

# GrailQA++: A Challenging Zero-Shot Benchmark for Knowledge Base Question Answering

**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 designed for question answering over knowledge bases (KBQA) operate with the i.i.d. assumption where one encounters the same schema items during inference as those observed during training. Recently, the GrailQA dataset was established to evaluate zero-shot generalization capabilities of KBQA models as a departure from the i.i.d. assumption. 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 more questions relying on complex reasoning. We leverage the concept of graph isomorphisms to control the complexity of the questions and to ensure that our proposed test set has a fair distribution of simple and complex questions. Existing KBQA models suffer a substantial drop in performance on our constructed new test set as compared to the GrailQA zero-shot split. Our analysis reveals how isomorphisms can be used to understand the complementary strengths of different KBQA models and provide a deeper insight into model mispredictions. Overall, our paper highlights the *non-generalizability* of existing models and the necessity for designing more challenging benchmarks. Our dataset is available at <https://github.com/sopankhosla/GrailQA-PlusPlus>

## 1 Introduction

The task of KBQA involves querying a knowledge base (KB) for a set of entities that satisfies a natural language question. Most prior work in KBQA has been restricted to an i.i.d. setting (Yih et al., 2016; Talmor and Berant, 2018), where the classes and

relations constituting the KB remains unchanged during inference and training. However, the ubiquitous applications of KBQA in 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 generalization to facilitate transfer to these domains (Dutt et al., 2022; Das et al., 2021; Neelam et al., 2022; Jiang and Usbeck, 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, which they term “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.<sup>1</sup>

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 put forward the notion of graph isomorphisms to characterize the complexity of the questions, which is similar in spirit to the idea of reasoning paths or semantic structures of (Li and Ji, 2022; Das et al., 2022). We observe a pronounced skewness in the distribution of isomorphisms in the GrailQA dataset. The simplest isomorphism, where the answer is located one hop away from starting entity, comprises 78.5% of the GrailQA zero-shot samples, while a more complex isomorphism with answers three hops away accounts for only 0.53%. In this work, we leverage the concept of isomorphisms to explore the generalization abilities of KBQA models on questions of varying complexity.

\*Work conducted during an internship at Amazon.

<sup>1</sup><https://dki-lab.github.io/GrailQA/>

We propose a new zero-shot benchmark called GrailQA++ that has a balanced distribution of simple and complex isomorphisms. The dataset comprises of questions annotated by domain experts as well as questions from well-known pre-existing KBQA datasets that are built over the same Freebase database as GrailQA. We evaluate two state-of-the-art (SOTA) KBQA models on this benchmark and observe that the performance falls significantly (28.5 on GrailQA++ as opposed to 83.5 on GrailQA). Our analysis shows that this drop can be attributed partly to the skewed distribution in GrailQA and that different models fare better on different isomorphism categories.

Our contributions are the following:

- We leverage the concept of graph isomorphisms to analyze the complexity of KBQA questions.
- We create a new benchmark (GrailQA++) with complex questions to evaluate zero-shot generalizability of KBQA models.
- Our experiments show that SOTA models perform poorly on the new dataset, emphasizing that KBQA generalizability is still a challenge.<sup>2</sup>
- We also carry out extensive error analysis to inspect model mispredictions and non-generalizability that would serve subsequent research in creating better benchmarks.

## 2 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).

### 2.1 Task Formulation

**Knowledge Base:** We denote a Knowledge Base or a 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 exist between two classes, which are denoted by  $\mathcal{C}$  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  $\lambda$ -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 but different logical forms 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 but different logical templates since the former involves only one constraint or entity (“*Pride and Prejudice*”) while the latter specifies two (“*Talisman*” and “*It*”),

### 2.2 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.

## 3 Isomorphisms 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.

<sup>2</sup>Our dataset is available here at <https://github.com/sopankhosla/GrailQA-PlusPlus>.




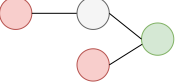
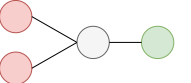

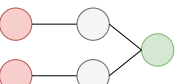
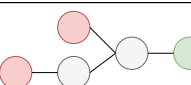
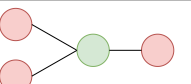
Iso-Code	Pictoral Desc.	GrailQA++											
		GrailQA		EAD		GraphQ		WebQSP		CWQ		Tot	
		Freq	Perc	Freq	Perc	Freq	Perc	Freq	Perc	Freq	Perc	Freq	Perc
Iso-0		2809	77.9	83	11.9	292	43.9	245	43.2	0	0.0	620	16.1
Iso-1		559	15.5	151	21.7	237	35.6	177	31.2	324	16.8	889	23.0
Iso-2		135	3.8	96	13.8	33	5.0	6	1.1	289	14.9	424	11.0
Iso-3		18	0.5	81	11.6	31	4.7	3	0.5	695	35.9	810	21.0
Iso-4		61	1.7	101	14.5	39	5.9	136	24.0	0	0.0	276	7.2
Iso-5		22	0.6	98	14.1	33	5.0	0	0.0	252	13.0	383	9.9
Iso-6		0	0.0	0	0.0	0	0.0	0	0.0	302	15.6	302	7.8
Iso-8		0	0.0	0	0.0	0	0.0	0	0.0	72	3.7	72	1.9
Iso-11		0	0.0	85	12.2	0	0.0	0	0.0	0	0.0	85	2.2

Table 1: Distribution of isomorphisms in the GrailQA (Dev) set and our curated GrailQA++ dataset (Tot). We show the total count of isomorphisms for each of the datasets (Freq) and their corresponding proportion in % (Perc). Note that complex isomorphisms belonging to Iso-6, Iso-8, and Iso-11 do not occur in the original GrailQA dataset. The red and green nodes in each isomorphism correspond to the constraints and the final answer respectively.

In this section we propose a way to categorize these logical forms using the notion of isomorphisms.

### 3.1 Isomorphisms

Each logical form  $L_q$  has an equivalent graphical notation  $\mathcal{G}_q$ , where the set of vertices  $V_q$  correspond to the different constraints ( $\mathcal{E}, \mathcal{L}$ ) and classes  $\mathcal{C}$  in the  $L_q$  while edges  $E_q$  represents the relations  $\mathcal{R}$  present in  $L_q$ . This notation is similar to the design of query-graphs (Lan and Jiang, 2020) but where the operations (aggregation or comparative) do not have any specialized vertices. We however denote one of the vertices in  $V_q$  that correspond to the answers as  $\mathcal{A}_q$  and call it the root. The nodes corresponding to the root and the constraints are denoted in green and red respectively in Figure 1.

We say two logical forms for questions  $q_i$  and  $q_j$  belong to the same isomorphism category, iff their equivalent graphs  $\mathcal{G}_{q_i}$  and  $\mathcal{G}_{q_j}$  are isomorphic. Subsequently, two graphs  $\mathcal{G}_{q_i}$  and  $\mathcal{G}_{q_j}$  are isomorphic iff there exists a mapping function  $\psi$  from  $V_{q_i}$  to  $V_{q_j}$  such that  $\forall m, n$  nodes in  $V_{q_i}$ , that correspond to an edge in  $\mathcal{G}_{q_i}$  i.e.  $(m, n) \in E_{q_i}$ , the mapping of

the nodes should also correspond to an edge in  $\mathcal{G}_{q_j}$  or,  $(\psi(m), \psi(n)) \in E_{q_j}$ . This mapping is bijective. Furthermore the roots in the two graphs also share the same mapping, i.e.  $\mathcal{A}_{q_j} = \psi(\mathcal{A}_{q_i})$ .

Isomorphisms describe how the constraints in the query graph are connected to the root (or the answer). It obfuscates any specific information such as the name of the entities or classes in the graph. 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 Iso-2. The distribution of isomorphisms spanning all datasets appears in Table 7.

While the notion of isomorphisms is similar in concept to the idea of reasoning paths (Das et al., 2022) or semantic structures (Li and Ji, 2022), we use the generic definition of “isomorphisms” to account for the fact that these graph isomorphisms

can also have cycles in them. For example, in Table 7, we note instances of isomorphisms (CIso-0 to CIso-4) where at least one cycle is present.

### 3.2 Statistics for GrailQA

We categorize the questions in GrailQA according to the isomorphism type of the corresponding logical form. We refer to isomorphisms with fewer than 3 relations as simple and the rest as complex isomorphisms. The simple isomorphisms for the remainder of the paper are Iso-0,1,2. We show the distribution of the isomorphisms in the zero-shot development data of GrailQA in Table 1.

We observe that the simple isomorphisms (Iso-0, 1, 2) comprise more than 97% of all zero-shot examples in the development set. A similar story holds true for the train set where 95% of all isomorphisms belong to these three classes (See Table 7 in the Appendix). 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 isomorphisms.

## 4 GrailQA++

To gauge whether KBQA models exhibit zero-shot generalization capabilities across different isomorphisms, we propose GrailQA++, a challenging dataset with an equal distribution of simple and complex isomorphisms. To create GrailQA++, we not only employ annotators with prior expertise in KBQA, but also leverage pre-existing KBQA datasets. We outline the creation process below and illustrate the same in Figure 1.

### 4.1 Expert Annotated Instances

We describe our controlled approach to sample and annotate instances of different isomorphism classes. Our process is similar to that of GrailQA albeit with a few differences, namely in terms of query sampling and natural language query generation.

**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 isomorphisms, they appear higher in GrailQA. We, however, follow a more controlled algorithm to sample the query graph.

We first choose a particular isomorphism, which determines the number of constraints. If there is

exactly one constraint (Iso-0, 1, and 5), we first choose a class at random and then sample an entity randomly from that class. We then follow the relations that originate from the instantiated entity and continue our traversal of the KB till we reach the answer node. In case of multiple constraints (Iso-2, 3, 4, and 11), we first randomly sample the answer class and then traverse the KB by adding relations in a manner that conforms with the isomorphism 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.

The authors chose to sample instances corresponding to Iso-0,1,2,3,4,5 since these were already present in the zero-shot split of GrailQA. Additionally, we also sampled and annotated instances of Iso-11, since it was the simplest isomorphism that could be formed with three constraints.

**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.

**Natural Language Query Annotation:** To create the corresponding natural language question we choose annotators who are fluent in English, are current working professionals with a graduate degree to their name, and 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 compose the corresponding question.<sup>3</sup> We randomly select 35 instances (5 from each isomorphism) 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

<sup>3</sup>Example screenshots provided in Appendix C.

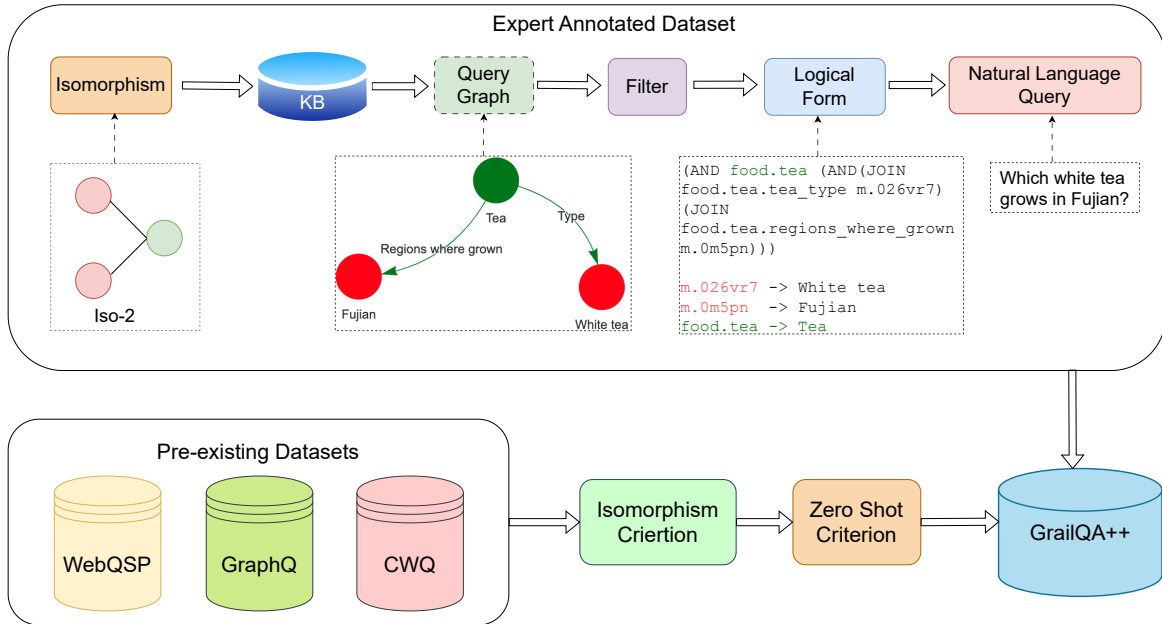


Figure 1: Schematic diagram that outlines the GrailQA++ dataset creation. The dataset comprises of question and corresponding logical forms, from two different sources. The former are instances which are hand-annotated by domain experts, and the latter are instances obtained from pre-existing datasets (WebQSP, CWQ, and GraphQ) which also operate over the same Freebase KB. (more details in Section 4).

at least two agree on 97%. The main causes of disagreement was determining how explicitly the entities should be referred in the NL query. The annotators decided to be explicit in specifying the hidden nodes to facilitate evaluation. Finally, we sample a large set with 1000 unique query-graphs equally distributed among the three annotators. We ensure a balanced distribution between the different kinds of isomorphisms (see Table 1). All annotations were carried out by domain experts and we did not employ any crowd-workers unlike in Gu et al. (2021) for paraphrasing.

## 4.2 Pre-existing Datasets

We also leveraged pre-existing public datasets that were built over the same Freebase KB as GrailQA. These datasets were chosen since they were designed to evaluate progress on KBQA.

**WebQSP** (Yih et al., 2016) uses Amazon Mechanical Turk to answer questions from non-experts collected using the Google Suggest API. Since the dataset is restricted to to "wh" questions from non-experts the questions tend to more colloquial.

**GraphQ** (Su et al., 2016) was created in a fashion similar to GrailQA with questions exhibiting variation in terms of complexity, topic space, and number of answers.

**ComplexWebQuestions (CWQ)** (Talmor and Berant, 2018) was created on top of WebQSP with the

intention of generating complex questions by incorporating compositions (more hops), conjunctions (more constraints), and superlatives and comparatives (more function types).

**Zero-shot splits:** We consider only the questions in the test splits of the pre-existing datasets which satisfy the zero-shot criteria of Gu et al. (2021). Specifically, zero-shot instances have at least one schema item (class or relation) that were not seen during training in the training data of GrailQA. Following Khosla et al. (2023), we also exclude questions if a relation’s corresponding inverse relation was observed during training to make the task more challenging. We follow the same criteria for the expert annotated dataset as well.

**Isomorphism criterion:** We sample instances corresponding to the following isomorphisms, Iso-0,1,2,3,4,5,6,8. The selection of these isomorphisms were driven by two criteria, namely (i) the isomorphisms should be present in the training split of the GrailQA dataset and (ii) there should be sufficient representation of these isomorphisms in the combined test-split of GrailQA++(>50).

## 4.3 Statistics of GrailQA++

We present the distribution of isomorphisms corresponding to our curated GrailQA++ in Table 1. We see that simple and complex isomorphisms

	RNG-KBQA		ArcaneQA	
Dataset	EM	F1	EM	F1
GrailQA (dev)	83.5	86.0	77.9	81.7
GrailQA++	28.5	38.6	18.6	32.5
- EAD	56.1	70.2	31.5	49.9
- GraphQ	53.2	61.7	30.2	44.8
- WebQSP	19.9	25.9	17.6	28.7
- CWQ	12.6	23.0	10.2	23.2

Table 2: EM and F1 scores for RNG-KBQA and the ArcaneQA model on the GrailQA and GrailQA++ datasets (with gold entities). EAD stands for the Expert Annotated Dataset that we had created.

are equally represented in the dataset, where the simple isomorphisms that correspond to Iso-0,1,2 comprise 50.1% of the dataset. We also include isomorphisms corresponding to Iso-6, Iso-8, and Iso-11 which are absent in the original dev split of GrailQA. This enables us to evaluate the zero-shot generalization performance of KBQA models on these unseen isomorphism categories.

## 5 Experimental Setup

**Baselines:** We experiment with two semantic-parsing baselines for KBQA namely RNG-KBQA (Ye et al., 2021) and ArcaneQA (Gu and Su, 2022). We chose these models because they encapsulate two different strategies of carrying out semantic parsing in the context of KBQA (Gu et al., 2022). Furthermore, they achieve impressive performance on the GrailQA leaderboard and also have publicly available checkpoints which can be used for evaluation. We follow the inference setting mentioned in their Github repositories, with the single exception that for RNG-KBQA we do not restrict ourselves to the subset of Freebase domains for GrailQA.

RNG-KBQA (Ye et al., 2021) follow a ranking-based approach wherein they first enumerate all possible candidates and then perform semantic matching to rank the enumerated candidates in decreasing order of relevance. They then use a pre-trained LM (T5-large) to generate an executable query from the top-ranked candidates.

ArcaneQA (Gu and Su, 2022) employ a seq2seq generative LM to obtain the final logical form from the natural language query. They leverage a constrained decoding paradigm that leverages the information in the KB during query generation to ensure executability.

**Evaluation Criteria:** We evaluate the performance of the two baselines in terms of EM (exact match) and F1 scores (between the predicted and gold answers). We decouple the impact of entity recognition and entity linking from the main task of KBQA by providing gold entities during inference. 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.

## 6 Results

In this section we put forward the following research questions and attempt to answer the same.

### RQ1. How well do the baselines generalize to our proposed GrailQA++ dataset?

We present the zero-shot performance of RNG-KBQA and ArcaneQA on GrailQA and GrailQA++ in Table 2. We observe that models show impressive performance on GrailQA with RNG-KBQA achieving a very high F1 score of 86.0 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 perfect EL, the performance drops sharply on GrailQA++, resulting in an F1 score of 38.6 and 32.5 for RNG-KBQA and ArcaneQA respectively. We attribute this to the skewed distribution of isomorphisms in the original GrailQA dev split, where the simpler isomorphisms (Iso-0,1,2) accounts for 97% of the dataset. RNG-KBQA achieves an F1 score of 86.5 and 30.1 on the simple and complex isomorphisms in GrailQA respectively (see Table 3).

We also investigate the models’ performance on questions with additional functions. These functions are (i) comparatives (ex. greater than, less than), (ii) superlatives (argmax, argmin), (iii) counting or aggregation, and (iv) none (absence of any specific operation). The results in Table 4 highlights that ArcaneQA scores higher on superlatives and comparative functions (in terms of F1 score) as opposed to RNG-KBQA for GrailQA++.

### RQ2. Do models exhibit similar performance on different isomorphism types?

We present a breakdown of the model performance according to the isomorphism type for GrailQA and GrailQA++ in Table 3.

The enumeration strategy of RNG-KBQA generates candidates corresponding to the first 5 isomorphisms (Iso-0,1,2,3 and 4). Consequently, we

Iso-Codes	GrailQA (Dev)		GrailQA++		EAD		GraphQ		WebQSP		CWQ	
	RNG	Arc	RNG	Arc	RNG	Arc	RNG	Arc	RNG	Arc	RNG	Arc
0	87.1/ 88.0	83.8/ 86.4	53.2/ 59.7	36.9/ 47.6	89.2/ 91.2	71.1/ 77.1	63.4/ 69.9	31.8/ 42.7	29.0/ 36.7	31.4/ 43.4	-	-
1	81.9/ 85.1	66.7/ 70.5	47.9/ 53.4	36.7/ 47.0	81.5/ 83.7	52.6/ 59.3	53.2/ 57.6	32.1/ 44.8	9.0/ 13.3	13.0/ 26.2	49.7/ 58.1	45.4/ 53.9
2	74.8/ 86.2	53.3/ 75.8	39.2/ 51.9	15.8/ 34.7	96.9/ 97.9	50.0/ 67.8	63.6/ 87.9	6.1/ 6.1	0.0/ 16.9	0.0/ 16.7	18.0/ 33.2	5.9/ 27.3
3	5.6/ 44.8	0.0/ 20.2	13.2/ 28.3	2.1/ 24.3	75.3/ 88.8	14.8/ 44.1	48.4/ 98.9	0.0/ 72.1	0.0/ 13.3	0.0/ 13.3	4.5/ 18.2	0.7/ 19.9
4	9.8/ 47.6	11.5/ 27.5	25.0/ 32.9	1.8/ 16.8	35.6/ 49.0	4.9/ 25.6	17.9/ 24.7	0.0/ 30.9	19.1/ 23.4	0.0/ 6.3	-	-
5	0.0/ 1.5	0.0/ 0.0	0.8/ 10.1	16.7/ 22.1	3.1/ 19.2	15.3/ 25.9	0.0/ 0.0	90.9/ 92.1	-	-	0.0/ 7.9	7.5/ 11.5
6	-	-	0.0/ 3.4	3.0/ 8.8	-	-	-	-	-	-	0.0/ 3.4	3.0/ 8.8
8	-	-	0.0/ 4.4	0.0/ 1.6	-	-	-	-	-	-	0.0/ 4.4	0.0/ 1.6
11	-	-	0.0/ 61.2	0.0/ 47.5	0.0/ 61.2	0.0/ 47.5	-	-	-	-	-	-

Table 3: EM / F1 scores for RNG-KBQA (RNG) and ArcaneQA (Arc), across the different Isomorphisms (Iso) in GrailQA (zero-shot subset) and GrailQA++. EAD stands for the expert annotated dataset that was created.

Dataset	RNG-KBQA				ArcaneQA			
	None	Count	Comparative	Superlative	None	Count	Comparative	Superlative
GrailQA (Dev)	90.1/ 90.9	91.1/ 95.3	38.6/ 73.8	0.0/ 7.8	80.2/ 83.2	68.1/ 71.7	41.2/ 65.5	72.1/ 76.5
GrailQA++	29.9/ 40.9	84.3/ 84.3	0.0/ 1.9	0.0/ 6.6	20.0/ 34.7	20.6/ 21.6	0.0/ 15.5	8.7/ 15.5

Table 4: EM/ F1 scores for RNG-KBQA and the ArcaneQA model on the GrailQA and GrailQA++ datasets with different functional forms. None means no special function was present.

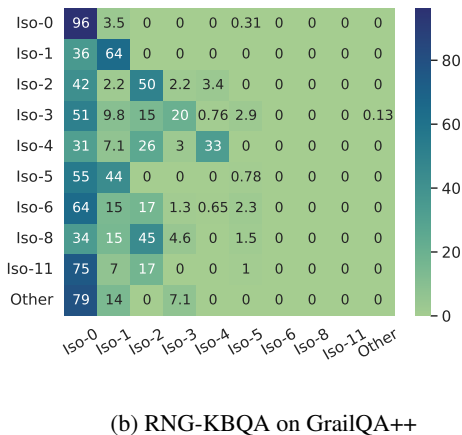
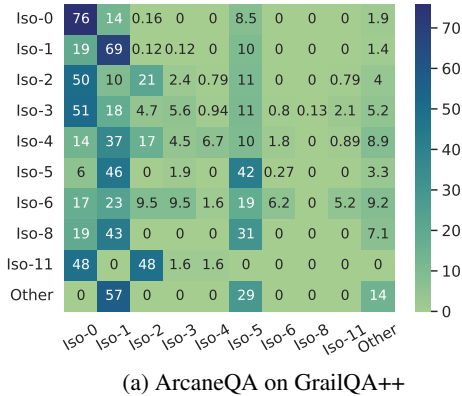


Figure 2: Confusion matrices for gold Isomorphisms vs predicted Isomorphisms on the GrailQA++ dataset for ArcaneQA (top) and RNG-KBQA (bottom).

obtain high scores for those specific isomorphisms and low (or zero) EM for the others. This suggests that a ranking-based approach, such as RNG-KBQA, requires prior knowledge of all possible isomorphisms to facilitate meaningful generalization. Nevertheless, RNG-KBQA achieves a comparatively higher F1 score for most isomorphisms in GrailQA++. ArcaneQA, on the other hand, has a higher score on Iso-5, and Iso-6 for GrailQA++.

We hypothesize that different KBQA models are biased towards generating/retrieving logical forms that conform to specific isomorphisms. To delve deeper, we categorize the models' mispredictions into different isomorphism types. We obtain confusion matrices for correct isomorphisms against the predicted isomorphism type for ArcaneQA and RNG-KBQA in Figure 2.

We observe that ArcaneQA is biased towards generating logical forms with longer hops (See the column corresponding to Iso-5 and Iso-1 in Figure 2a) which explains the higher EM of ArcaneQA on GrailQA++ for Iso-5. Furthermore, since RNG-KBQA outputs logical forms corresponding to the first 5 isomorphisms (Iso-0,1,2,3,4), the mispredictions are mostly confined to those specific forms.

Our experiments demonstrate the complementary strengths of these models such that RNG-KBQA fares better in presence of multiple con-

Dimension	GrailQA	EAD	GraphQ	WebQSP	CWQ	All
Complexity Score	-0.282***	+0.001	+0.00	+0.00	-0.124	-0.093*
Grammaticality	+0.013	+0.011	-0.063	+0.037	+0.027	-0.023
Readability	+0.000	+0.001	-0.001	-0.001	-0.001	-0.002***
Coherence	-0.069***	-0.075***	-0.085***	-0.031**	-0.024***	-0.068***
Sentence Length (#W)	+0.010***	-0.006	-0.015*	+0.028*	+0.006	+0.0021
Common Nouns (#N)	+0.037***	+0.000	+0.031	-0.022	+0.027***	+0.026***
Zero-shot Items (#Z)	-0.065***	+0.011	-0.005	-0.114***	-0.100***	-0.035***

Table 5: Coefficients of the different dimensions on the F1 score obtained through linear regression and their corresponding p-values. A positive coefficient indicates a positive correlation and vice versa. \*, \*\*, \*\*\* indicate that the coefficient is statistically significant with a p-value  $\leq 0.05$ , 0.01, and 0.001 respectively.

Dimension	GrailQA(Dev)	EAD	GraphQ	WebQSP	CWQ
Complexity Score	0.0 (0.1)	0.2 (0.4)	0.0 (0.0)	0.0 (0.0)	0.0 (0.1)
Grammaticality	0.7 (0.5)	0.6 (0.5)	0.8 (0.4)	0.7 (0.4)	0.8 (0.4)
Readability	60.5 (26.9)	58.4 (24.5)	71.8 (25.7)	77.0 (25.3)	69.9 (22.1)
Coherence	-9.8 (1.2)	-9.7 (1.2)	-9.4 (1.2)	-9.9 (1.3)	-9.3 (1.2)
Sentence Length (#W)	12.6 (3.7)	17.3 (5.2)	11.1 (3.0)	6.7 (1.6)	14.4 (3.3)
Common Nouns (#N)	4.7 (1.8)	6.6 (2.4)	3.4 (1.3)	2.2 (1.0)	5.3 (1.6)
Zero-shot items (#Z)	2.1 (0.9)	2.6 (1.3)	2.4 (1.2)	1.6 (0.7)	2.1 (0.9)

Table 6: We present the mean (std) on different linguistic dimensions on the zero-shot split of GrailQA development set (Dev), and GrailQA++.

straints (Iso-3,4) whereas ArcaneQA is better for multiple hops (Iso-5).

### RQ3. What linguistic characteristics of a dataset enable zero-shot generalization?

We observe from Table 2 that the constituent datasets of GrailQA++ exhibit wide variation in performance for both models. While complex isomorphisms usually have lower scores than the simpler ones, there are a few exceptions. For example, on the GraphQ split in Table 3, RNG-KBQA has a very high F1 score of 98.9 on Iso-3 as opposed to 69.9 for Iso-0. This motivates us to delve deeper and investigate whether certain dataset characteristics can explain this variation.

We inspect the following dataset characteristics namely the sentence length (#W), number of common nouns (#N), number of zero-shot items (#Z), readability, grammaticality, complexity, and coherence. The number of common nouns (#N) serves as a proxy for explicitness, i.e how thorough were the annotators in framing the question. The metrics corresponding to readability, complexity, and grammaticality helps to gauge the naturalness of a question, whereas coherence is used to quantify fluency. We describe the use of the metrics in the Appendix. We also outline the mean and standard deviation for each metric in Table 6.

We perform a multivariate regression analysis over the combined dataset or “All” with F1 score as the dependent variable and the aforementioned linguistic factors and number of zero-shot items as the independent variables to identify which dimensions are statistically significant. We carry out the same analysis for each individual dataset. We present the results in Table 5.

For the combined dataset, All, we observe that all factors except grammaticality and sentence length, are significant. We also note that complexity, readability, coherence, and the number of zero-shot items are negatively correlated with F1, while the number of common nouns (#N) is positively correlated.

While, there are fluctuations in trends, we note that for all the datasets, “coherence” is significantly and negatively correlated with performance. This observation aligns with prior findings of Linjordet and Balog (2022) where the fluency and naturalness of questions degrades KBQA performance. Moreover, the negative correlation with #Z implies that questions with a greater proportion of unseen classes and relations are harder for models to answer. Furthermore, a positive correlation with #N signifies that being more explicit in framing questions is beneficial for model performance. We see



similar trends in #N and #Z across most datasets.

One interesting observation is that our constructed dataset, EAD, is most similar to GraphQuestions both in terms of the EM/F1 scores (Table 2) as well as coefficients of different linguistic dimensions (Table 5). One possible hypothesis is that both these datasets were created in a similar fashion.

We observe that GrailQA mostly follows a similar trend to All since it accounts for 50% of the entire dataset. However, the readability metric for All is influenced by the pre-existing datasets (CWQ, GraphQ, and WebQSP) which have comparatively higher scores in Table 6. All in all, we note that KBQA systems struggle with fluent and natural questions (high coherence and readability scores).

## 7 Conclusion

We propose a new dataset, GrailQA++, to benchmark the zero-shot generalization capabilities of KBQA models on complex questions. We characterize the question complexity by introducing the concept of graph isomorphisms 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 isomorphisms and how isomorphisms can be used to categorize and group model mispredictions. We also carry out extensive analysis on our proposed dataset to identify linguistic factors, that are correlated with non-generalizability such as high coherence and readability. Our research sheds light on how to design harder benchmarks to evaluate zero-shot generalization of KBQA models.

## 8 Limitations

Since the major contribution of our work is the creation of a new dataset to test the zero-shot generalization capabilities of KBQA, the limitations of the work could be stated with regards to the dataset creation. An important thing to note is that unlike other benchmarks, our proposed GrailQA++ is designed solely for the purpose of evaluation. The dataset was created keeping in mind that the only training data one has access to is the train split of GrailQA. This is because the test splits in pre-existing datasets share a huge overlap with their

corresponding train splits. Consequently KBQA models trained on any additional dataset might violate the zero-shot criteria.

Additionally, we decouple the act of entity linking from question answering and thus the performance of models stated in this work are expected to be higher than using models that do not have access to 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 isomorphisms which motivated the following design choice.

## 9 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. Furthermore, 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 Gangadhariah, 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.
- Longquan Jiang and Ricardo Usbeck. 2022. Knowledge graph question answering datasets and their generalizability: Are they enough for future research? *arXiv preprint arXiv:2205.06573*.
- Pei Ke, Hao Zhou, Yankai Lin, Peng Li, Jie Zhou, Xiaoyan Zhu, and Minlie Huang. 2022. CtrlEval: An unsupervised reference-free metric for evaluating controlled text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2319.
- Sopan Khosla, Ritam Dutt, Vinayshekhar Bannihatti Kumar, and Rashmi Gangadharaiyah. 2023. Exploring the reasons for non-generalizability of kbqa systems. In *The Fourth Workshop on Insights from Negative Results in NLP*, pages 88–93.
- Yunshi Lan and Jing Jiang. 2020. [Query graph generation for answering multi-hop complex questions from knowledge bases](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.
- Mingchen Li and Jonathan Shihao Ji. 2022. Semantic structure based query graph prediction for question answering over knowledge graph. *arXiv preprint arXiv:2204.10194*.
- Trond Linjordet and Krisztian Balog. 2022. Would you ask it that way? measuring and improving question naturalness for knowledge graph question answering. *arXiv preprint arXiv:2205.12768*.
- Ye Liu, Semih Yavuz, Rui Meng, Dragomir Radev, Caiming Xiong, and Yingbo Zhou. 2022. Uni-parser: 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 aggressive tax planning strategies. In *International Semantic Web Conference*, pages 395–410. Springer.
- Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikkal, 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.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohanney, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.
- Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. [Neural network acceptability judgments](#). *Transactions of the Association for Computational Linguistics*, 7:625–641.
- 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 Isomorphism Distribution

We show the distribution of different isomorphisms in the training and test data of the different datasets in Table 7.

### B Analyzing Linguistic Variation

We adopt the following dimensions of [Khosla et al. \(2023\)](#) on our proposed dataset.

**Sentence Length (#W):** We simply count the number of words for each natural language questions across all datasets.

**Common Nouns (#N):** We use NLTK’s POS-tagger to identify common nouns that corresponding to “NN” and “NNS” tags.

**Grammaticality & Complexity:** We use the BLIMP ([Warstadt et al., 2020](#)) and COLA corpora ([Warstadt et al., 2019](#)) to fine-tune BERT-based text classification model to detect whether a given question is grammatical or not. We follow the same to determine whether a given question is complex or not, i.e. has several clauses.

**Readability:** We use the Flesch-reading score to characterize the readability of each question in the dataset, using the readability library in python. <sup>4</sup>

**Coherency:** We quantify fluency or naturalness of a question using coherency. We measure coherency using a reference free metric called CTRL-Eval ([Ke et al., 2022](#)).

### C Annotation Screenshots

We present examples of annotation screenshots for different Isomorphisms that we considered for curating the expert-annotations from the GrailQA++ dataset. Each screenshot includes a pictorial 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.

---

<sup>4</sup><https://pypi.org/project/py-readability-metrics/>

## Iso-2

### S-expression

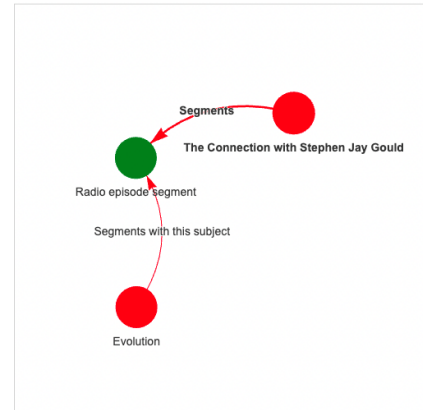
```
(AND radio.radio_episode_segment (AND (JOIN
(R radio.radio_subject.segments_with_this_subject) m.02j8z)
(JOIN (R radio.radio_program_episode.segments) m.0blhhfg)))
```

### Constraints and Relations

```
m.02j8z --> Evolution
m.0blhhfg --> The Connection with Stephen Jay Gould
radio.radio_episode_segment --> Radio episode segment
```

### Answer

```
m.0blhk6j --> Christopher Lydon with Stephen Jay Gould
```



## Iso-3

### S-expression

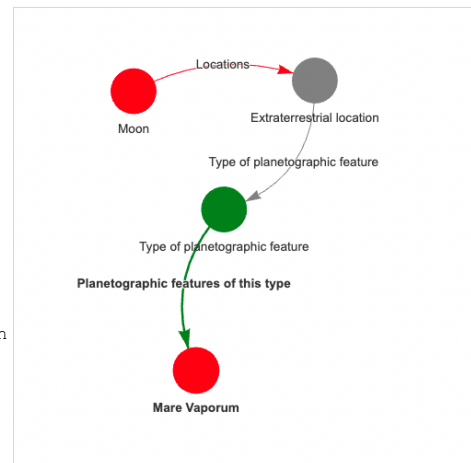
```
(AND astronomy.type_of_planetographic_feature (AND (JOIN
(R astronomy.extraterrestrial_location.
type_of_planetographic_feature)
(JOIN (R astronomy.celestial_object.locations) m.04wv_)
(JOIN astronomy.type_of_planetographic_feature.
planetographic_features_of_this_type m.01d80r)))
```

### Constraints and Relations

```
astronomy.type_of_planetographic_feature -->
Type of planetographic feature
astronomy.extraterrestrial_location --> Extraterrestrial location
m.04wv_ --> Moon
m.01d80r --> Mare Vaporum
```

### Answer

```
m.03dy13 --> Lunar mare
```



## Iso-5

### S-expression

```
(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))))
```

### Constraints and Relations

```
m.0452jfk --> Van
metropolitan_transit.transit_line --> Transit Line
metropolitan_transit.transit_stop --> Transit Stop
metropolitan_transit.transit_line --> Transit Line
```

### Answer

```
m.0452j59 --> Quartzsite Transit Service
m.0403pn4 --> Walnut Line
```

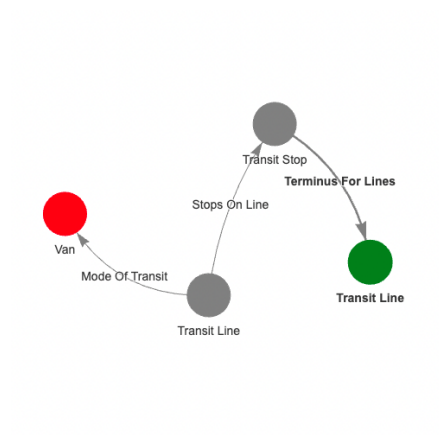


Figure 3: Annotation screenshots for three isomorphism categories; Iso-2 (top), Iso-3 (middle), and Iso-5 (bottom). For each instance, we provide the S-expression, the friendly name for each entity and relation, as well as the answers.

Iso-Code	Pictorial Desc	GrailQA		GraphQ		WebQSP		CWQ	
Iso-0		30496 (68.8)	2809 (77.9)	1533 (64.4)	640 (52.2)	1750 (58.3)	288 (43.8)	0 (0.0)	0 (0.0)
Iso-1		8900 (20.1)	559 (15.5)	544 (22.8)	391 (31.9)	716 (23.8)	197 (30.0)	5869 (21.8)	360 (14.5)
Iso-2		2676 (6.0)	135 (3.7)	167 (7.0)	37 (3.0)	106 (3.5)	19 (2.9)	4046 (15.0)	311 (12.5)
Iso-3		1066 (2.4)	18 (0.5)	69 (2.9)	50 (4.1)	35 (1.2)	6 (0.9)	6496 (24.1)	755 (30.4)
Iso-4		523 (1.2)	61 (1.7)	5 (0.2)	68 (5.6)	352 (11.7)	136 (20.7)	0 (0.0)	0 (0.0)
Iso-5		540 (1.2)	22 (0.6)	48 (2.0)	33 (2.7)	1 (0.0)	0 (0.0)	3712 (13.8)	279 (11.2)
Iso-6		42 (0.1)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	2454 (9.1)	324 (13.0)
Iso-7		26 (0.1)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	188 (0.7)	6 (0.2)
Iso-8		18 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.0)	0 (0.0)	644 (2.4)	76 (3.1)
Iso-9		12 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	7 (0.2)	2 (0.3)	217 (0.8)	19 (0.8)
Iso-10		15 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
Iso-11		23 (0.1)	0 (0.0)	0 (0.0)	6 (0.5)	4 (0.1)	1 (0.2)	225 (0.8)	17 (0.7)
Iso-12		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	30 (1.0)	8 (1.2)	0 (0.0)	0 (0.0)
Iso-13		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	69 (0.3)	14 (0.6)
Iso-14		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	810 (3.0)	111 (4.5)
Iso-15		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	57 (0.2)	9 (0.4)
Iso-16		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1102 (4.1)	117 (4.7)
Iso-17		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	501 (1.9)	49 (2.0)
Iso-18		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	319 (1.2)	27 (1.1)
Iso-19		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	67 (0.2)	3 (0.1)
Iso-20		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	5 (0.0)	0 (0.0)
Iso-21		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	72 (0.3)	10 (0.4)
Iso-22		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	12 (0.0)	0 (0.0)
Iso-23		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	13 (0.0)	0 (0.0)
Iso-24		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	9 (0.0)	0 (0.0)
Iso-25		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.0)	0 (0.0)
Iso-26		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	2 (0.0)	0 (0.0)
CIso-0		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
CIso-1		0 (0.0)	0 (0.0)	15 (0.6)	0 (0.0)	1 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
CIso-2		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	13 (0.0)	0 (0.0)
CIso-3		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	39 (0.1)	0 (0.0)
CIso-4		0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	2 (0.0)	0 (0.0)

Table 7: Distribution of different isomorphisms across the training and test splits for KBQA datasets. We include only instances in the test split that conform with the zero-shot criteria of GrailQA.