Designing harder benchmarks for evaluating zero-shot generalizability in Question Answering over Knowledge Bases

Ritam Dutt[*]
Carnegie Mellon University
rdutt@andrew.cmu.edu

Sopan Khosla
AWS AI Labs
sopankh@amazon.com

Vinayshekhar Bannihatti Kumar
AWS AI Labs
vinayshk@amazon.com

Rashmi Gangadharaiah
AWS AI Labs
rgangad@amazon.com

Abstract

Most benchmarks for question answering on knowledge bases (KBQA) operate with the i.i.d. assumption. Recently, the GrailQA dataset was established to evaluate zero-shot generalization capabilities of KBQA models. Reasonable performance of current KBQA systems on the zero-shot GrailQA split hints that the field might be moving towards more generalizable systems. In this work, we observe a bias in the GrailQA dataset towards simpler one or two-hop questions which results in an inaccurate assessment of the aforementioned prowess. We propose GrailQA++, a challenging zero-shot KBQA test set that contains a larger number of questions relying on complex reasoning. We leverage the concept of reasoning paths to control the complexity of the questions and to ensure that our proposed test set has a fair distribution of simple and complex questions. Evaluating existing KBQA models on this new test set shows that they suffer a substantial drop in performance as compared to the GrailQA zero-shot split. This highlights the *non-generalizability* of existing models and the necessity for harder benchmarks. Our analysis reveals how reasoning paths can be used to understand complementary strengths of different KBQA models, and provide a deeper insight into model mispredictions.

1 Introduction

The task of KBQA involves querying a knowledge base (KB) for a set of entities that satisfies a natural language question. Despite being a well-studied research area, most of the prior work in KBQA has been restricted to an i.i.d. setting (Yih et al., 2016; Talmor and Berant, 2018). However, the ubiquitous applications of KBQA to different domains such as tax, insurance, and healthcare (Lüdemann et al., 2020; Huang et al., 2021; Park et al., 2020) has prompted research on KBQA generalizability (Dutt et al., 2022; Das et al., 2021; Neelam et al., 2022).

The most salient work is that of Gu et al. (2021) where they propose the task of KBQA generalizability beyond the i.i.d setting, specifically zero-shot generalizability. In a zero-shot setting, KBQA models operate upon classes and relations which were unobserved during training. They also create a dataset called GrailQA to benchmark the generalizability of KBQA models. This dataset has garnered significant research interest with state-of-the-art KBQA models (Ye et al., 2021; Yu et al., 2022; Gu and Su, 2022; Shu et al., 2022; Liu et al., 2022) achieving remarkable performance on the leaderboard, specifically on the zero-shot setting. [1]

However, a closer inspection of the GrailQA dataset reveals that it is biased towards simpler questions and that existing KBQA systems cannot deal with complex cases in a non-i.i.d. setting. We use reasoning paths to characterize the complexity of the questions (Li and Ji, 2022; Das et al., 2022). We observe a pronounced skewness in the distribution of reasoning paths (henceforth RPs) in the GrailQA dataset. The simplest RP, where the answer is located one hop away from starting entity, comprises 78.5% of the GrailQA zero-shot samples, while a more complicated RP with answers located three hops away accounts for only 0.53%. Consequently, RPs provide a more nuanced view into the generalization abilities of different KBQA models on questions of varying complexity.

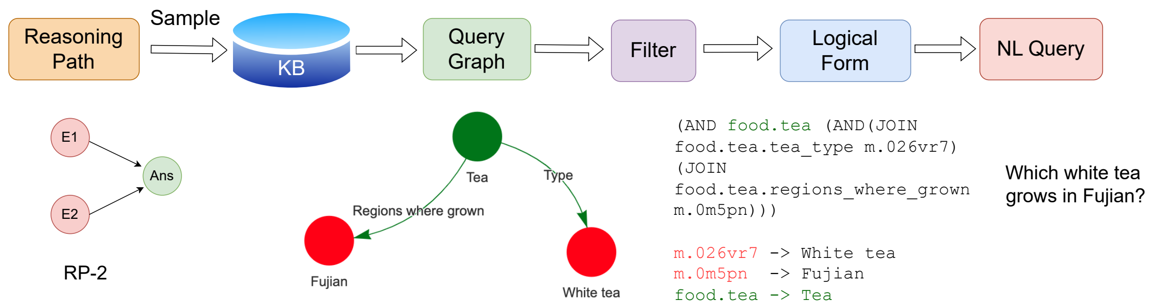
We thus propose a new zero-shot benchmark called GrailQA++ that has a balanced distribution of RPs. We evaluate four state-of-the-art (SOTA) KBQA models on this benchmark and observe that the performance falls significantly for all systems. Our analysis shows that this drop can be attributed to their inability to adapt to complex RPs.

Our contributions are the following:

- We leverage the concept of reasoning paths (RPs) to analyze the complexity of KBQA questions.
- We create a new benchmark with complex KBQA

[*]Work conducted during an internship at Amazon.

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



```
(AND food.tea (AND(JOIN
food.tea.tea_type m.026vr7)
(JOIN
food.tea.regions_where_grown
m.0m5pn)))

m.026vr7 -> White tea
m.0m5pn  -> Fujian
food.tea -> Tea
```

Figure 1: Schematic diagram that outlines the GrailQA++ dataset creation. We sample query graphs from Freebase KB based on the type of the reasoning paths. After filtering, they are converted into their logical form. We then provide them to human annotators who create the corresponding NL query (more details in Section 3).

questions to evaluate zero-shot generalizability.

- Our experiments show that SOTA models perform poorly on the new dataset, emphasizing that KBQA generalizability is still a challenge.[2]

2 Reasoning Paths in GrailQA

In a semantic-parsing based KBQA setting, a natural language question is first converted to a logical form and then executed over the KB to yield an answer. To ensure generalization, such KBQA models need to handle different kinds of logical forms. In this section we propose a way to categorize these logical forms using reasoning paths.

Reasoning Paths: A logical form has an equivalent query graph representation: the nodes and edges in the query graph denote the classes in the KB and relationship between those classes respectively. Furthermore, we denote the node corresponding to the answer class in the query graph as the root, and the nodes corresponding to entities and literals as constraints (green and red nodes respectively Figure 1).

A reasoning path (RP) simply outlines the traversal path from the constraints in the query graph to the root (or the answer). It obfuscates any specific information such as the name of the entities or classes. They provide a unified way to characterize a query graph (and subsequently a logical form) based on the number of constraints, and the number of hops required to reach the answer from said constraints. For example, in Figure 1, the green Tea node corresponds to Ans while the red constraint nodes, Fujian and White tea, corresponds to E1 and E2 respectively. Thus the given logical form is an instance of RP-2. We also present instances of different reasoning paths with their corresponding examples in Table 1.

[2]We will make the dataset public after acceptance.

Statistics for GrailQA: We categorize the questions in GrailQA according to the RP type. We refer to RPs with fewer than 3 relations as simple (RP-0, 1, and 2) and the rest as complex RPs (RP-3, 4, 5, and 6). We show the distribution of the RPs in the zero-shot development data of GrailQA in Table 3. We also present examples of each of these RPs in the Appendix.

We observe that simple reasoning paths (RP-0, 1, 2) comprise more than 98% of all zero-shot examples in the development set. A similar story holds true for the train set. We hypothesize that this skewness could exaggerate the perceived generalization capabilities of KBQA models, such that the staggering numbers on the leaderboard reflect the performance on these simpler reasoning paths.

3 GrailQA++

To gauge whether KBQA models exhibit zero-shot generalization capabilities across different RPs, we propose GrailQA++, a dataset with a more balanced distribution of seven RPs. We outline the creation process below and illustrate in Figure 1.

Query Graph Sampling: GrailQA was created using the OVERNIGHT process (Su et al., 2016) which extracts templates by traversing Freebase and obtains a query graph. Since traversal is easier for simpler hops and subsequently simpler RPs, they appear higher in GrailQA. We, however, follow a more controlled algorithm to generate the query graph. Firstly, we choose a particular RP, which determines the number of constraints. If there is exactly one constraint (RP-0, 1, and 3), we first choose a class at random in the KB and then sample an entity randomly from that class. Then,

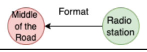
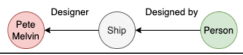
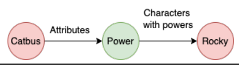
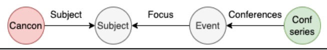
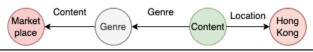
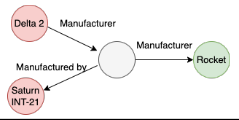
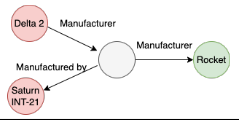
RP-Code	RP Example	Question
RP-0		“which radio station does have middle of the road?”
RP-1		“what ship designer designed a ship that is designed by pete melvin?”
RP-2		“which powers do both catbus and rocky the flying squirrel have?”
RP-3		“can-con has which conference series that focuses on it?”
RP-4		“genres of marketplace can be found in what broadcast content in hong kong?”
RP-5	 Delta 2 Manufacturer Manufacturer Rocket Manufactured by Saturn INT-21	“what other rocket did the manufacturer of saturn int-21 and delta 2 create?”
RP-6	 Delta 2 Manufacturer Manufacturer Rocket Manufactured by Saturn INT-21	“what other rocket did the manufacturer of saturn int-21 and delta 2 create?”

Table 1: We present a representative natural-language question and the RP class the questions corresponds to. Red and green nodes in the graph correspond to constraints (entities and literals), and the answer respectively. These examples are obtained directly from the development split of the GrailQA dataset.

we traverse the KB based on the relations that originate from the instantiated entity and continue till we reach the answer node. In case of multiple constraints (RP-2, 4, 5, and 6), we first randomly sample the answer class and then traverse the KB by adding relations in a manner that conforms with the RP structure. At each expansion step, we ensure that there exists an entity which can be instantiated using the new relation. This ensures executability of the current sub-query and thus of the main query.

Filtering: We filter query graphs that do not conform with the zero-shot generalizability criteria. Specifically, the query graph should have at least one class or relation absent from the GrailQA training split. Later, we employ the filtering techniques proposed in Gu et al. (2021) to discard illegal relations, and ignore instances with entities or relations written in a language other than English.

Logical Form: Once we obtain the filtered query graph, we convert it to its canonical logical form using the deterministic algorithm of Gu et al. (2021). We then execute this logical form over Freebase to obtain the answers, and discard instances where the logical form was inexecutable or unanswerable.

NL Query Annotation: To create the corresponding natural language question we choose annotators who have prior domain expertise in KBQA. The annotators are first provided with a design document with examples of query graphs and their corresponding logical form. We also provide the annotators with aliases of the constraints and relations to better interpret the query graph as they curate the natural language question.[3] We randomly select 35 instances (5 from each RP) to include in the pilot study after which the annotators meet to discuss their interpretations and resolve any differences. We find that all three annotators agree on 75% of the examples, while at least two agree on 97%. The main causes of disagreement were the direction of relations, and deciding how explicitly the entities should be referred in the NL query. Finally, we sample a larger set with 1000 unique query-graphs equally distributed among the three annotators. We ensure a balanced distribution between the different kinds of RPs (see Table 3).

4 Experimental Setup

Baselines: We experiment with four semantic-parsing baselines for KBQA. These include (i) RNG-KBQA (Ye et al., 2021), (ii) ArcaneQA (Gu and Su, 2022), (iii) BERT-Ranker (Gu et al., 2021), and (iv) BERT-Transducer (Gu et al., 2021).

Both RNG-KBQA and BERT-Ranker follow a ranking-based approach wherein they first enumerate all possible candidates and then perform semantic matching to find the most relevant candidate. RNG-KBQA in addition to the ranking phase, uses a pre-trained LM to generate an executable

[3]Example screenshots provided in Appendix B.

	GrailQA				GrailQA++	
Model	EM	F1	EM(Z)	F1(Z)	EM	F1
RNG-KBQA	90.7	91.9	90.1	90.9	53.6	68.8
ArcaneQA	83.3	86.4	79.0	82.0	28.7	50.3
BERT-Ranker	76.9	81.3	77.4	80.7	22.2	56.6
BERT-Transducer	51.9	54.6	43.8	46.0	10.3	18.8

Table 2: EM and F1 scores for different baselines for the GrailQA and GrailQA++ datasets (with gold entities). Z refers to the zero-shot subset of GrailQA dev set.

query from the top-ranked candidates. Whereas, ArcaneQA and BERT-Transducer employ an end-to-end generation model to obtain the final logical form. While ArcaneQA leverages a constrained decoding paradigm using facts in the KB, BERT-Transducer relies on unconstrained decoding.

These baselines encapsulate different strategies of carrying out semantic parsing in the context of KBQA (Gu et al., 2022). They also achieve impressive performance on the GrailQA leaderboard and also have publicly available checkpoints which can be used for evaluation. [4] We follow the exact inference setting mentioned in their Github repositories.

Evaluation Criteria: We evaluate the performance of these 4 baselines in terms of EM and F1 scores. To ensure a fair comparison between the two datasets, we (i) exclude questions with operations (like comparative or aggregation), literals or temporal constraints; and (ii) use gold entities to discount the entity linking errors. These serve as additional control for measuring the generalizability of KBQA models across different RPs.

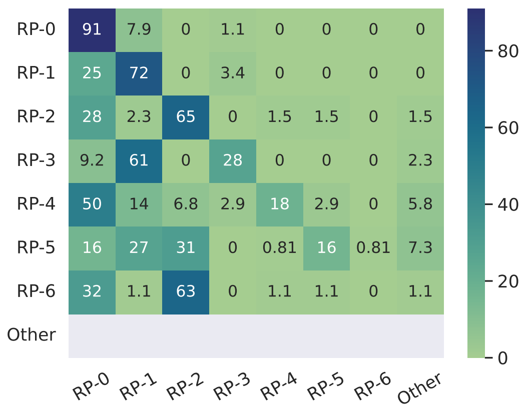
All experiments are carried out on a RTX-1080Ti GPU with 12GB RAM, using the author-provided model-checkpoints on the public GrailQA dev set.

5 Results and Analysis

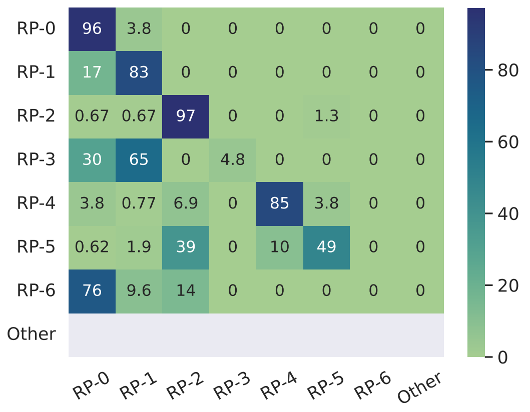
We present the zero-shot performance of all models on GrailQA and GrailQA++ in Table 2. We observe that models show impressive performance on GrailQA with RNG-KBQA achieving the highest F1 score of 90.9 on the zero-shot split and 91.9 overall. We also note that these models suffer a drop of at least 10 points in Gu and Su (2022) in absence of gold entities, emphasizing the importance of NER and entity-linking (EL) for KBQA.

Nevertheless, even while controlling for gold EL, the performance drops sharply on GrailQA++: the most noticeable drop occurs for BERT-Ranker and ArcaneQA. We attribute this to the skewed RP

[4]At the time of writing this paper.



(a) ArcaneQA on GrailQA++



(b) RNG-KBQA on GrailQA++

Figure 2: Confusion matrices for gold reasoning paths vs predicted reasoning paths on the GrailQA++ dataset.

distribution and report the results for each category in Table 3 for both datasets.

BERT-Ranker enumerates candidates for only RP-0 and RP-1 and thus scores an EM of 0.0 on the remaining RPs on GrailQA and GrailQA++. Likewise for RNG-KBQA, its enumeration strategy includes all reasoning paths except RP-3 and RP-6 resulting in low (or zero) EM scores for these forms. This suggests that for ranking-based approaches, prior knowledge of all possible reasoning paths aids generalization. The confusion matrix in Figure 2 shows a strong correlation: models perform better on reasoning paths more prevalent during training. E.g. RNG-KBQA outputs primarily logical forms corresponding to RP-0 for RP-6.

ArcaneQA has the best performance of all models on RP-3 questions on GrailQA++ and we observe that it is biased towards generating logical forms with longer hops (RP-1 and 3). This also demonstrates the complementary strengths of these models where RNG-KBQA fares better in pres-

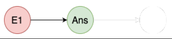
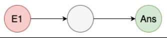
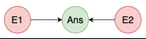
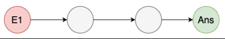
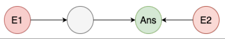
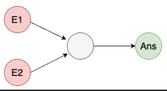
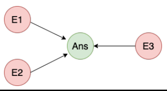
RP-Codes		GrailQA (ZS)				GrailQA++			
		Dist	RNG	Arc	B-Rank	Dist	RNG	Arc	B-Rank
RP-0		78.5	93.1 / 93.3	83.4 / 85.9	82.4 / 83.4	10.4	90.4 / 93.3	70.2 / 76.9	85.6 / 85.6
RP-1		16.6	81.9 / 85.3	68.0 / 71.8	76.4 / 80.0	17.2	75.6 / 79.5	51.2 / 57.5	76.7 / 80.1
RP-2		3.2	98.1 / 98.1	67.9 / 79.6	0.0 / 50.9	14.9	97.3 / 98.0	53.7 / 72.6	0.0 / 73.7
RP-3		0.5	0.0 / 2.0	0.0 / 0.0	0.0 / 7.8	14.6	4.8 / 19.8	15.1 / 27.2	0.0 / 14.5
RP-4		0.2	0.0 / 29.4	0.0 / 25.9	0.0 / 4.6	13.1	73.3 / 86.7	11.5 / 46.3	0.0 / 39.5
RP-5		0.9	22.2 / 22.2	0.0 / 3.7	0.0 / 28.2	16.0	38.8 / 50.6	5.0 / 25.8	0.0 / 51.3
RP-6		0.0	-	-	-	13.5	0.0 / 61.1	0.0 / 53.6	0.0 / 53.6

Table 3: EM / F1 scores for RNG-KBQA (RNG), ArcaneQA (Arc), and BERT-Ranker (B-Rank) across the different reasoning paths (RPs) in GrailQA (zero-shot subset; 3154 samples) and GrailQA++ (1000 samples). We also show the distribution of those RPs in the Dist column.

ence of multiple constraints (RP-4 and 5) whereas ArcaneQA is better on multiple hops (RP-3).

6 Conclusion

We propose a new dataset, GrailQA++, to benchmark the zero-shot generalization capabilities of KBQA models to complex questions. We characterize the question complexity with reasoning paths that characterizes both the dimensions of hops and constraints. Our experiments reveal poor generalization performance of SOTA KBQA models on our proposed dataset even when gold entities are provided during inference. Our analysis also reveals complementary strengths of different KBQA models on different types of reasoning paths. We also demonstrate how reasoning paths can be used to categorize and group model mispredictions.

7 Limitations

Since the major contribution of the work involved creation of a new dataset to test the zero-shot generalization capabilities of KBQA, the limitations of the work could be stated in that aspect. Specifically, to ensure we focus only on reasoning paths as a measure of complexity, we do not include any literals nor any comparative or aggregation functions. We also decouple the act of entity linking from KBQA and hence provide gold entities. Furthermore, to aid machine understanding we avoid paraphrasing and try to construct natural language queries with explicit mention of classes and relations of interest. We intend to address these limitations in the future, but for the time being we wanted to control for complexity using reasoning paths which motivated the following design choice.

8 Ethics Statement

The task of KBQA involves querying a knowledge graph to return an answer. Like most NLP research, the work leverages the prowess of large pre-trained language models like BERT, and T-5 and thus the harms associated with these models are to be noted during deployment. Furthemore, since KBQA involves interaction with an user to answer queries, these models should undergo rigorous model-testing before deployment.

References

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*, volume 162 of *Proceedings of Machine Learning Research*, pages 4777–4793. PMLR.

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.

Ritam Dutt, Kasturi Bhattacharjee, Rashmi Gangadharaiah, Dan Roth, and Carolyn Rose. 2022. Perkgqa: Question answering over personalized knowledge graphs. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 253–268.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. Knowledge base question answering: A semantic parsing perspective. *arXiv preprint arXiv:2209.04994*.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. *arXiv preprint arXiv:2204.08109*.

Xiaofeng Huang, Jixin Zhang, Zisang Xu, Lu Ou, and Jianbin Tong. 2021. A knowledge graph based question answering method for medical domain. *PeerJ Computer Science*, 7:e667.

Mingchen Li and Jonathan Shihao Ji. 2022. Semantic structure based query graph prediction for question answering over knowledge graph. *arXiv preprint arXiv:2204.10194*.

Ye Liu, Semih Yavuz, Rui Meng, Dragomir Radev, Caiming Xiong, and Yingbo Zhou. 2022. Uniparser: Unified semantic parser for question answering on knowledge base and database. *arXiv preprint arXiv:2211.05165*.

Niklas Lüdemann, Ageda Shiba, Nikolaos Thymianis, Nicolas Heist, Christopher Ludwig, and Heiko Paulheim. 2020. A knowledge graph for assessing agressive tax planning strategies. In *International Semantic Web Conference*, pages 395–410. Springer.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, et al. 2022. A benchmark for generalizable and interpretable temporal question answering over knowledge bases. *arXiv preprint arXiv:2201.05793*.

Junwoo Park, Youngwoo Cho, Haneol Lee, Jaegul Choo, and Edward Choi. 2020. Knowledge graph-based question answering with electronic health records. *arXiv preprint arXiv:2010.09394*.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. Tiara: Multi-grained retrieval for robust question answering over large knowledge bases. *arXiv preprint arXiv:2210.12925*.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572.

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)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv preprint arXiv:2109.08678*.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. 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.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *arXiv preprint arXiv:2210.00063*.

Supplementary Material

A Preliminaries

In this section, we describe the task setting and the different levels of generalization in the context of KBQA. A more detailed description can be found in (Gu et al., 2022).

A.1 Task Formulation

Knowledge Base: We denote Knowledge Base or KB as $\mathcal{K} = (\mathcal{O}, \mathcal{M})$, where $\mathcal{O}$ defines the ontology of the KB and $\mathcal{M}$ specifies the set of relational facts present in $\mathcal{K}$ on the basis of $\mathcal{O}$. The ontology is a subset of all possible relations $\mathcal{R}$ that can exists between two classes $\mathcal{C}$ i.e., $\mathcal{O} \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$. Likewise, the set of facts is represented as $\mathcal{M} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{L} \cup \mathcal{E} \cup \mathcal{C})$, where $\mathcal{E}$ and $\mathcal{L}$ denote the set of possible entities and literals respectively.

Semantic-parsing based KBQA: Given the KB, $\mathcal{K}$, and a natural language question q, the objective of KBQA is to find a set of entities (answers $\mathcal{A}$) that satisfies the question q. In a semantic-parsing or translation based setting, the task of KBQA involves converting q into its corresponding logical form L_q. This L_q is executed over the $\mathcal{K}$ to obtain the answers. Examples of logical forms include S-expressions, SPARQL queries, and λ-calculus.

Each logical form L_q has a particular schema $\mathcal{S}_q$ that includes elements from the set of relations, classes, and other constructs specific to the logical-form. The specific composition of items in $\mathcal{S}_q$ forms a logical template or $\mathcal{T}_q$. E.g., the questions "Who wrote Pride and Prejudice?" and "Who was the author of Oliver Twist?" have the same template $\mathcal{T}_q$ but different logical forms L_q since they refer to different novels. However the questions "Who wrote Pride and Prejudice?" and "Which author wrote both the Talisman and It?" have the same schema $\mathcal{S}_q$ but different logical templates $\mathcal{T}_q$.

A.2 KBQA Generalization

Gu et al. (2021) puts forward the three levels of generalization based on how the schema $\mathcal{S}_q$ and logical template $\mathcal{T}_q$ for a question q differs from the set of all possible schema items and templates seen during training, i.e. $\mathcal{S}_{train}$ and $\mathcal{T}_{train}$ respectively.

(i) **I.I.D.** generalization occurs when $\mathcal{S}_q \subset \mathcal{S}_{train}$ and $\mathcal{T}_q \in \mathcal{T}_{train}$.

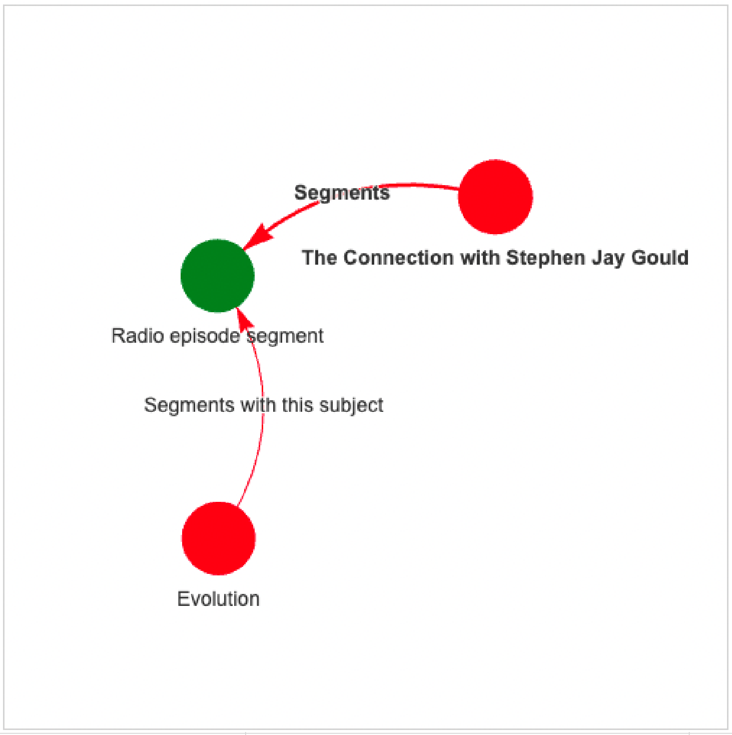
(ii) **Compositional** generalization occurs when $\mathcal{S}_q \subset \mathcal{S}_{train}$ but $\mathcal{T}_q \notin \mathcal{T}_{train}$. Thus the questions operate upon a subset of schema items seen during training but they have new templates.

(iii) **Zero Shot** generalization occurs when $\exists s \in \mathcal{S}_q$ such that $s \notin \mathcal{S}_{train}$. Thus the questions operate upon novel schemas, mostly new classes and relations that were not encountered during training.

Conceptually, these three levels of generalization could be stacked in an hierarchical fashion in increasing order of difficulty; with I.I.D. being the least challenging since it operates over templates seen during training, followed by Compositional, which occurs over unseen templates, and then Zero Shot which have unseen schema items.

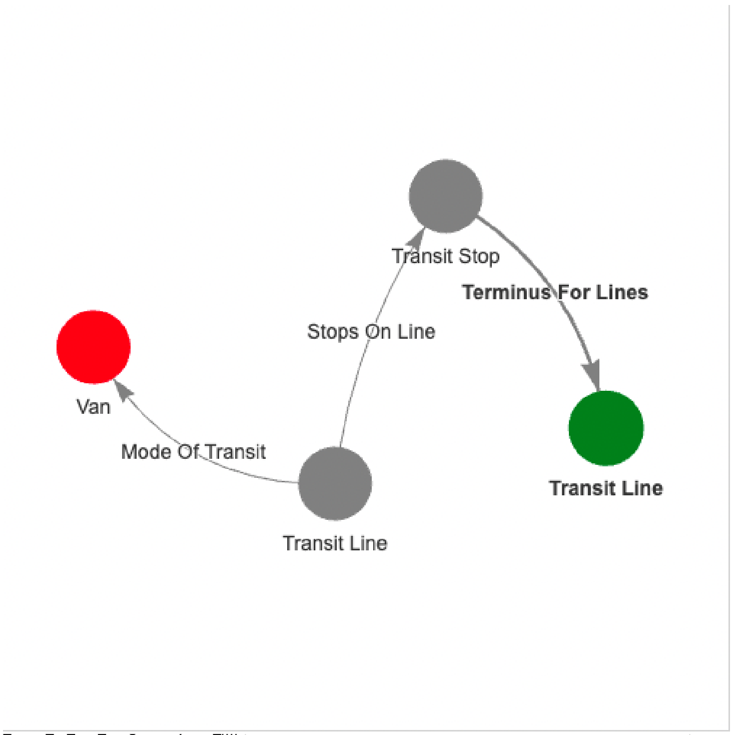
B Annotation Screenshots

We present examples of annotation screenshots for different reasoning paths that we considered in the GrailQA++ dataset. Each screenshot includes a pictoral representation of a query graph along with additional information, namely the S-expression or logical form corresponding to the query graph, the constraints in the form of entities and literals, and the answers. Each annotator was presented with each of these information before constructing the natural language query for it. Additionally, we also show each annotator examples of RPs in Table 1.


Segments
The Connection with Stephen Jay Gould
Radio episode segment
Segments with this subject
Evolution

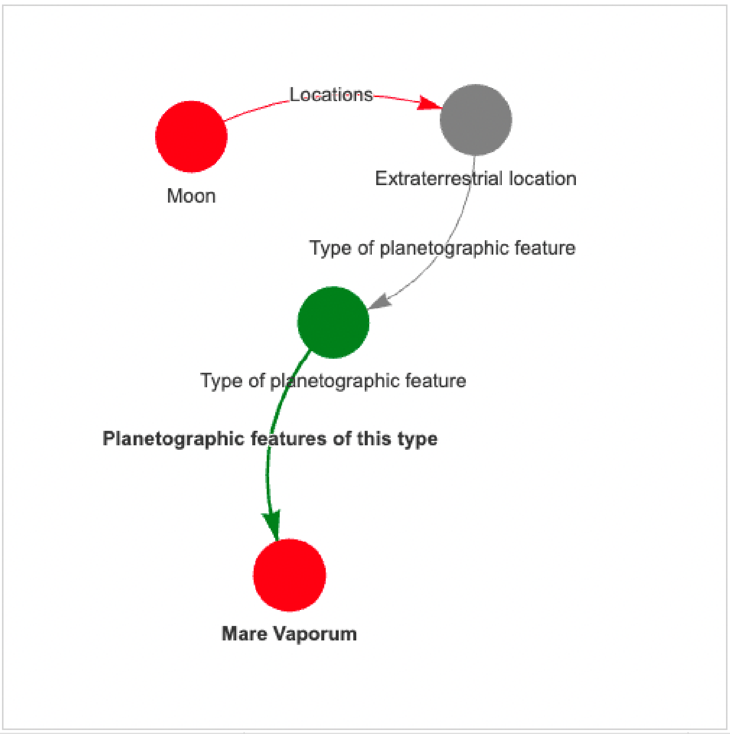
(AND radio.radio_episode_segment (AND (JOIN (R radio.radio_subject.segments_with_this_subject) m.02j8z) (JOIN (R radio.radio_program_episode.segments) m.0blhhfg)))	m.02j8z --> Evolution m.0blhhfg --> The Connection with Stephen Jay Gould radio.radio_episode_segment --> Radio episode segment	m.0blhk6j --> Christopher Lydon with Stephen Jay Gould

Figure 3: Example of a RP-2 graph and csv row (S-expression; Common names for entities and answer class; Answer node and name) shown to the annotator.



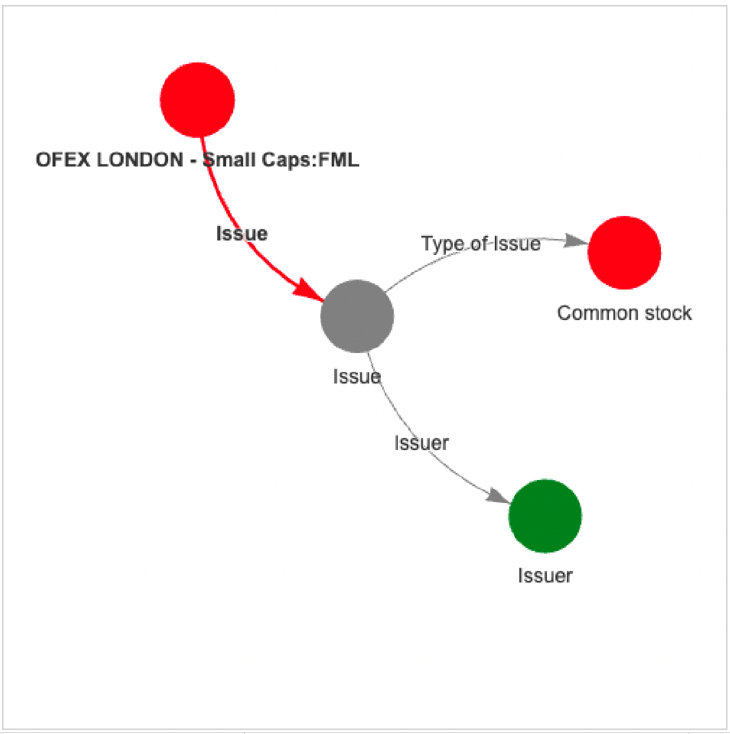
(AND metropolitan_transit.transit_line (JOIN (R metropolitan_transit.transit_stop.terminus_for_lines) (JOIN (R metropolitan_transit.transit_line.stops) (JOIN metropolitan_transit.transit_line.service_type m.0452jfk))))	m.0452jfk --> Van metropolitan_transit.transit_line --> Transit Line metropolitan_transit.transit_stop --> Transit Stop metropolitan_transit.transit_line --> Transit Line	m.0452j59 --> Quartzsite Transit Service m.0403pn4 --> Walnut Line

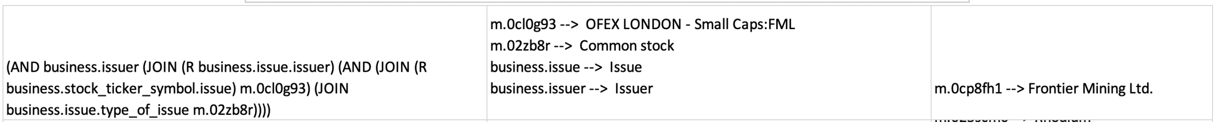
Figure 4: Example of a RP-3 graph and csv row (S-expression; Common names for entities and answer class; Answer node and name) shown to the annotator.


Locations
Moon
Extraterrestrial location
Type of planetographic feature
Type of planetographic feature
Planetographic features of this type
Mare Vaporum

(AND astronomy.type_of_planetographic_feature (AND (JOIN (R astronomy.extraterrestrial_location.type_of_planetographic_feature) (JOIN (R astronomy.celestial_object.locations) m.04wv_)) (JOIN astronomy.type_of_planetographic_feature.planetographic_features_of_this_type m.01d80r)))	astronomy.type_of_planetographic_feature --> Type of planetographic feature astronomy.extraterrestrial_location --> Extraterrestrial location m.04wv_ --> Moon m.01d80r --> Mare Vaporum	m.03dy13 --> Lunar mare

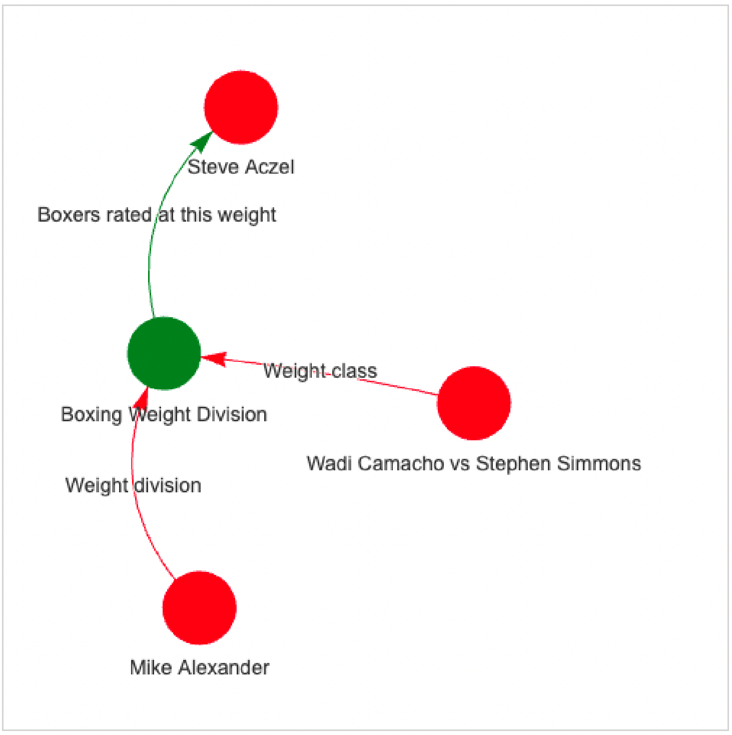
Figure 5: Example of a RP-4 graph and csv row (S-expression; Common names for entities and answer class; Answer node and name) shown to the annotator.





(AND business.issuer (JOIN (R business.issue.issuer) (AND (JOIN (R business.stock_ticker_symbol.issue) m.0cl0g93) (JOIN business.issue.type_of_issue m.02zb8r))))	m.0cl0g93 --> OFEX LONDON - Small Caps:FML m.02zb8r --> Common stock business.issue --> Issue business.issuer --> Issuer	m.0cp8fh1 --> Frontier Mining Ltd.

Figure 6: Example of a RP-5 graph and csv row (S-expression; Common names for entities and answer class; Answer node and name) shown to the annotator.



(AND sports.boxing_weight_division (AND (JOIN (R sports.boxer.weight_division) m.04d_1yl) (AND (JOIN (R boxing.boxing_match.weight_class) m.0110yljq) (JOIN sports.boxing_weight_division.boxers_rated_at_this_weight m.0ynrkh_))))	m.04d_1yl --> Mike Alexander m.0110yljq --> Wadi Camacho vs Stephen Simmons m.0ynrkh_ --> Steve Aczel sports.boxing_weight_division --> Boxing Weight Division	m.05_zmx --> Cruiserweight

Figure 7: Example of a RP-6 graph and csv row (S-expression; Common names for entities and answer class; Answer node and name) shown to the annotator.