex question answering over knowledge base. In *ACL'22*.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022c. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. ReTraCk: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Rajarshi Das, Ameya Godbole, Ankita Naik, Elliot Tower, Manzil Zaheer, Hannaneh Hajishirzi, Robin Jia, and Andrew Mccallum. 2022. Knowledge base question answering by case-based reasoning over subgraphs. In *Proceedings of the 39th International Conference on Machine Learning*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *North American Chapter of the Association for Computational Linguistics*.

Sayantan Mitra, Roshni Ramnani, and Shubhashis Sengupta. 2022. Constraint-based multi-hop question answering with knowledge graph. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*.

Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Re. 2020. Bootleg: Chasing the tail with self-supervised named entity disambiguation. *arXiv preprint arXiv:2010.10363*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Vatsal Raina and Mark Gales. 2022. Answer uncertainty and unanswerability in multiple-choice machine reading comprehension. In *Findings of the Association for Computational Linguistics: ACL 2022*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. 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)*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Elior Sulem, Jamaal Hay, and Dan Roth. 2022. Yes, no or IDK: The challenge of unanswerable yes/no questions. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Dung Thai, Srinivas Ravishankar, Ibrahim Abdelaziz, Mudit Chaudhary, Nandana Mihindukulasooriya, Tahira Naseem, Rajarshi Das, Pavan Kapanipathi, Achille Fokoue, and Andrew McCallum. 2022. Cbr-ikb: A case-based reasoning approach for question answering over incomplete knowledge bases.

Yu Wang, Vijay Srinivasan, and Hongxia Jin. 2022. A new concept of knowledge based question answering (KBQA) system for multi-hop reasoning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System*

Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016a. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016b. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

A Appendix

A.1 Details of Dataset Creation

In this section, we describe more details of the dataset creation process.

We assume the given KB to be the ideal KB G^* and the given logical forms and answers to be the ideal answers a^* and ideal logical forms l^* for the questions Q. We then create a KBQA dataset Q_{au} with answerable and unanswerable questions with an 'incomplete' KB G_{au} by iteratively dropping KB elements from G^*. Prior work on QA over incomplete KBs has explored algorithms for dropping facts from KBs (Saxena et al., 2020; Thai et al., 2022). We extend this for all categories of KB elements (type, relation, entity and fact) and explicitly track and control unanswerability. At step t, we sample a KG element g from the current KB G_{au}^{t-1}, identify all questions q in Q_{au}^{t-1} whose current logical form l^{t-1} or path p^{t-1} contains g, and remove g from it. Since q may have multiple answer paths, this may only eliminate some answers from a^{t-1} but not make it empty. If g eliminate all answers from a^{t-1}, thereby making q unanswerable. If q becomes unanswerable, we mark it appropriately (with a^t =NA or l^t =NK) and update $G_{au}^t = G_{au}^{t-1} \setminus \{g\}$. This process is continued until Q^t contains a desired percentage p_u of unanswerable questions.

One of the important details is sampling KB element g to drop. In an iterative KB creation or population process, whether manual or automated, popular KB elements are less likely to be missing at any time. Therefore we sample g according to inverse popularity in G^*. However, the naive sampling process is inefficient since it is likely to affect the same questions across iterations or not affect any question at all. So, the sampling additionally considers the presence of g for Q_{au}^t — the set of questions in Q_{au}^t whose current logical form or answer paths contains g. Unlike schema elements, for selecting data elements to drop, we consider all data elements to be equally popular.

Next we describe how we drop all categories of KB elements in the same dataset.

Combining Drops: Our final objective is a dataset Q_{au} that contains p_u percentage of unanswerable questions with contributions p_u^f, p_u^e, p_u^r and p_u^t from the four categories of incompleteness. Starting with the original questions Q^* and KB G^*, we execute type drop, relation drop, entity drop and fact drop with the corresponding percentage in sequence, in each step operating on the updated dataset and KB. For analysis, we label questions with the drop category that caused unanswerability. Note that a question may be affected by multiple categories of drops at the same time.

GrailQA (Gu et al., 2021) only contains the SPARQL queries for the questions (in English language) and the final answers, but not the answer paths. To retrieve the answer paths, we modify the provided SPARQL queries to return the answer paths in addition to the final answer, and then execute these queries. In Tab.7, we include detailed statistics for unanswerable questions in GrailQAbility. We will release the GrailQAbility under the same license as GrailQA i.e., CC BY-SA 4.0.

A.2 Lenient Answer Evaluation

Under lenient evaluation (L) for a given question, we calculate precision and recall w.r.t both gold answer in Q_{au} and ideal answer in Q. We consider maximum over Q_{au} and Q for precision and recall, and then calculate F1 as usual for calculating F1(L).

Split	Type Drop			Relation Drop			Entity Drop				Fact Drop	
	IID	Z-Shot		IID	Z-Shot		IID		Z-Shot		IID	Z-Shot
	NK	P (NK)	F (NK)	NK	P (NK)	F (NK)	NA	NK	NA	NK	NA	NA
Train	2667	0	0	2780	0	0	1288	1663	0	0	2952	0
Dev	211	154	47	211	146	55	91	89	87	151	176	241
Test	422	330	96	425	298	107	178	193	179	291	352	487

Table 7: Statistics for unanswerable questions (P: Partial and F: Full) in GrailQAbility due to different types of KB incompleteness.

Split	IID	Compositional	Z-Shot
Train	23,933	0	0
Dev	1691	488	1220
Test	3386	981	2441

Table 8: Statistics for answerable questions in GrailQAbility.

A.3 Model Adaptation and Training Details

RnG-KBQA: RnG-KBQA (Ye et al., 2022) consists of four modules: Entity Linker, Entity Disambiguation, Ranker and Generator. We use the same training objective and base models for re-training of these components on GrailQAbility. Similar to GrailQA (Gu et al., 2021), for mention detection, we fine-tune a BERT-base-uncased model for 3 epochs with a learning rate of 5e-5 and a batch size of 32. For training the Entity Disambiguator, similar to RnG-KBQA, we fine-tuned a BERT-base-uncased (Devlin et al., 2019) model for 3 epochs with a learning rate of 1e-3 and a batch size of 16. We use a non-bootstrapped strategy for sampling negative logical forms during the training of the ranker and fine-tune a BERT-based-uncased model for 3 epochs with a learning rate of 1e-3 and a batch size of 2. As a generator, we fine-tune T5-base (Raffel et al., 2020a) for 10 epochs with a learning rate of 3e-5 and a batch size of 8. During inference with the generator, similar to RnG-KBQA, we use a beam size of 10 but due to the presence of NA questions in the test we do not perform execution augmented-inference. We compute the entity threshold τ_e and and logical form threshold τ_l based on disambiguation score and perplexity respectively by tuning on the validation set. During inference we use $\tau_e = -1.3890$ and $\tau_l = 1.0030$ for RnG-KBQA A and $\tau_e = -0.7682$ and $\tau_l = 1.0230$ for RnG-KBQA A+U.

RnG-KBQA takes the (question, logical form) pair as input during training where the valid logical form also contains information about the mentioned entities in the question. We train two RnG-KBQA based KBQA models, one with answerable questions and the other with a combination of answerable and unanswerable questions. During training with A+U, we train mention detection and entity disambiguation model with questions having valid logic form i.e., $l =$ NK, and perform entity linking for questions where $l \neq$ NK. And Generator is trained to predict “no logical form” for unanswerable questions with $l =$ NK and valid logical form for remaining training questions.

We use Hugging Face (Wolf et al., 2020), PyTorch (Paszke et al., 2019) for our experiments and use the Freebase setup specified on github [6]. We use NVIDIA A100 GPU with 20 GB GPU memory and 60 GB RAM for training and inference of RnG-KBQA on GrailQAbility which takes 60 hours.

ReTraCk: ReTraCk (Chen et al., 2021) includes three main components - retriever, transducer, and checker. Retriever consists of an entity linker that links entity mentions to corresponding entities in KB and a schema retriever that retrieves relevant schema items given a question. The entity linker has two stages - the first stage follows the entity linking pipeline described in (Gu et al., 2021) followed by a BOOTLEG (Orr et al., 2020) model used for entity disambiguation. We have used the pre-trained entity linker of ReTraCk. We remove the dropped entities from the predictions of the entity linker. The schema retriever leverages the dense retriever framework (Mazaré et al., 2018; Humeau et al., 2019; Wolf et al., 2020) for obtaining classes(types) and relations. Same as ReTraCk, we use pre-trained BERT-base-uncased model as a schema retriever and fine-tune it on GrailQAbility for 10 epochs with a learning rate of 1e-5. The best model is selected on basis of recall@top_k where top_k is 100 and 150 for types and relations respectively. We train two schema retriever models, one for A and one for A+U. For A, all answerable questions are used for training, while for A+U we use non-NK questions i.e. questions having only

[6] https://github.com/dki-lab/Freebase-Setup

Train	Model	IID			Compositional			Zero-Shot		
		F1(L)	F1(R)	EM	F1(L)	F1(R)	EM	F1(L)	F1(R)	EM
A	RnG-KBQA	85.5	85.4	83.2	65.9	65.9	60.2	72.7	72.7	67.3
	RnG-KBQA+T	79.0	79.0	77.3	58.8	58.8	54.5	65.8	65.8	61.9
	ReTraCk	79.6	79.5	75.6	63.1	63.1	55.4	51.0	51.0	46.8
	ReTraCk+T	79.0	78.9	75.2	61.6	61.6	53.9	47.8	47.8	44.5
	TIARA	88.9	88.8	86.8	**74.2**	**74.2**	65.9	**78.1**	**78.1**	**73.9**
	TIARA+T	84.1	84.0	82.6	67.7	67.7	60.9	70.6	70.6	67.5
A+U	RnG-KBQA	85.4	85.3	83.3	65.8	65.8	60.8	66.9	66.9	62.6
	RnG-KBQA+T	80.9	80.9	79.2	60.5	60.5	56.1	61.1	61.1	57.6
	ReTraCk	77.8	77.6	73.9	60.6	60.6	53.5	39.0	39.0	35.8
	ReTraCk+T	77.7	77.5	73.8	59.9	59.9	52.8	38.9	38.9	35.7
	TIARA	**89.1**	**89.0**	**87.3**	73.1	73.1	**68.7**	72.9	72.9	69.6
	TIARA+T	85.5	85.5	84.2	66.3	66.3	62.8	66.8	66.8	64.2

Table 9: Performance of different models for answerable IID, compositional, and zero-shot test scenarios in GrailQAbility. Names have the same meanings as in Tab. 3.

valid logical forms.

Transducer modules consist of a question encoder and a grammar-based decoder. ReTraCk uses a set of grammar rules for logical form. For NA training we have added a new grammar rule i.e. num → NK where NK is a terminal symbol representing No Knowledge. So for a question with no logical form, the sequence of grammar rules will be @start@ → num and num → NK. We have trained the transducer model with updated grammar rules for GrailQAbility. Training settings and hyperparameters are same as ReTraCk i.e. the BERT-base-uncased model with Adam optimizer and learning rate 1e-3, while learning rate for BERT is set to 2e-5. The best model is selected on basis of the average exact match calculated between predicted logical form and golden logical form. Additionally, ReTraCk uses a Checker to improve the decoding process via incorporating semantics of KB. It consists of 4 types of checks i.e; Instance level, Ontology level, real and virtual execution. We have modified the stopping criteria for real execution. ReTraCk's real execution terminates only when it finds a non-empty answer after query execution whereas we accept empty answers also after the execution of the query successfully (since unanswerable training involves empty answers). We compute the logical form threshold τ_l by tuning on the validation set. During inference we use $\tau_l = -6.5$ for ReTraCk A and $\tau_l = -7.5$ for ReTraCk A+U. We use NVIDIA V100 GPU with 32 GB GPU memory and 60 GB RAM for the training of ReTraCk on GrailQAbility which takes 50 hours. And we do inference on a CPU machine with 80GB RAM which takes 3 hours.

TIARA: TIARA (Shu et al., 2022) consist of four modules - Entity Retrieval, Schema Retriver, Exemplary Logical Form Retrieval and Generator. Entity Retrieval has three steps - mention detection, candidate generation, and entity disambiguation. They have used there own mention detector called SpanMD. But since SpanMD is not open sourced so as suggested by authors we have used PURE mention detector which has similar performance to SpanMD. Candidates are generated using FACC1 (Gabrilovich et al., 2013) and entity disambiguation pipeline is leveraged from (Ye et al., 2022). The logical form retrieval includes enumeration and ranking. It follows same methods as proposed in (Gu et al., 2021) and (Ye et al., 2022). So training process and hyperparameters for this module is same as described in RnG-KBQA section above. Schema retrieval is implemented by a cross-encoder using pretrained BERT-base-uncased model. The model is trained for 10 epochs and best model is selected on the basis of recall@top_k where k is 10 for both relations and classes. To train schema-retriever for A model we use all answerable questions while for A+U model we use questions with valid logical forms. Generator in TIARA takes following input - question, outputs of Entity Retrieval, Schema Retriver, Exemplary Logical Form Retrieval and outputs a logical form. Generation is performed by a transformer-based seq2seq model - T5(base) (Raffel et al., 2020a). The Generator is fine-tuned for 10 epochs with learning rate 3e-5 and batch size of 8. We have trained two Generator models - A and A+U. For A model, all answerable questions are used for training, and for A+U model we use all answerable and unanswerable questions for training. For unanswerable questions the model is trained to generate output as "no logical form". Similar to above models TIARA also performs beam search during inference with a beam size of 10. Additionally TIARA also performs constraint decoding to reduce generation errors on logical form operators and schema tokens. It uses a prefix trie to validate the sequence of tokens generated. After generation it is checked if the output is executable or not. Output is considered valid only if it executable (after constrained generation). Note: We consider executable queries with empty answers as valid query.

We use Hugging Face (Wolf et al., 2020), PyTorch (Paszke et al., 2019) for our experiments and use the Freebase setup specified on github [7].Training configurations for schema retriver are same as mentioned in ReTraCk and training configurations for Exemplary Logical Form Retrieval is same as mentioned in Rng-KBQA. We use NVIDIA A100 GPU with 40 GB GPU memory and 32 GB RAM for training TIARA Generator which takes around 8 hours for one model. Inference is performed parallely on 8 A100 GPUs with 40 GB GPU memory which takes around 1.5-2 hours.

[7] https://github.com/dki-lab/Freebase-Setup

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 8

☑ A2. Did you discuss any potential risks of your work?
Section 9

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3

☑ B1. Did you cite the creators of artifacts you used?
Section 3 and Appendix A.1

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix A.1

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix A.1

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
We have provided relevant statistics about the data in Table 2,6 and 7.

C ☑ Did you run computational experiments?

Appendix A.3

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A.3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix A.3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5 and Section 8

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix A.3

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.