Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases

Yu Gu
The Ohio State University
gu.826@osu.edu

Sue Kase
U.S. Army Research Laboratory
sue.e.kase.civ@mail.mil

Michelle T. Vanni
U.S. Army Research Laboratory
michelle.t.vanni.civ@mail.mil

Brian M. Sadler
U.S. Army Research Laboratory
brian.m.sadler6.civ@mail.mil

Percy Liang
Stanford University
pliang@cs.stanford.edu

Xifeng Yan
University of California, Santa Barbara
xyan@cs.ucsb.edu

Yu Su
The Ohio State University
su.809@osu.edu

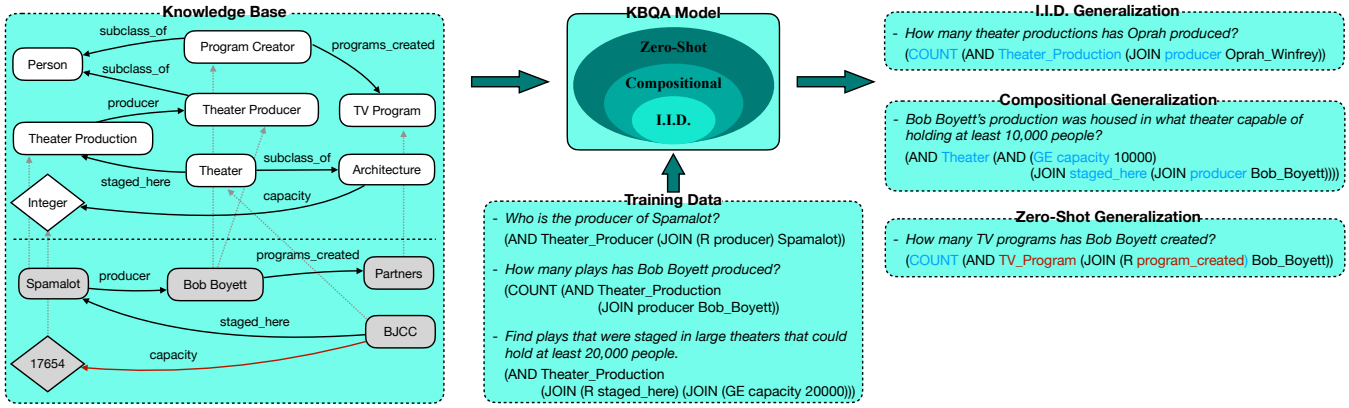


Figure 1: On large-scale KBs, collecting sufficient training data for KBQA to ensure i.i.d. distribution at test time is very difficult, if possible at all. We argue that practical KBQA models should have three levels of *built-in generalization* rather than solely relying on training data: (1) *i.i.d. generalization* to questions following the training distribution, (2) *compositional generalization* to novel compositions of schema items seen in training (marked blue), and (3) *zero-shot generalization* to unseen schema items or even domains (marked red). Our definition of generalization is based on the underlying logical forms (shown as S-expressions). Orthogonally, as illustrated by the examples, KBQA models should also have strong generalization to linguistic variation. Figure best viewed in color.

ABSTRACT

Existing studies on question answering on knowledge bases (KBQA) mainly operate with the standard i.i.d. assumption, i.e., training distribution over questions is the same as the test distribution. However, i.i.d. may be neither reasonably achievable nor desirable on large-scale KBs because 1) true user distribution is hard to capture and 2) randomly sampling training examples from the enormous space would be highly data-inefficient. Instead, we suggest that KBQA models should have three levels of built-in generalization: *i.i.d.*, *compositional*, and *zero-shot*. To facilitate the development of KBQA models with stronger generalization, we construct and release a new large-scale, high-quality dataset with 64,331 questions, GRAILQA, and provide evaluation settings for all three levels of generalization. In addition, we propose a novel BERT-based KBQA model. The combination of our dataset and model enables us to thoroughly examine and demonstrate, for the first time, the key role of pre-trained contextual embeddings like BERT in the generalization of KBQA.[1]

i.i.d.在大规模KB上可能既不合理也不可取

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449992

ACM Reference Format:
Yu Gu, Sue Kase, Michelle T. Vanni, Brian M. 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), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449992

[1] Data and leaderboard: http://dki-lab.github.io/GrailQA/
Code: https://github.com/dki-lab/GrailQA

1 INTRODUCTION

Question answering on knowledge bases (KBQA) has emerged as a promising technique to provide unified, user-friendly access to knowledge bases (KBs) and shield users from the heterogeneity underneath [4, 7, 22, 31]. As the scale and coverage of KBs increase, KBQA is becoming even more important due to the increasing difficulty of writing structured queries like SPARQL.[2]

There has been an array of datasets and models for KBQA in recent years [1, 4, 5, 12, 31, 34, 39, 40, 44]. Most existing studies are (implicitly) focused on the *i.i.d.* setting, that is, assuming training distribution is representative of the true user distribution and questions at test time will be drawn from the same distribution. While it is standard for machine learning, it could be problematic for KBQA on large-scale KBs. First, it is very difficult to collect sufficient training data to cover all the questions users may ask due to the broad coverage and combinatorial explosion. Second, even if one strives to achieve that, e.g., by iteratively annotating all user questions to a deployed KBQA system, it may hurt user experience because the system would keep failing on new out-of-distribution questions not covered by existing training data in each iteration.

现有KBQA方案很难保证iid，原因：1. 很难cover所有用户的问题表达 2. 用户新的表达在不断增加，无法覆盖到新的表达 ps:这里列举的例子paper都是19年前的（更准确地说都是18年前的，只有1篇19年）

Therefore, we argue that *practical KBQA models should be built with strong generalizability to out-of-distribution questions at test time*. More specifically, we propose three levels of generalization: *i.i.d.*, *compositional*, and *zero-shot* (Figure 1). In addition to the standard i.i.d. generalization, KBQA models should also generalize to novel compositions of seen schema items (relations, classes, functions). For example, if a model has been trained with questions about relations like producer, staged_here, capacity, classes like Theater, and functions like GE, it should be able to answer complex questions involving all these schema items even though this specific composition is not covered in training. Furthermore, a KBQA model may also encounter questions about schema items or even entire domains that are not covered in training at all (e.g., TV_Program and program_created) and needs to generalize in a zero-shot fashion [32].

High-quality datasets are of great importance for the community to advance towards KBQA models with stronger generalization. In addition to providing a benchmark for all three levels of generalization, an ideal dataset should also be large-scale, diverse, and capture other practical challenges for KBQA such as entity linking, complex questions, and language variation. However, existing KBQA datasets are usually constrained in one or more dimensions. Most of them are primarily focused on the i.i.d. setting [4, 6, 39, 45]. GRAPHQ [34] and QALD [31] could be used to test compositional generalization but not zero-shot generalization. They are also relatively small in scale and have limited coverage of the KB ontology. SIMPLEQ [6] is large in scale but only contains single-relational questions with limited diversity. A quantitative comparison can be found in Table 1.

Therefore, we construct a new large-scale, high-quality dataset for KBQA, named GRAILQA (Strongly Generalizable Question Answering), that supports evaluation of all three levels of generalization. It contains 64,331 crowdsourced questions involving up to 4 relations and functions like counting, comparatives, and superlatives, making it the largest KBQA dataset with complex questions to date. The dataset covers all the 86 domains in FREEBASE COMMONS, the part of FREEBASE considered to be high-quality and ready for public use, and most of the classes and relations in those domains. Furthermore, the questions in our dataset involve entities spanning the full spectrum of popularity from United_States_of_America to, e.g., Tune_Hotels — a small hotel chain mainly operated in Malaysia. To make the dataset even more diverse and realistic, we collect common surface forms of entities, e.g., *"Obama"* and *"President Obama"* for Barack_Obama, via large-scale web mining and crowdsourcing and use them in the questions. Finally, we develop multiple quality control mechanisms for crowdsourcing to ensure dataset quality, and the final dataset is contributed by 6,685 crowd workers with highly diverse demographics.

In addition to the dataset, we also study important factors towards stronger generalization and propose a novel KBQA model based on pre-trained language models like BERT [10]. We first show that our model performs competitively with existing models: On GRAPHQ, our model sets a new state of the art, beating prior models by a good margin (3.5%). On GRAILQA, our model significantly outperforms a state-of-the-art KBQA model. More importantly, the combination of our dataset and model enables us to thoroughly examine several challenges in KBQA such as search space pruning and language-ontology alignment. The comparison of different variants of our model clearly demonstrates the critical role of BERT in compositional and zero-shot generalization. To the best of our knowledge, *this work is among the first to demonstrate the key role of pre-trained contextual embeddings such as BERT at multiple levels of generalization for KBQA*. Finally, we also show that our dataset could serve as a valuable pre-training corpus for KBQA in general: pre-trained on our dataset, our model can generalize to other datasets (WEBQ) and reach similar performance fine-tuned with only 10% of the data, and can even generalize in a zero-shot fashion across datasets. To summarize, the key contributions of this paper are three-fold:

- We present the first systematic study on three levels of generalization, i.e., *i.i.d.*, *compositional*, and *zero-shot*, for KBQA and discuss their importance for practical KBQA systems.
- We construct and release to the public a large-scale, high-quality KBQA dataset containing 64K questions with diverse characteristics to support the development and evaluation of KBQA models with stronger generalization at all three levels.
- We propose a novel BERT-based KBQA model with competitive performance and thoroughly examine and demonstrate the effectiveness of pre-trained contextual embeddings in generalization. We also present fine-grained analyses which point out promising venues for further improvement.

2 BACKGROUND

2.1 Knowledge Base

A knowledge base consists of two parts, an ontology $O \subseteq C \times \mathcal{R} \times C$ and the relational facts $\mathcal{M} \subseteq \mathcal{E} \times \mathcal{R} \times (C \cup \mathcal{E} \cup \mathcal{L})$, where C is a set of classes, $\mathcal{E}$ is a set of entities, $\mathcal{L}$ is a set of literals and $\mathcal{R}$ is a set of binary relations. An example is shown in Figure 2, where the top part is a snippet of the FREEBASE ontology and the bottom part is some of the facts. We base our dataset on the latest version of FREEBASE. Although FREEBASE has stopped getting updated, it is still one of the largest publicly available KBs, and its high quality from human

[2] FREEBASE contains over 100 domains, 45 million entities, and 3 billion facts. GOOGLE KNOWLEDGE GRAPH has amassed over 500 billion facts about 5 billion entities [36].

curation makes it a solid choice for benchmarking KBQA. We use its COMMONS subset which contains 86 domains, 2,038 classes, 6,265 relations, and over 45 million entities.

2.2 Three Levels of Generalization: Definition

The definitions are based on the underlying logical form of natural language questions. We denote $\mathcal{S}$ as the full set of *schema items* that includes $\mathcal{R}$, C, and optionally a set of language-specific constructs from the meaning representation language (e.g., SPARQL). In our case we include the set of functions (Section 3). Note that entities and literals are not included. Denote $\mathcal{S}_{train}$ as the set of schema items in any of the training examples and $\mathcal{S}_q$ that of a question q. A qualifying test set $\mathcal{Q}$ for each level of generalization is defined as:

- **I.I.D. generalization**: $\forall q \in \mathcal{Q}, \mathcal{S}_q \subset \mathcal{S}_{train}$. In addition, the test questions follow the training distribution, e.g., randomly sampled from the training data.
- **Compositional generalization**: $\forall q \in \mathcal{Q}, \mathcal{S}_q \subset \mathcal{S}_{train}$, however, the specific logical form of q is not covered in training.
- **Zero-shot generalization**: $\forall q \in \mathcal{Q}, \exists s \in \mathcal{S}_q, s \in \mathcal{S} \setminus \mathcal{S}_{train}$.

对于任意测试集中的q，q的schema集合存在于训练集schema结合中，但q本身的if不再其中（即需要拼装组合才能得到）

The levels of generalization represent the expectation on KBQA models: A model should minimally be able to handle questions i.i.d. to what it has been trained with. One step further, it should also generalize to novel compositions of the seen constructs. Ideally, a model should also handle novel schema items or even entire domains not covered by the limited training data, which also includes compositions of novel constructs. By explicitly laying out the different levels of generalization, we hope to encourage the development of models built with stronger generalization. Orthogonal to these, practical KBQA models should also have strong generalization to language variation, i.e., different paraphrases corresponding to the same logical form. The dataset we will introduce next also puts special emphasis on this dimension.

3 DATA

In this section, we describe how our dataset, GRAILQA, is constructed and present analyses on its quality and diversity. How to split the dataset to create evaluation setups for different levels of generalization is described in Section 5.1.

3.1 Data Collection

Wang et al. [42] propose the OVERNIGHT approach which collects question-logical form pairs with three steps: (1) generate logical forms from a KB, (2) convert logical forms into canonical questions, and (3) paraphrase canonical questions into more natural forms via crowdsourcing. Su et al. [34] extend the OVERNIGHT approach to large KBs like FREEBASE with algorithms to generate logical forms that correspond to meaningful questions from the massive space. However, the data collection in Su et al. [34] was entirely done by a handful of expert annotators, which significantly limits its scalability and question diversity. We build on the approach in Su et al. [34] and develop a crowdsourcing framework with carefully-designed quality control mechanisms, which is much more scalable and yield more diversified questions. As a result, we are able to construct a dataset one order of magnitude larger with much better coverage of the FREEBASE ontology (Table 1). Our data collection pipeline is illustrated in Figure 2.

Canonical logical form generation. We leverage the logical form generation algorithm from Su et al. [34], which randomly generates logical forms with rich characteristics. The algorithm guarantees some important properties of the generated logical forms such as well-formedness and non-redundancy. It first traverses the KB ontology to generate graph-shaped templates that only consist of classes, relations, and functions, and then ground certain nodes to compatible entities to generate logical forms in their meaning representation called *graph query*[3]. At this stage, we only ground each template with one set of compatible entities to generate one *canonical logical form* and use it to drive the subsequent steps. Following prior work, we generate logical forms containing up to 4 relations and optionally containing one function selected from counting, superlatives (argmax, argmin), and comparatives ($>$, $\geq$, $<$, $\leq$). The exemplar logical form in Figure 2 has 3 relations and one function.

Each canonical logical form is validated by a graduate student against the criterion – *could we reasonably expect a real human user to ask this question?* We only keep the valid ones, which reduces the number of artificial questions that are common in other KBQA datasets with generated logical forms such as COMPLEXWEBQ [39] and SIMPLEQ [6] and makes the dataset more realistic. The overall approval rate at this step is 86.5%.

Canonical question annotation. Each validated canonical logical form is annotated with a canonical question by a graduate student, which is then cross-validated by another student to ensure its fidelity and fluency. The graduate students are all knowledgeable of KBQA and are trained with detailed materials about the annotation task. To further facilitate the task, we develop a graphical interface (Figure 7) which displays an interactive graphical visualization of the logical form. One can click on each component to see auxiliary information such as a short description and its domain/class. Annotators are required to enclose entities and literals in brackets (Figure 2).

Crowd-powered paraphrasing. We use Amazon Mechanical Turk to crowdsource paraphrases of the canonical questions and limit our task to native English speakers with at least 95% task approval rate. We develop a crowdsourcing framework with automated quality control mechanisms that contains three tasks (instructions in Figure 8):

- *Task 1: Paraphrasing*. A crowd worker is presented with a canonical question as well as auxiliary information such as topic entity description and answer to the question, and is tasked to come up with a plausible and natural paraphrase. Entities and literals enclosed with brackets are retained verbatim in the paraphrase. The system keeps running until 5 *valid* paraphrases (Task 2) are obtained for each canonical question. We pay $0.10 per paraphrase.
- *Task 2: Cross-validation*. Each paraphrase is then judged by multiple independent workers (the original author is excluded) on its fluency and fidelity (i.e., semantic equivalence) to the corresponding canonical question. Each paraphrase gets 4 judgements on average and those with fidelity approval rate below 75% *or* fluency approval rate below 60% are discarded. Overall 17.4% of paraphrases are discarded. A worker judges 10 paraphrases a time, one of which is a *control question*. We manually craft an initial set of paraphrase-canonical question

[3] Refer to Su et al. [34] for the syntax and semantics of graph query.

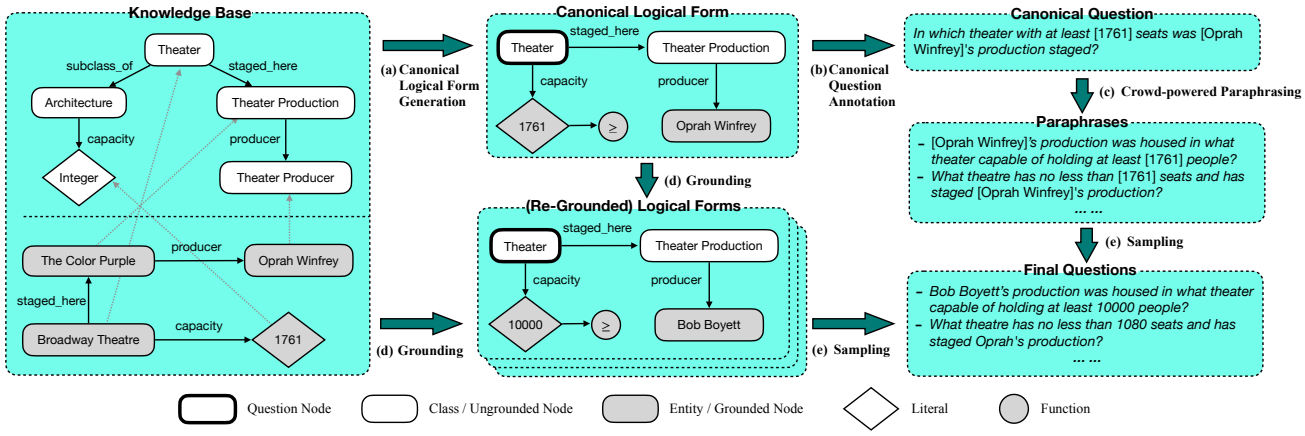


Figure 2: Our data collection pipeline with illustration of how one example question in Figure 1 is derived: (a) Generate canonical logical forms from the given KB up to the specified complexity. (b) Experts annotate canonical questions for the canonical logical forms. Entities and literals are enclosed in brackets so that they can be easily replaced. (c) Get high-quality and diverse paraphrases via crowdsourcing. (d) Generate different logical forms from each canonical logical form with different entity groundings. (e) Sample different combinations of logical form and paraphrase to generate the final questions. We additionally mine common surface forms of entities from the web, e.g., *"Oprah"* for Oprah_Winfrey, and use them in the final questions to make entity linking more realistic.

	Questions	Canonical LF	Domains	Relations	Classes	Entities	Literals	Generalization Assumption
WEBQ [4, 45]	4,737	794	56	661	408	2,593	47	i.i.d.
COMPLEXWEBQ [39]	34,689	6,236	61	1,181	501	11,840	996	i.i.d.
SIMPLEQ [6]	108,442	2,004	76	2,004	741	89,312	0	i.i.d.
QALD [31]	558	558	N/A	254	119	439	28	comp.
GRAPHQ [34]	5,166	500	70	596	506	376	81	comp.
GRAILQA	64,331	4,969	86	3,720	1,534	32,585	3,239	i.i.d. + comp.+ zero-shot

Table 1: Comparison of KBQA datasets. The number of distinct canonical logical forms (LFs) provides a view into the diversity of logical structures. For example, even though SIMPLEQ appears to be the largest, its diversity is limited because it only contains single-relational logical forms with no function (e.g., over 3,000 questions asking about the place_of_birth of different persons). Datasets other than GRAILQA are also likely to have test samples including schema items not covered in training, but that is not sufficient to support systematic evaluation of zero-shot generalization. See more details in Appendix C.

Question	Domain	Answer	# of Relations	Function
Beats of Rage is a series of games made for which platform?	Computer Video Game	DOS	1	none
Which tropical cyclone has affected Palau and part of Hong Kong?	Location, Meteorology	Typhoon Sanba	3	none
Marc Bulger had the most yards rushing in what season?	Sports, American Football	2008 NFL Season	3	superlative
How many titles from Netflix have the same genre as The Big Hustle?	Media Common	20,104	2	count
What bipropellant rocket engine has less than 3 chambers?	Spaceflight	RD-114 RD-112, ...	1	comparative

Table 2: Example questions from GRAILQA.

pairs known to be good/bad as control questions and gradually expand the set with new pairs validated by crowd workers. We ask workers to reconsider their validation decision of the whole batch when they have failed the latent control question, and we found that by doing so spammers or low-quality workers tend to quit the task and we end up with mostly high-quality workers. We pay $0.15 per batch.

- *Task 3: Entity surface form mining*. We collect a list of common surface forms ranked by frequency for each entity from a large entity linking dataset, FACC1 [14], which identifies around 10 billion mentions of FREEBASE entities in over 1 billion web documents from ClueWeb. For example, some common surface forms (and frequency) for Barack_Obama are *"obama"* (21M), *"barack obama"* (5.3M), and *"barack hussein obama"* (101K). We then ask at least three crowd workers to select true surface forms from the mined list, and those selected by less than 60% of the workers are discarded. We pay $0.05 per task.

Grounding and sampling. More logical forms are generated by grounding each canonical logical form with *compatible* entity groundings.[4] We do controlled sampling to generate the final questions: From the pool of logical forms and paraphrases associated with the same canonical logical form, we sample one from each pool at a time to generate a question (Figure 2). We start with uniform weights and each time a logical form or paraphrase is selected, its weight is divided by ρ_l and ρ_p, respectively. We set ρ_l to 2 and ρ_p to 10 to enforce more linguistic diversity. Finally, we randomly replace entity surface forms with the ones mined in Task 3 (if there is any). This way, we are able to generate a large-scale dataset with carefully-controlled quality and diversity.

3.2 Dataset Analysis

In total we have collected 4,969 canonical logical forms and 29,457 paraphrases (including the canonical questions). The final dataset contains 64,331 question-logical form pairs after sampling, which is by far the largest dataset on KBQA with complex questions. We note that such sampling is a unique feature of GRAILQA because of our data collection design. A detailed comparison with existing datasets is shown in Table 1 and some example questions are shown in Table 2. GRAILQA has significantly broader coverage and more unique canonical logical forms than existing dataset except COMPLEXWEBQ.[5] GRAILQA is also the only KBQA dataset that explicitly evaluates zero-shot generalization (Section 5.1). Question distributions over different characteristics are shown below:

# of Relations				Function				Answer Cardinality	
1	2	3	4	none	count	super.	comp.	1	> 1
44,340	16,610	3,254	127	53,526	3,463	4,111	3,231	44,044	20,287

Quality. As a qualitative study on data quality, we manually analyze 100 randomly sampled canonical logical forms, their canonical questions, and the associated paraphrases: 3 of the 100 canonical questions have mismatching meaning with the canonical logical form, mostly due to misinterpreting the directionality of some relation, and 12 of the 576 paraphrases do not match the corresponding canonical question, leading to 3% error rate of canonical question annotation, 2.1% error rate of paraphrasing, and 5.6% error rate overall. All the examined paraphrases are reasonably fluent.

Linguistic diversity. The dataset is contributed by 11 graduate students and 6,685 crowd workers with diverse demographics in terms of age group, education background, and gender (Appendix A). A common concern with crowd-powered paraphrasing as a means of data collection is lack of linguistic diversity — crowd workers may be biased towards the canonical question [16]. However, that may not be the case for GRAILQA. The average Jaccard similarity between each paraphrase and the corresponding canonical question, when all lower-cased, is 0.569 and 0.286 for unigrams and bigrams, respectively. The average Levenshtein edit distance is 26.1 (with average total length of 50.1 characters). The same statistics between the paraphrases of the same canonical question are 0.527, 0.257, and 28.2. The dissimilarity is even higher when excluding entities. We believe this is a decent level of linguistic diversity and we attribute it to our diverse crowd workers and quality control mechanisms.

Entity linking. GRAILQA is also featured with more realistic and challenging entity linking thanks to our large-scale mining of entity surface forms. Common types of surface forms include acronym (*"FDA"* for Food_and_Drug_Administration), last/first name (*"Obama"* for Barack_Obama), commonsense (*"Her Majesty the Queen"* for Elizabeth_II), and colloquial vs. formal (*"Obama vs. Romney"* for United_States_Presidential_Election_2012). In general the mined surface forms are the more typical, colloquial way of referring to an entity than its formal name in the KB, which makes the questions more realistic. We note that this is an important challenge towards practical KBQA systems but is largely neglected in existing KBQA datasets. For example, Yao [43] shows that, for the WEBQ dataset, simple fuzzy string matching is sufficient for named entity recognition (NER) due to the way the dataset is constructed.

3.3 Logical Form in S-expression

In addition to graph query, we provide an alternative linearized version of it in S-expressions, which is needed to apply the mainstream sequence-to-sequence (Seq2Seq) neural models [12, 20, 47]. We find that S-expression provides a good trade-off on compactness, compositionality, and readability, but one may choose any other formalism for linearization such as λ-DCS [26] or directly use the corresponding SPARQL queries. Every graph query can be easily converted to an equivalent S-expression. For example, The graph query in Figure 2(d) can be converted into (AND Theater (AND (GE capacity 10000) (JOIN staged_here (JOIN producer Bob_Boyett)))). More details can be found in Appendix B. Both graph queries and S-expressions can be easily converted into SPARQL queries to get answers. We will use S-expressions as logical form in our modeling.

4 MODELING

In this section, we discuss some of the challenges arising from non-i.i.d. generalization and potential solutions. The proposed models, combined with GRAILQA, will enable us to evaluate and compare different strategies for stronger generalization.

Compositional and zero-shot generalization pose two unique challenges compared with i.i.d. generalization: *large search space* and *language-ontology alignment*. The first challenge is the significantly larger search space. Under the i.i.d. assumption, a model only needs to consider the part of the ontology observed during training. For zero-shot generalization, however, one could not assume that to hold at test time and needs to consider the entire ontology. Models that build their vocabulary solely from training data (e.g., [47]) would almost certainly fail at zero-shot generalization. *Question-specific search space pruning* is thus more important than it is in the i.i.d. setting. Secondly, a major challenge in KBQA is to build a precise alignment between natural language and schema items in the ontology in order to understand what each question is asking about. While it is already important for i.i.d. generalization, it becomes even more critical for non-i.i.d. generalization: For zero-shot generalization, one apparently needs to understand the unseen schema items in order for it to possibly handle the corresponding questions. Even for

[4] An entity grounding, e.g., (Bob Boyett, 10000), is compatible with a canonical logical form if the re-grounded logical form yields a non-empty denotation on the KB.

[5] Logical forms in COMPLEXWEBQ are automatically extended from WEBQ with extra SPARQL statements. This adds structural diversity but may lead to unnatural questions.

compositional generalization, where the schema items are covered in training, precise language-ontology alignment is still critical in order for the model to generate novel compositions other than something it has memorized in training. Conventional methods [1, 4, 7] use web mining to build a lexicon from natural language phrases to KB schema items, which may be biased towards popular schema items. The emergence of pre-trained contextual embeddings such as BERT [10] provides an alternative solution to building language-ontology alignment. Next, we propose a BERT-based KBQA model which will enable us to examine the effectiveness of pre-trained contextual embeddings in non-i.i.d. generalization.

4.1 Model Overview

Our goal is to learn a model that maps an input question $q = x_1, ..., x_{|q|}$ to its corresponding logical form $a = y_1, ..., y_{|a|}$. The entire model is based on Seq2Seq [2, 38], which is commonly used in many semantic parsing tasks [12, 15, 19, 20, 48]. Seq2Seq comprises an encoder that encodes input q into vectors and a decoder that autoregressively output one token $y' \in \mathcal{V}$ conditioned on the input representation, where $\mathcal{V}$ is the decoding vocabulary. Specifically, the conditional probability $p(a|q)$ is decomposed as:

将问题定义为一个seq2seq序列生成问题
输入query，输出logical form
本论文中encode和decode都采用LSTM

$$p(a|q) = \prod_{t=1}^{|a|} p(y_t|y_{<t}, q), \tag{1}$$

where $y_{<t} = y_1, ..., y_{t-1}$. Our encoder and decoder are two different recurrent neural networks with long short-term memory units (LSTM) [17]. LSTM recursively processes one token at each time step as the following function:

$$\mathbf{h}_t = LSTM(\mathbf{h}_{t-1}, \mathbf{g}_t), \tag{2}$$

where $\mathbf{g}_t$ is the representation of the input token at the t-th step, and $\mathbf{h}_{t-1}$ is the hidden state from last time step. During encoding, $\mathbf{g}_t = f(q, \mathcal{V}, t)$, with f being a function takes $q, \mathcal{V}, t$ as input and returns the embedding for x_t; during decoding, $\mathbf{g}_t = [\mathbf{W}]_{y_t}$, which denotes the the embedding corresponding to y_t in the embedding matrix $\mathbf{W} \in \mathbf{R}^{|\mathcal{V}| \times d}$. We will elaborate how we use BERT to define f and $\mathbf{W}$ in the next subsection.

During decoding, the model computes the probability for each output token based on the current hidden state $\mathbf{h}_t$ and $\mathbf{W}$:

$$p(y_t|y_{<t}, q) = [Softmax(\mathbf{W}\mathbf{h}_t)]_{y_t}. \tag{3}$$

This means we tie the input and output embeddings using $\mathbf{W}$ in Seq2Seq terminology. In this way, we can assign semantic meanings to words from pre-trained embeddings, which can facilitate the open vocabulary learning [41].

4.2 BERT Encoding

Now we discuss how we use BERT to compute $\mathbf{W}$ and $f(q, \mathcal{V}, t)$. BERT has demonstrated its effectiveness in learning good alignment in cross-domain text-to-SQL parsing with its contextual representation. Specifically, in text-to-SQL, Hwang et al. [19] successfully apply BERT by concatenating the question and all vocabulary items together to construct the input for BERT. However, the vocabulary in KBQA is much larger. Directly doing this will exceed BERT's maximum input length of 512 tokens. To address this, we split items from $\mathcal{V}$ into chunks and then concatenate each chunk independently with q, and feed them to BERT one by one. Figure 3 depicts an example where each chunk has 2 items from $\mathcal{V}$. Specifically, q is separated with each chunk by a special token "[SEP]", and the items inside the same chunk are delimited by ";". Then $f(q, \mathcal{V}, t)$ can be simply defined as the t-th output from BERT for the first chunk. For $\mathbf{W}$, each row in it is computed by taking average over BERT's output for the constituting words of the corresponding item. For example, the row corresponding to architecture.venue.capacity in $\mathbf{W}$ is computed by averaging the output from BERT for word *"architecture"*, *"venue"* and *"capacity"* in the first chunk as shown in Figure 3.

Vocabulary construction. We have not talked about what constitutes $\mathcal{V}$. The most trivial way is just to include all schema items from KB ontology to deal with zero-shot generalization. However, this will not only lead to an enormous search space during decoding that makes the prediction very difficult, but also makes it extremely time-consuming for training as we need to create a large number of chunks to feed to BERT for every single question. To reduce the size of $\mathcal{V}$, we leverage entities identified from q as anchors in the KB, and choose to only include KB items that are reachable by at least one of the anchor entities within 2 hops in KB. We refer to this as vocabulary pruning (VP), which is applied during both training and inference. This means we have a dynamic $\mathcal{V}$ for different input q. Note that all the identified entities, functions, and syntax constants used in S-expressions are also included. We similarly get the representation for an entity based on the output from BERT that corresponds to its surface form.

4.3 Entity Linking

Entity linking in most of the existing KBQA datasets is not a major challenge, and exhaustive fuzzy string matching [43] may suffice to achieve a reasonable performance. However, the entities in GRAILQA span the full spectrum of popularity and appear in surface forms mined from the web, which is more realistic but also makes entity linking more challenging. We use a BERT-based NER system[6] and train it on our training set. For entity disambiguation from the identified mentions, we simply select the most popular one(s) based on FACC1. As a comparison, we also try Aqqu [3], a rule-based entity linker using linguistic and entity popularity features and achieves high accuracy on WEBQ. The entity linking results on GRAILQA is shown as follows:

	Recall	Precision	F1
Aqqu	**77.8**	9.7	17.2
BERT	77.0	**68.0**	**72.2**

The BERT-based entity linker is slightly worse in recall but much better in precision. We will therefore use the BERT-based entity linker afterwards.

4.4 Inference

We propose two different modes of doing inference using our model, namely TRANSDUCTION and RANKING. For TRANSDUCTION, we simply use the model in the normal way, i.e., use the model to

[6] https://github.com/kamalkraj/BERT-NER

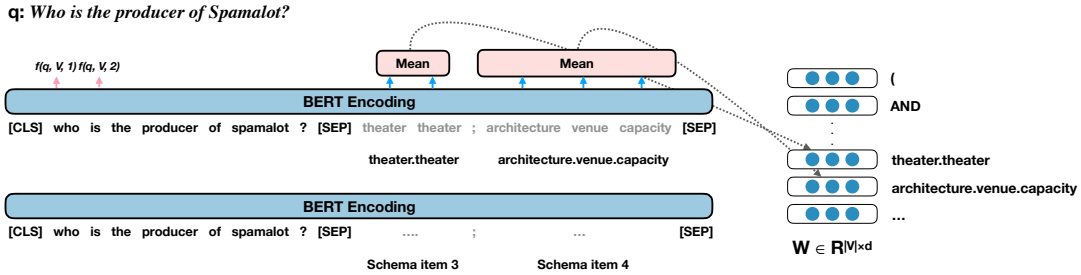


Figure 3: An overview of using BERT to jointly encode the input question and schema items in the decoder vocabulary. We evenly split all items in the vocabulary into chunks and concatenate each chunk with the question one by one. For the purpose of visualization, here we set the chunk size as 2. The first chunk contains two schema items, theater.theater and architecture.venue.capacity. For each item in $\mathcal{V}$, we compute its embedding in W by averaging the last layer's output of BERT of all its word-pieces. To compute $f(q, \mathcal{V}, t)$, we simply take the last layer's output of BERT in the first chunk for the t-th word-piece of the question. Note that, we show whole words instead of actual word-pieces in the figure for brevity.

autoregressively predict one token from $\mathcal{V}$ at each time step until the model predicts the stop token. For RANKING, instead of using Seq2Seq as a generator we use it as a ranker to score each candidate logical form and return the top-ranked candidate. We employ a simple yet effective strategy to generate candidate logical forms: We enumerate all logical forms, optionally with a count function, within 2 hops starting from each entity identified in the question.[7] The recall is 80% on GRAILQA. Questions with superlatives and comparatives often do not have a topic entity and are therefore not covered. RANKING can prune the search space more effectively than the first one, while TRANSDUCTION is more flexible and can handle more types of questions.

Effectiveness on existing dataset. Before getting to experiments on our GRAILQA dataset, we conduct an experiment on the existing dataset GRAPHQ to show the competitive performance of our model. Our RANKING model achieves an F1 of 25.0%, which significantly outperforms the prior art SPARQA [37] by 3.5 percent. With this superior performance, we believe our new model is reasonable to serve as a strong baseline on GRAILQA and support the subsequent investigations on three levels of generalization.

5 EXPERIMENTS

5.1 Experimental Setup

Data split. We split GRAILQA to set up evaluation for all three levels of generalization. Specifically, our training/validation/test sets contain about 70%/10%/20% of the data, which correspond to 44,337, 6,763 and 13,231 questions, respectively. For the validation and test sets, 50% of the questions are from held-out domains not covered in training (**zero-shot**), 25% of the questions correspond to canonical logical forms not covered in training (**compositional**), and the rest 25% are randomly sampled from training (**i.i.d.**). The i.i.d. and compositional subsets have the additional constraint that the involved schema items are all covered in training. For the zero-shot subset, 5 domains are held out for validation and 10 for test.

Evaluation metrics. We use two standard evaluation metrics. The first one is *exact match accuracy* (EM), i.e., the percentage of questions where the predicted and gold logical forms are semantically equivalent. To determine semantic equivalence, we first convert S-expressions back to graph queries and then determine graph isomorphism.[8] Unlike string-based or set-based EM [46], our graph isomorphism based EM is both sound and complete. In addition, we also report the F1 score based on the predicted and gold answer sets [4]. This is better suited for models that directly predict the final answers without any intermediate meaning representation and gives partial credits to imperfect answers. We host a local SPARQL endpoint via Virtuoso to compute answers.

5.2 Models

Our primary goal of the experiments is to thoroughly examine the challenges of different levels of generalization and explore potential solutions. Therefore, we will evaluate an array of variants of our models with different strategies for search space pruning and language-ontology alignment. To better situate our models in the literature and demonstrate their competitive performance, we also adapt QGG [24], the state-of-the-art model on COMPLEXWEBQ and WEBQ, to GRAILQA. We have also looked into models developed based on QALD or LC-QUAD but the adaptation cost would be too high because most of the source codes are not available. All models use the same entity linker (Section 4.3).

Our model variants. For both TRANSDUCTION and RANKING, we introduce a variant that uses GloVe embeddings[9] instead of BERT for language-ontology alignment. Specifically, we use the average GloVe embedding for each schema item instead of the encodings from BERT. Unlike BERT, GloVe embeddings are not contextual and are thus not jointly encoded with the current natural language question. This variant will then allow us to examine the role of contextual embedding in generalization. For TRANSDUCTION, we additionally introduce a variant that does not employ entity-based vocabulary pruning (VP). This is not applicable to RANKING because RANKING always relies on the identified entities for candidate generation. We use uncased BERT-base and fine-tune BERT but fix GloVe (similar to previous work [15]), because we find fine-tuning GloVe makes it slightly worse. Other implementation details such as hyper-parameters and training scheme can be found in Appendix D.

[7] There are prohibitively many candidates with 3 hops.

[8] https://networkx.github.io

[9] We choose to use GloVe-6B-300d.

	Overall		I.I.D.		Compositional		Zero-shot	
	EM	F1	EM	F1	EM	F1	EM	F1
QGG [24]	–	36.7	–	40.5	–	33.0	–	36.6
TRANSDUCTION	33.3	36.8	51.8	53.9	31.0	36.0	25.7	29.3
– VP	26.6	29.6	40.1	42.7	25.6	30.0	20.6	23.4
– BERT	17.6	18.4	50.5	51.6	16.4	18.5	3.0	3.1
– VP – BERT	15.4	16.1	48.3	49.1	13.5	15.3	1.0	1.3
RANKING	**50.6**	**58.0**	59.9	67.0	**45.5**	**53.9**	**48.6**	**55.7**
– BERT	39.5	45.1	**62.2**	**67.3**	40.0	47.8	28.9	33.8

Table 3: Overall results. "– VP" denotes without vocabulary pruning. "– BERT" denotes using GloVe embeddings instead of BERT.

QGG. This model learns to generate query graphs from question-answer pairs using reinforcement learning. BERT is also used but only to get a matching score between question and logical form via joint encoding, which is served as one of the seven hand-crafted features used for ranking. To train this model on GRAILQA, we use the same configuration from the original paper for WEBQ, i.e., considering up to 2-hop relations in the KB for candidate generation. This can cover 80% of the training questions in GRAILQA. Increasing it to 3 hops will take a few months (estimated) to train this model due to the large number of entities in GRAILQA. We only report F1 for QGG because it uses a different meaning representation.

5.3 Results

5.3.1 Overall Evaluation. We show the overall results in Table 3. RANKING achieves the best overall performance on GRAILQA. Both of our models outperform QGG, demonstrating their competitive performance. We also observe a significant performance drop on all the variants of our models, which suggest that both BERT encoding and VP play an important role.

Next we drill down to different levels of generalization. The results clearly demonstrate the key role of contextualized encoding via BERT for compositional and zero-shot generalization: While BERT and GloVe perform similarly in i.i.d. generalization, using GloVe instead of BERT, TRANSDUCTION's F1 drops by 17.5% in compositional generalization and by 26.2% in zero-shot generalization. For RANKING, it also brings a 21.9% drop in zero-shot generalization. We do a more in-depth analysis on TRANSDUCTION and confirm that this is mainly because BERT enables better generalization to unseen schema items. For example, for question *"What home games did the Oakland Raiders play?"* about an unseen domain american_football, TRANSDUCTION can find the correct relation american_football.football_game.home_team, while TRANSDUCTION without BERT is confused by the wrong relation sports.sports_team.arena_stadium, which is seen during training. This is a common type of error by GloVe-based models but much less frequent by BERT-based models, which shows how pre-trained contextual embeddings help address the *language-ontology alignment* challenge. Since QGG is also a ranking model that uses BERT, it is not surprising it also performs reasonably in compositional and zero-shot generalization.

On the other hand, RANKING outperforms TRANSDUCTION significantly in compositional and zero-shot generalization. This is largely due to RANKING's effectiveness in *search space pruning*, i.e., RANKING prunes the search space based on the each identified entity's neighboring facts, while TRANSDUCTION only uses the ontology, which is less discriminating. VP helps TRANSDUCTION to some extent, but still not quite up to the same level of effective pruning RANKING enjoys. This also provides a plausible explanation for the interesting observation that BERT seems to play a less significant role in compositional generalization in ranking models than in transductive models — because of the effective search space pruning, RANKING's candidate set includes much less of the logical forms seen in training that may be confusing to the model.

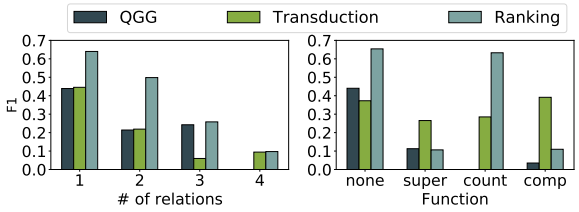


Figure 4: Fine-grained results.

5.3.2 Fine-Grained Evaluation. We now present a more fine-grained analysis along several other dimensions in Figure 4.

Structural complexity. The performance of all models degrades rapidly as questions become more complex. Note that, though our RANKING model only generates candidates with up to 2 relations, it still has a chance to get partially correct on more complex questions. In fact, it even outperforms the more flexible TRANSDUCTION model on questions with 3 relations, which again demonstrates the importance of effective search space pruning.

Function. Ranking models (including RANKING and QGG) rely on topic entities for candidate generation which are often absent in questions with comparatives or superlatives, e.g., *"which chemical element was first discovered?"* Their performance suffers as a result. QGG gets zero F1 on questions that requires counting since it only returns entities as answer. On the other hand, TRANSDUCTION performs significantly better than the other two models on superlatives and comparatives due to its flexibility in generating all types of logical forms, though that comes with a penalty on simpler questions. Better guided search for both RANKING and TRANSDUCTION may be a promising future direction to enjoy the best of both worlds.

5.3.3 Robustness Analysis. A good KBQA model should be robust to different paraphrases and entity groundings (e.g., should not succeed at *"Where is the Trump tower?"* but fail at *"Where is the Tune Hotels?"*). GRAILQA provides an opportunity to directly

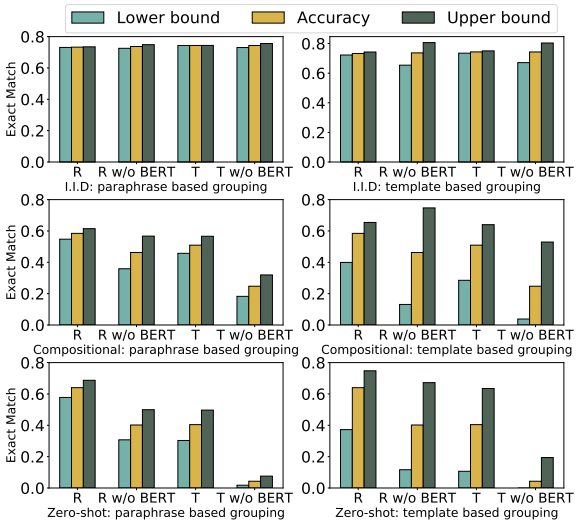


Figure 5: Robustness analysis. T denotes TRANSDUCTION and R denotes RANKING. For robustness analysis, we assume correct entities are given.

test that. We create different groupings of questions: *grouping by paraphrase* — questions that are the same except for their entity groundings fall into the same group, and *grouping by template* — questions from the same canonical logical form but with different groundings or paraphrases are grouped together. Each grouping provides upper and lower bounds for evaluating model robustness. The upper bound is derived by treating all the questions in a group as correct if any of them is correctly answered, while the lower bound is derived by treating all the questions as incorrect if any of them is incorrectly answered. The true accuracy should lie between the bounds. The closer the three are, the more robust a model is to the corresponding mutation. To reduce the number of variables in this analysis, we assume perfect entity linking and eliminate entity linking errors. The results are shown in Figure 5, which clearly show that BERT-based models are generally more robust than GloVe-based models (reflected by the smaller gap between their upper and lower bounds) and provides further explanation to its superior performance in non-i.i.d. generalization. It is interesting to observe that the RANKING model yields a higher upper bound with GloVe than with BERT in non-i.i.d. generalization, which shows that the major problem with GloVe is its robustness to variations.

5.4 Error Analysis

We analyze 100 randomly sampled questions that our RANKING model gets wrong to discuss venues for future improvement. We summarize the errors into the following categories.

Coverage limitation (34%): Due to the diversity of questions in GRAILQA in terms of both complexity and functions, candidates generated by RANKING can only cover around 80% of the questions, as it can only enumerate up to 2-relational logical forms due to combinatorial explosion. Uncovered questions constitute 34% of errors. More intelligent ways for search and candidate generation are promising future directions.

Entity linking (33%): The large-scale mining of entity surfaces forms makes entity linking a unique challenge in GRAILQA. For example, in *"After apr. the 2nd, 2009, which song was added to RB?"*, *"RB"* refers to the music video game Rock Band, while our entity linker fails to identify any entity from this question. Also, less popular entities form another source of entity linking errors. To be specific, popularity is a very strong signal for entity disambiguation in previous datasets, but not as strong in GRAILQA because many questions are intentionally not just about very popular entities. For *"What is the belief of Mo?"* our entity linker can identify the entity mention *Mo*, and the most popular entity that has surface form *Mo* is the state of Missouri, while the question is asking about the Mo religion. Accurate entity linking on large-scale KBs is still a major challenge.

Relation mis-classification (26%): Another major type of errors comes from misclassified relations. Precisely understanding a vaguely or implicitly expressed relation is challenging, especially for zero-shot generalization. For example, question *"Which video game engine is the previous version of Frostbite 2?"* is from an unseen domain cvg, and the correct relation for this question should be cvg.computer_game_engine.successor_engine, while RANKING predicts cvg.computer_game_engine.predecessor_engine, which fails to capture the alignment between successor_engine and *"previous version"*. Besides, around 20% of relation mis-classifications are due to that the model cannot correctly determine the answer type. For question *"Name the soundtrack that Amy Winehouse features."*, which is asking about a music soundtrack, but RANKING predicts a relation associated with the answer type music.album.

Other (7%): The rest of the errors mainly include typing errors and function errors. For typing errors, the question can seek for information about more specific types (e.g., looking for politician instead of just person), while RANKING does not explicitly handle type constraint as introducing type constraints by enumeration will result in prohibitively many candidate logical forms. A more flexible way of handling type constraint is in need. Also, sometimes RANKING can be confused by the function of a question. For instance, for question *"Name the image which appears in the topic gallery armenian rock."*, there shouldn't be any aggregation function, however, RANKING mistakenly predicts a counting function for it.

5.5 Transfer Learning

We also show that GRAILQA could serve as a valuable pre-training corpus for KBQA in general by pre-training RANKING on GRAILQA and testing its transferability to WEBQ, which contains naturally-occurring questions from Google search log. To adapt RANKING to WEBQ, we first convert SPARQL queries in WEBQ to our S-expression logical forms. We test three settings: Full training that fine-tunes using all training data of WEBQ, few-shot that only uses 10%, and zero-shot that directly applies models trained on GRAILQA to WEBQ. For simplicity, we assume perfect entity linking. The results are shown below. Pre-trianing on GRAILQA uniformly improves the performance, especially in the low-data regime. Most remarkably, pre-trained on GRAILQA, RANKING can achieve an F1 of 43% without *any* in-domain training on WEBQ. These results provide further supporting evidence for the quality and diversity of GRAILQA.

		Exact Match	F1
Full Training	RANKING	0.59	0.67
	+pre-training	**0.60**	**0.70**
Few-shot	RANKING	0.44	0.53
	+pre-training	**0.55**	**0.65**
Zero-shot	RANKING	0.00	0.00
	+pre-training	**0.35**	**0.43**

6 RELATED WORK

Existing KBQA datasets. There have been an array of datasets for KBQA in recent years. WEBQ [4] is collected from Google search logs with the answers annotated via crowdsoursing. Yih et al. [45] later provide logical form annotation for the questions in WEBQ. Most of the questions in WEBQ are simple single-relational questions, and Talmor and Berant [39] generate more complex questions by automatically extending the questions in WEBQ with additional SPARQL statements, leading to the COMPLEXWEBQ dataset. However, such automatic extension may lead to unnatural questions. SIMPLEQ [6] is another popular KBQA dataset created by sampling individual facts from FREEBASE and annotating them as natural language questions. It therefore only contains single-relational questions. GRAPHQ [34] is the most related to ours. It proposes an algorithm to automatically generate logical forms from large-scale KBs like FREEBASE with guarantees on well-formedness and non-redundancy. We also use their algorithm for logical form generation, and develop a crowdsourcing framework which significantly improves the scalability and diversity. The aforementioned datasets are all based on FREEBASE. QALD [31] and LC-QUAD [40] are popular datasets on DBPEDIA. QALD is manually created by expert annotators, while LC-QUAD first generates SPARQL queries and unnatural canonical questions with templates and have expert annotators paraphrase the canonical questions.

Most KBQA datasets primarily operate with the i.i.d. assumption because their test set is a randomly sampled subset of all the data [4, 6, 39, 40, 45]. GRAPHQ [34] is set up to primarily test compositional generalization by splitting their training and test sets on logical forms, similarly for QALD [31]. However, none of the existing datasets supports the evaluation of all three levels of generalization. Our dataset also compares favorably to existing datasets in other dimensions such as size, coverage, diversity (Table 1).

Existing models on KBQA. KBQA models can be roughly categorized into semantic-parsing based methods and information-retrieval methods. The former maps a natural language utterance into a logical form that can be executed to get the final answer, while the latter directly ranks a list of candidate entities without explicitly generating a logical form. We focus on semantic-parsing methods due to their superior performance and better interpretability. Existing methods are mainly focused on i.i.d. setting and are limited in generalizability. Conventional rule-based methods [4] suffer from the coverage of their hand-crafted rules and therefore have rather limited generalizability. More recent models either employ encoder-decoder framework [12, 20, 47] to decode the logical form auto-regressively or first generate a set of candidate logical forms according to predefined templates or rules, and then match each candiate with the utterance to get the best matched one[1, 4, 9, 11, 18, 24, 28, 29, 33, 37, 44]. These models already achieve impressive results on i.i.d. dataset like WEBQ, however, on GRAPHQ that mainly tests compositional generalization, the best model can only achieve an F1 of 21.5 [37]. This demonstrates that non-i.i.d. generalization in KBQA has not drawn enough attention from existing methods.

Pre-training and non-i.i.d. generalization. The problem of non-i.i.d. generalization has drawn attention under related semantic parsing settings. Su and Yan [35] recognize the key role of pre-trained word embeddings [30] in cross-domain semantic parsing. Contextual embeddings like BERT are later shown to be successful for cross-domain text-to-SQL parsing [19]. Another line of work has been focusing on compositional generalization on a number of specifically-synthesized datasets [21, 23]. Concurrent to this work, Furrer et al. [13] find that pre-trained contextual embeddings plays a more vital role in compositional generalization than specialized architectures. To the best of our knowledge, our work is among the first to demonstrate the key role of contextual embeddings like BERT at multiple levels of generalization for KBQA.

7 CONCLUDING REMARKS

In this paper, we explicitly lay out and study three levels of generalization for KBQA, i.e., i.i.d., compositional, and zero-shot generalization. We construct and release GRAILQA, a large-scale, high-quality KBQA dataset with 64,331 questions that can be used to evaluate all three levels of generalization. We also propose a novel BERT-based KBQA model. The combination of our dataset and model enables us to thoroughly examine and demonstrate, for the first time, the different challenges and promising solutions in non-i.i.d. generalization of KBQA. In particular, we demonstrate the key role of pre-trained contextual embeddings like BERT in compositional and zero-shot generalization of KBQA.

This work is just a starting point towards building more practical KBQA models with stronger generalization. It opens up a number of future directions. First, for a full-fledged QA system on large-scale KBs, more sophisticated, context-sensitive entity linkers that can work for long-tail entities and variation in surface forms are needed. Entity linking is of particular importance because topic entities provide important anchors in the large KB to dramatically prune the search space, but is also more challenging due to the large number of entities. Second, for complex questions, the search space is still prohibitively large if we enumerate candidate logical forms in a brute-force way. More intelligently guided search is a promising future direction to efficiently generate the most promising candidates and prune less promising ones. Last but not least, even though we have empirically demonstrated how pre-trained contextual embeddings like BERT significantly facilitate compositional and zero-shot generalization by providing better language-ontology alignment, there is still a lack of deeper understanding on why that is the case, which may inspire better ways of exploiting these models. We are also interested in experimenting with other pre-trained contextual embeddings such as RoBERTa [27] and BART [25].

8 ACKNOWLEDGEMENTS

We would like to thank Jefferson Hoye for helping with the crowdsourcing pipeline, the graduate students who helped with canonical question annotation, and the anonymous reviewers for their helpful comments. This research was sponsored in part by NSF 1528175, a Fujitsu gift grant and Ohio Supercomputer Center [8].

REFERENCES

[1] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum. 2017. Automated Template Generation for Question Answering over Knowledge Graphs. In *WWW*.

[2] D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[3] H. Bast and E. Haussmann. 2015. More Accurate Question Answering on Freebase. In *CIKM*.

[4] J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.

[5] N. Bhutani, X. Zheng, and H. Jagadish. 2019. Learning to answer complex questions over knowledge bases with query composition. In *CIKM*.

[6] A. Bordes, N. Usunier, S. Chopra, and J. Weston. 2015. Large-scale Simple Question Answering with Memory Networks. *CoRR* abs/1506.02075 (2015).

[7] Q. Cai and Al. Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *ACL*.

[8] Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. `http://osc.edu/ark:/19495/f5s1ph73`

[9] W. Cui, Y. Xiao, H. Wang, Y. Song, S. Hwang, and W. Wang. 2017. KBQA: Learning Question Answering over QA Corpora and Knowledge Bases. *Proc. VLDB Endow.* (2017).

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[11] D. Diefenbach, A. Both, K. Singh, and P. Maret. 2020. Towards a question answering system over the semantic web. *Semantic Web* (2020).

[12] L. Dong and M. Lapata. 2016. Language to Logical Form with Neural Attention. In *ACL*.

[13] D. Furrer, M. van Zee, N. Scales, and N. Schärli. 2020. Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. *arXiv preprint arXiv:2007.08970* (2020).

[14] E. Gabrilovich, M. Ringgaard, and A. Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1.

[15] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J. Lou, T. Liu, and D. Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *ACL*.

[16] J. Herzig and J. Berant. 2019. Don't paraphrase, detect! Rapid and Effective Data Collection for Semantic Parsing. In *EMNLP-IJCNLP*.

[17] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).

[18] Y. Hua, Y. Li, G. Qi, W. Wu, J. Zhang, and D. Qi. 2020. Less is more: Data-efficient complex question answering over knowledge bases. In *Journal of Web Semantics*.

[19] W. Hwang, J. Yim, S. Park, and M. Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. *arXiv preprint arXiv:1902.01069* (2019).

[20] R. Jia and P. Liang. 2016. Data Recombination for Neural Semantic Parsing. In *ACL*.

[21] D. Keysers, N. Schärli, N. Scales, H. Buisman, D. Furrer, S. Kashubin, N. Momchev, D. Sinopalnikov, L. Stafiniak, T. Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713* (2019).

[22] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-Fly Ontology Matching. In *EMNLP*.

[23] B. Lake and M. Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*.

[24] Y. Lan and J. Jiang. 2020. Query Graph Generation for Answering Multi-hop Complex Questions from Knowledge Bases. In *ACL*.

[25] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[26] P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408* (2013).

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[28] K. Luo, F. Lin, X. Luo, and K. Zhu. 2018. Knowledge Base Question Answering via Encoding of Complex Query Graphs. In *EMNLP*.

[29] G. Maheshwari, P. Trivedi, D. Lukovnikov, N. Chakraborty, A. Fischer, and J. Lehmann. 2019. Learning to rank query graphs for complex question answering over knowledge graphs. In *ISWC*.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

[31] Ngonga Ngomo. 2018. 9th challenge on question answering over linked data (QALD-9). *language* (2018).

[32] M. Palatucci, D. Pomerleau, G. E Hinton, and T. M Mitchell. 2009. Zero-shot learning with semantic output codes. In *NIPS*.

[33] S. Reddy, O. Täckström, S. Petrov, M. Steedman, and M. Lapata. 2017. Universal Semantic Parsing. In *EMNLP*.

[34] Y. Su, H. Sun, B. Sadler, M. Srivatsa, I. Gür, Z. Yan, and X. Yan. 2016. On Generating Characteristic-rich Question Sets for QA Evaluation. In *EMNLP*.

[35] Y. Su and X. Yan. 2017. Cross-domain Semantic Parsing via Paraphrasing. In *EMNLP*.

[36] Danny Sullivan. 2020. A reintroduction to our Knowledge Graph and knowledge panels. `https://blog.google/products/search/about-knowledge-graph-and-knowledge-panels/`.

[37] Y. Sun, L. Zhang, G. Cheng, and Y. Qu. 2020. SPARQA: Skeleton-based Semantic Parsing for Complex Questions over Knowledge Bases. In *AAAI*.

[38] I. Sutskever, O. Vinyals, and Q. V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*.

[39] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *NAACL*.

[40] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *ISWC*.

[41] S. Venugopalan, L. Anne Hendricks, M. Rohrbach, R. Mooney, T. Darrell, and K. Saenko. 2017. Captioning images with diverse objects. In *CVPR*.

[42] Y. Wang, J. Berant, and P. Liang. 2015. Building a Semantic Parser Overnight. In *ACL*.

[43] X. Yao. 2015. Lean Question Answering over Freebase from Scratch. In *NAACL: Demonstrations*.

[44] W. Yih, M. Chang, X. He, and J. Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *ACL*.

[45] W. Yih, M. Richardson, C. Meek, M. Chang, and J. Suh. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In *ACL*.

[46] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *EMNLP*.

[47] H. Zhang, J. Cai, J. Xu, and J. Wang. 2019. Complex Question Decomposition for Semantic Parsing. In *ACL*.

[48] R. Zhang, T. Yu, H. Er, S. Shim, E. Xue, X. V. Lin, T. Shi, C. Xiong, R. Socher, and D. Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. In *EMNLP-IJCNLP*.

A CROWD WORKER DEMOGRAPHICS

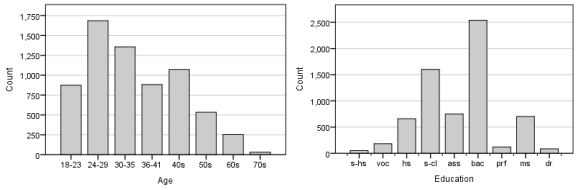


Figure 6: Demographics of participating crowd workers: age (top) and level of education (bottom). s-hs = some high school, voc = vocational school, hs = high school, s-cl = some college, ass = associate degree, bac = bachelor degree, prf = professional degree, ms = master degree, dr = doctoral degree.

Our data collection included 6,685 crowd workers recruited through Amazon Mechanical Turk (AMT). Each worker completed a short anonymous demographic survey requesting information on their gender, age, and completed level of education. Gender representation was fairly even, with only slightly higher participation by females (59.1%) than by males (40.9%). The crowd workers also exhibited a diverse demographic in terms of both age and level of education, as shown in Figure 6. These facts provide a strong indication of the high degree of diversity in this dataset.

B S-EXPRESSION

Our S-expressions employ set-based semantics: Each function takes a number of arguments, and both the arguments and the denotation of the functions are either a set of entities or entity tuples. Function definitions are listed in Table 4. There are 3 argument types in total,

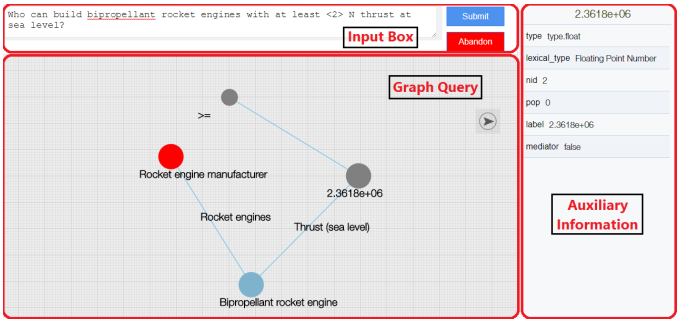


Figure 7: Interface for canonical question annotation.

namely, a set of entities, a set of (entity, entity) tuples and a set of (entity, value) tuples. Classes and single entities are the most fundamental examples of set of entities, and relations are the most fundamental examples of set of binary tuples. By applying those functions defined in our grammar, we are able to get more complex set of entities and binary tuples.

Function	Returns	Description
(**AND** u1 u2)	a set of entities	**AND** function returns the intersection of two arguments
(**COUNT** u)	a singleton set of integer	**COUNT** function returns the cardinality of the argument
(**R** b)	a set of (entity, entity) tuples	**R** function reverse each binary tuple (x, y) in the input to (y, x)
(**JOIN** b u)	a set of entities	Inner join based on items in u and the second element of items in b
(**JOIN** b1 b2)	a set of (entity, entity) tuples	Inner join based on the first element of items in b2 and the second element of items in b1
(**ARGMAX** u b) (**ARGMIN** u b)	a set of entities	Return x in u such that (x, y) is in b and y is the largest / smallest
(**LT** b n) (**LE** b n) (**GT** b n) (**GE** b n)	a set of entities	Return all x such that (x, v) in b and v $<$ / $\leq$ / $>$ / $\geq$ n

Table 4: Functions defined in our S-expressions; u: a set of entities, b: a set of (entity, entity) tuples, n: a numerical value.

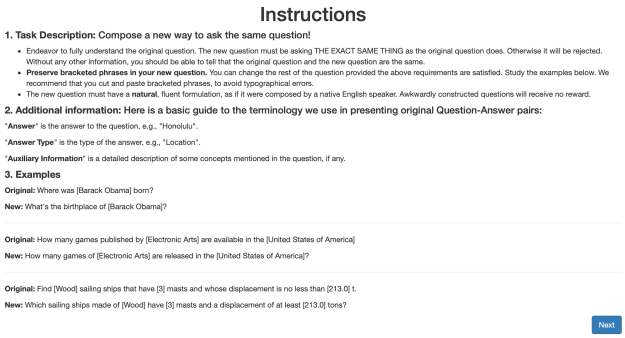
C DETAILED INFORMATION OF TABLE 1

It is not trivial to make statistics of question types, relations, classes, entities and relations for all datasets. Retrieving all those information from GRAPHQ and SIMPLEQ is straightforward as those information are almost explicitly given in these two datasets. WEBQ also provides inferential chain that connects the topic entity and the answer entity together with a list of other constraints. Based on these information, we are able to retrieve all the information with no much effort. On the other hand, QALD and COMPLEXWEBQ only provide SPARQL queries with no structured information. It's a little bit challenging to get the information from them. Specifically, we rely on hand-crafted rules (including regular expressions) to retrieve entities, relations and literals. Then we rely on SPARQL algebra to determine whether two question fall into the same type after replacing all entities and literals with placeholders. Also, not all datasets have explict class constraint in their queries like our dataset, so to count the number of different classes, we simply take into account the domain class and range class of all relations in the dataset.

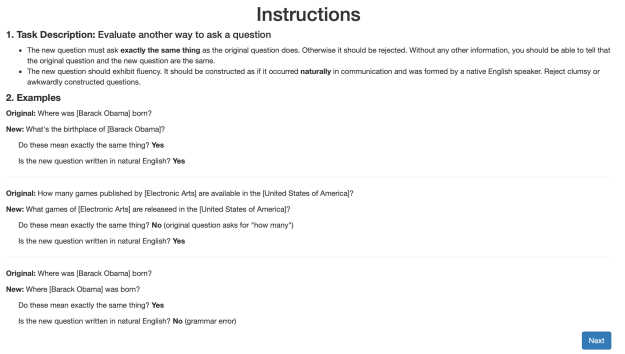
D DETAILED EXPERIMENTAL SETUP

D.1 Hyper-parameters

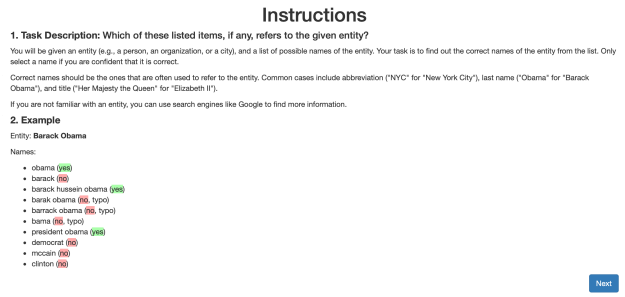
We train GloVe-based models with batch size 32 and an initial learning rate of 0.001 using Adam optimizer. For BERT-based models, we are only able to train our model with batch size 1 due to the memory consumption, so we choose a workaround to set the number



(a) Task 1: Paraphrasing.



(b) Task 2: Cross-validation.



(c) Task 3: Entity surface form mining.

Figure 8: Crowdsourcing instructions.

of gradient accumulation to be 16. We also use Adam optimizer with an initial learning rate of 0.001 to update our own parameters in BERT-based models. For BERT's parameters, we fine-tune them with a learning rate of 2e-5. For all models, the hidden sizes of both encoder and decoder are set to be 768. All hyper-parameters are selected based on the validation set.

D.2 Training Scheme

We use early stopping to train all our models and choose the models with highest exact match on the validation set. The patience is set as 3. In total, we train 2 different models, i.e. GloVe-based Seq2Seq model and BERT-based Seq2Seq model. Note that, one difference in training GloVe-based model is that there is no need to do vocabulary pruning during training.