RnG-KBQA: Generation Augmented Iterative Ranking for Knowledge Base Question Answering

Anonymous ACL submission

Abstract

Existing KBQA approaches, despite achieving strong performance on i.i.d. test data, often struggle in generalizing to questions involving unseen KB schema items. Prior ranking-based approaches have shown some success in generalization, but suffer from the coverage issue. We present RnG-KBQA, a **R**ank-a**n**d-**G**enerate approach for KBQA, which remedies the coverage issue with a generation model while preserving a strong generalization capability. Our approach first uses a contrastive ranker to rank a set of candidate logical forms obtained by searching over the knowledge graph. It then introduces a tailored generation model conditioned on the question and the top-ranked candidates to compose the final logical form. We achieve new state-of-the-art results on GRAILQA and WEBQSP datasets. In particular, our method surpasses the prior state-of-the-art by a large margin on the GRAILQA leaderboard. In addition, RnG-KBQA outperforms all prior approaches on the popular WEBQSP benchmark, even including the ones that use the oracle entity linking. The experimental results demonstrate the effectiveness of the interplay between ranking and generation, which leads to the superior performance of our proposed approach across all settings with especially strong improvements in zero-shot generalization.[1]

1 Introduction

Modern knowledge bases (KB) are reliable sources of a huge amount of world knowledge but can be difficult to interact with since they are extremely large in scale and require specific query languages (e.g., Sparql) to access. Question Answering over Knowledge Base (KBQA) serves as a user-friendly way to query over KBs and has garnered increasing attention (Berant et al., 2013; Cai and Yates, 2013). Recent research has attempted to build systems

---
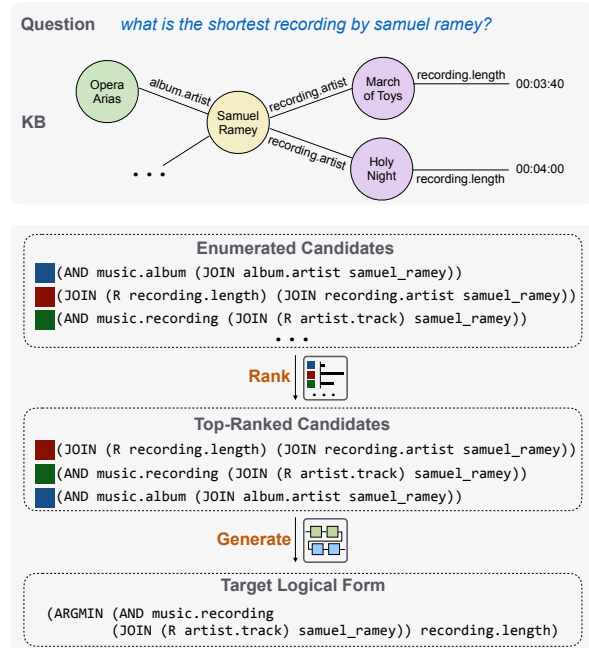
[1]Code available upon release.

Figure 1: Overview of our rank-and-generate approach. Given a question, we first rank logical form candidates obtained by searching over the KB based on predefined rules. Here, the ground truth logical form is not in the top-ranked candidates as it is not covered by the rules. We solve this problem using another generation step that produces the correct logical form based on top-ranked candidates. The final logical form is executed over the KB to yield the answer.

achieving strong results on several public benchmarks that contain i.i.d. train and test distribution such as SIMPLEQ (Bordes et al., 2015) and WEBQSP (Yih et al., 2016). However, users often want to ask questions involving unseen compositions or KB schema items (see Figure 5 for examples), which still remains a challenge. Generation-based approaches (e.g., a seq-to-seq parser) are not effective enough to handle such practical generalization scenarios due to the difficulty of generating unseen KB schema items. Ranking-based approaches, which first generate a set of candidate logical forms using predefined rules and then select the best-scored one according to the question, have

shown some success (Gu et al., 2021). However, it suffers from the coverage problem, because it is often impractical to exhaust all the rules to cover the desired logical form due to the scale of the KB.

We propose RNG-KBQA, a new framework targeted at generalization problems in the task of KBQA. Our approach combines a ranker with a generator, which addresses the coverage issue in ranking-only based approaches while still benefiting from their generalization power. As shown in Figure 1, we first employ a ranker to select a set of related logical forms from a pool of candidate logical forms obtained by searching over the graph. The selected logical forms are not required to cover the correct one, but are semantically coherent and aligned with the underlying intents in the question. Next, we introduce a generator that consumes both the question and the top-k ranked candidates to compose the final logical form. The core idea of our approach is the interplay between the ranker and the generator: the ranker provides essential ingredients of KB schema items to the generator, which then further refines the top-candidates by complementing missing constructions or constraints, and hence allows covering a broader range of logical form space.

We base both our ranker and generator on pre-trained language models for better generalization capability. Unlike prior systems which rank candidates using a grammar-based parser (Berant et al., 2013) or a seq-to-seq parser (Gu et al., 2021), our ranker is a BERT-based (Devlin et al., 2019) bi-encoder (taking as input question-candidate pair) trained to maximize the scores of ground truth logical forms while minimizing the scores of incorrect candidates. Such training schema allows learning from the contrast between the candidates in the entire territory, whereas prior parsing-based ranker (Berant et al., 2013; Gu et al., 2021) only learns to encourage the likelihood of the ground truth logical forms. We further develop an iterative-bootstrap-based training curriculum for efficiently training the ranker to distinguish spurious candidates (Section 2.2). In addition, we extend the proposed logical form ranker, keeping the architecture and logic the same, for the task of entity disambiguation, and show its effectiveness as a second-stage entity ranker. Our generator is a T5-based (Raffel et al., 2020) seq-to-seq model that fuses semantic and structural ingredients found in top-k candidates to compose the final logical form. To achieve this, we

feed the generator with the question followed by a linearized sequence of the top-k candidates, which allows it to distill a refined logical form that will fully reflect the question intent by complementing the missing pieces or discarding the irrelevant parts without having to learn the low-level dynamics.

We test RNG-KBQA on two datasets, GRAILQA and WEBQSP, and compare against an array of strong baselines. On GRAILQA, a challenging dataset focused on generalization in KBQA, our approach sets the new state-of-the-art performance of 68.8 exact match 74.4 F1 score, surpassing prior SOTA (58.1 exact match and 65.3 F1 score) by a large margin. On the popular WEBQSP dataset, RNG-KBQA also outperforms the best prior approach (QGG (Lan and Jiang, 2020)) and achieves a new SOTA performance of 75.7 F1 score. The results demonstrate the effectiveness of our approach across all settings and especially in compositional generalization and zero-shot generalization.

2 Generation Augmented KBQA

2.1 Preliminaries

A knowledge base collects knowledge data stored in the form of subject-relation-object triple (s, r, o), where s is an entity, r is a binary relation, and o can be entities or literals (e.g., date time, integer values, etc.). Let the question be x, our task is to obtain a logical form y that can be executed over the knowledge base to yield the final answer. Following Gu et al. (2021), we use *s-expressions* to represent queries over knowledge base. S-expression (examples in Figure 1) uses functions (e.g., JOIN) operating on set-based semantics and eliminates variable usages as in lambda DCS (Liang, 2013). This makes s-expression a suitable representation for the task of KBQA because it balances readability and compactness (Gu et al., 2021).

Enumeration of Candidates Recall that our approach first uses a ranker model to score a list of candidate logical forms $C = \{c_i\}_{i=1}^m$ obtained via enumeration. We'll first introduce how to enumerate the candidates before delving into the details of our ranking and generation models.

We start from every entity detected in the question and query the knowledge base for paths reachable within two hops. Next, we write down an s-expression corresponding to each of the paths, which constitutes a set of candidates. We note that
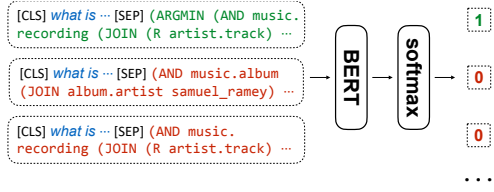
Figure 2: The ranker that learns from the contrast between the ground truth and negative candidates.

Figure 3: The generation model conditioned on question and top-ranked candidates returned by the ranker.

we do not exhaust all the possible compositions when enumerating (e.g., we do not include comparative operations and argmin/max operations), and hence does not guarantee to cover the target s-expression. A more comprehensive enumeration method is possible but will introduce a prohibitively large number (greater than 2,000,000 for some queries) of candidates. Therefore, it's impractical to cover every possible logical form when enumerating, and we seek to tackle this issue via our tailored generation model.

2.2 Logical Form Ranking

Our ranker model learns to score each candidate logical form by maximizing the similarity between question and ground truth logical form while minimizing the similarities between the question and the negative logical forms (Figure 2). Specifically, given the question x and a logical form candidate c, we use a BERT-based encoder that takes as input the concatenation of the question and the logical form and outputs a logit representing the similarity between them formulated as follows:

$$s(x, y) = \text{Linear}(\text{BertCls}([x; y]))$$

where BertCls denotes the [CLS] representation of the concatenated input; Linear is a projection layer reducing the representation to a scalar similarity score. The ranker is then optimized to minimize the following loss function:

$$\mathcal{L}_{\text{ranker}} = -\frac{e^{s(x,y)}}{e^{s(x,y)} + \sum_{c \in C \land c \neq y} e^{s(x,c)}} \quad (1)$$

where the idea is to promote the ground truth logical form while penalizing the negative ones via a contrastive objective. In contrast, the ranker employed in past work (Gu et al., 2021), a seq-to-seq model, aims to directly map the question to target logical form, only leveraging supervision from the ground truth. Consequently, our ranker is more effective in distinguishing the correct logical forms from *spurious ones* (similar but not equal to the ground truth ones).
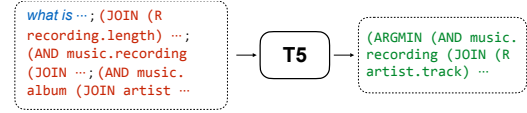
Bootstrapping Negative Samples in Training Due to the large number of candidates and limited GPU memory, it is impractical to feed all the candidates $c \in C$ as in Eq (1) when training the ranker. Therefore, we need to sample a subset of negatives logical forms $C' \subset C$ at each batch. A naive way for sampling negative logical forms is to draw random samples. However, because the number of candidates is often large compared to the allowed size of negative samples in each batch, it may not be possible to cover spurious logical forms within the randomly selected samples.

We propose to sample negative logical forms by bootstrapping, inspired by the negative sampling methods used in Karpukhin et al. (2020). That is, we first train the ranker using random samples for several epochs to warm start it, and then choose the spurious logical forms that are confusing to the model as the negative samples for further training the model. We find the ranker can benefit from this advanced negative sampling strategy and perform better compared to using random negative samples.

2.3 Target Logical Form Generation

Having a ranked list of candidates, we introduce a generation model to compose the final logical form conditioned on the question and the top-k logical forms. Our generator is a transformer-based seq-to-seq model (Vaswani et al., 2017) instantiated from T5 ((Raffel et al., 2020)), as it demonstrates strong performance in generation-related tasks. As shown in Figure 3, we construct the inputs by concatenating the question and the top-k candidates returned by the ranker separated by semi-colon (i.e., $[x; c_{t_1}; ...; c_{t_k}]$). We train the model to generate the ground truth logical form autoregressively with cross-entropy objective using teacher forcing. In the inference, we use beam-search to decode top-k target logical forms. To construct the top-k logical form candidates needed for training the generator, we first train the ranker, and then use the rankings it produces on the training data.

Since the generation model can now leverage both the question and KB schema information (contained in the candidates), the context is much more
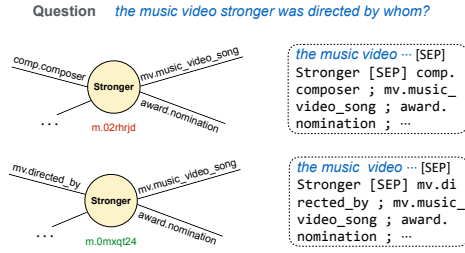
Figure 4: Illustrative example of running entity disambiguation as ranking. A confusing entity (red) and the correct entity (green) both match the surface form in the question. To distinguish them, we train an entity disambiguation model following the same architecture as in logical form ranking but construct inputs by concatenating the question and relations.

specified as compared to only conditioning on the question. This enables our generator to leverage the training data more efficiently by focusing only on correcting or supplementing existing logical forms instead of learning both the generation rule and correctness of logical forms.

Execution-Augmented Inference We use a vanilla T5 generation model without syntactic constraints, which does not guarantee the syntactic correctness nor executability of the produced logical forms. Therefore, we use an execution-augmented inference procedure, which is commonly used in prior semantic parsing related work (Devlin et al., 2017; Ye et al., 2020b). We first decode top-k logical forms using beam search and then execute each logical form until we find one that yields a valid (non-empty) answer. In case that none of the top-k logical forms is valid, we return the top-ranked candidate obtained using the ranker as the final logical form, which is guaranteed to be executable. This inference schema can ensure finding one valid logical form for each problem. It is possible to incorporate a more complex mechanism to control the syntactic correctness in decoding (e.g., using grammar-based decoder (Rabinovich et al., 2017) or dynamical beam pruning techniques (Ye et al., 2020a)). We leave such extension aside since we find that executability of produced logical forms is not the bottleneck (see Section 3.3 in experiments).

2.4 Extension: Entity Disambiguation as Ranking

Our ranking model is mainly proposed for the task of ranking candidate logical forms. Here, we introduce a simple way to adapt our ranking model for the task of entity disambiguation. A common paradigm of finding KB entities referred in a ques-

tion is to first detect the entity mentions with an NER system and then run fuzzy matching based on the surface forms. This paradigm has been employed in various methods (Yih et al., 2015; Sun et al., 2019; Chen et al., 2021; Gu et al., 2021). One problem with this paradigm lies in entity disambiguation: a mention usually matches surface forms of more than one entities in the KB.

A common way to disambiguate the matched entities is to choose the most popular one according to the popularity score provided by FACC1 project (Chen et al., 2021; Gu et al., 2021), which can be imprecise in some cases. We show an example in Figure 4. Consider the question "the music video stronger was directed by whom?" taken from GRAILQA, where the most popular matched entity is "Stronger" (m.02rhrjd, song by Kanye West)" and the second is also "Stronger" (m.0mxqqt24, music video by Britney Spears). The surface form matching and popularity scores do not provide sufficient information needed for disambiguation.

However, it is possible to leverage the relation information linked with an entity to further help assess if it matches a mention in the question. By querying relations over KB, we see there is a relation about mv director mv.directed_by linking to m.0mxqqt24, but there are no such kind of relations connected with m.02rhrjd. We therefore cast the disambiguation problem to an entity ranking problem, and adapt the ranking model used before to tackle this problem. Given a mention, we concatenate the question with the relations for each entity candidate matching the mention. We reuse the same model architecture and loss function as in Section 2.2 to train another entity disambiguation model to further improve the ranking of the target entity. We apply our entity disambiguation model on GRAILQA, and achieve substantial improvements in terms of entity linking.

3 Experiments

We mainly test our approach on GRAILQA (Gu et al., 2021), a challenging dataset focused on evaluating the generalization capabilities. We also experiment on WEBQSP and compare against a number of prior approaches to demonstrate the general applicability of our approach.

3.1 Experiment: GRAILQA

GRAILQA is the first dataset that evaluates the zero-shot generalization. Specifically, GRAILQA

	Overall		I.I.D.		Compositional		Zero-Shot	
	EM	F1	EM	F1	EM	F1	EM	F1
QGG (Lan and Jiang, 2020)	–	36.7	–	40.5	–	33.0	–	36.6
Bert Transduction (Gu et al., 2021)	33.3	36.8	51.8	53.9	31.0	36.0	25.7	29.3
Bert Ranking (Gu et al., 2021)	50.6	58.0	59.9	67.0	45.5	53.9	48.6	55.7
ArcaneQA (Anonymous)	57.9	64.9	76.5	79.5	56.4	63.5	50.0	58.8
ReTrack (Chen et al., 2021)	58.1	65.3	84.4	87.5	61.5	70.9	44.6	52.5
S2QL (Anonymous)	57.5	66.2	65.1	72.9	54.7	64.7	55.1	63.6
RnG-KBQA (Ours)	**68.8**	**74.4**	**86.2**	**89.0**	**63.8**	**71.2**	**63.0**	**69.2**
w/o Entity Disambiguation	61.4	67.4	78.0	81.8	55.0	63.2	56.7	63.0

Table 1: Exact match (EM) and F1 scores on the test split of GRAILQA. The numbers of other approaches are taken from the leaderboard. RNG-KBQA substantially outperforms prior methods by a large margin.

Figure 5: Examples of compositional generalization to new composition of KB schema items and zero-shot generalization to unseen schema items (red).

contains 64,331 questions in total and carefully splits the data so as to evaluate three levels of generalization in the task of KBQA, including *i.i.d.* setting, *compositional* generalization to unseen composition, and *zero-shot* generalization to unseen KB schema (examples in Figure 5). The fraction of each setting in the test set is 25%, 25%, and 50% , respectively. Aside from the generalization challenge, GRAILQA also presents additional difficulty in terms of the large number of involved entities/relations, complex compositionality in the logical forms (up to 4 hops), and noisiness of the entities mentioned in questions (Gu et al., 2021).

Implementation Detail We link an entity mention to an entity node in KB using our approach described in Section 2.4. We first use a BERT-NER systems provided by the authors of GRAILQA to detect mention spans in the question. For each mention span, we match the span with surface forms in FACC1 project (Gabrilovich et al., 2013), rank the matched entities using popularity score, and retain the top-5 entity candidates. Lastly, we use the disambiguation model trained on GRAILQA to select only one entity for each mention. Our entity ambulation model is initiated from BERT-base-uncased

model provided by huggingface library (Wolf et al., 2020), and finetuned for 3 epochs with a learning rate of 1e-5 and a batch size of 8.

When training the ranker, we sample 96 negative candidates using the strategy described in Section 2.2. Our ranker is finetuned from BERT-base-uncased for 3 epochs using a learning rate of 1e-5 and a batch size of 8. We do bootstrapping after every epoch. It is also noteworthy that we perform teacher-forcing when training the ranker, i.e., we use ground truth entity linking for training.

We base our generation model on T5-base (Raffel et al., 2020). We use top-5 candidates returned by the ranker and finetune for 10 epochs using a learning rate of 3e-5 and a batch size of 8.

Metrics For GRAILQA, we use exact match (EX) and F1 score (F1) as the metrics, all of which are computed using official evaluation script.

Results Table 1 summarizes the results on GRAILQA. The results of other approaches are directly taken from the leaderboard.[2] Overall, our approach sets the new state-of-the-art performance on GRAILQA dataset, achieving 68.8 EM and 74.4 F1. This exhibits a large margin over the other approaches: our approach outperforms ReTrack (Chen et al., 2021) by 10.7 EM and 8.2 F1.

Furthermore, RNG-KBQA performs generally well for all three levels of generalization and is particularly strong in zero-shot setting. Our approach is slightly better than ReTrack and substantially better than all the other approaches in i.i.d. setting and compositional setting. However, ReTrack fails in generalizing to unseen KB Schema items and only achieves poor performance in zero-shot setting, whereas our approach is generalizable and beats ReTrack with a margin of 16.1 F1.

[2]Accessed on 11/10/2021.

	F1	EM	Hits[@1]
PullNet* (Sun et al., 2019)	62.8	–	67.8
GraftNet* (Sun et al., 2018)	–	–	68.1
Bert Ranking* (Gu et al., 2021)	67.0	–	–
EmbedQA* (Saxena et al., 2020)	–	–	72.5
ReTrack* (Chen et al., 2021)	74.7	–	74.6
Topic Units (Lan et al., 2019)	67.9	–	68.2
UHop (Chen et al., 2019)	68.5	–	–
NSM (Liang et al., 2017)	69.0	–	–
ReTrack (Chen et al., 2021)	71.0	–	71.6
STAGG (Yih et al., 2015)	71.7	63.9	–
CBR (Das et al., 2021)	72.8	70.0	–
QGG (Lan and Jiang, 2020)	74.0	–	–
RNG-KBQA (**Ours**)	**75.6**	**71.1**	–

Table 2: Results of RNG-KBQA and baselines on WEBQSP. * (approach in the top section) denotes using oracle entity linking annotations provided by the dataset. Our approach achieves the new state-of-the-art performance (75.6 F1) with a discernible margin over the performance of best prior method (74.0 F1 obtained by QGG). Our approach even outperforms a number of prior work using oracle entity linking annotations.

To directly compare the effectiveness of our rank-and-generate framework against rank-only baseline (BERT Ranking), we also provide the performance of a variant of RNG-KBQA without the entity-disambiguation model. In this variant we directly use the entity linking results provided by the authors of Gu et al. (2021). Under the same entity linking performance, our ranking-and-generation framework is able to improve the performance by 9.7 EM and 8.2 F1. Furthermore, even without the entity-disambiguation module, our model still substantially outperforms all other approaches, even when some of them (e.g., ReTrack) use a better entity linking system (Chen et al., 2021). Please refer to the supplementary materials for more details on entity linking performance.

3.2 Experiment: WEBQSP

WEBQSP is a popular dataset which evaluates KBQA approaches in i.i.d. setting. It contains 4,937 question in total and requires reasoning chains with up to 2 hops. Since there is no official development split, we randomly sample 200 examples from the training set for validation.

Implementation Detail For experiments on WEBQSP, we use ELQ (Li et al., 2020) as the entity linker, which is trained on WEBQSP dataset to perform entity detection and entity linking, since it produces more precise entity linking results and hence leads to less number of candidate logical forms for each question. Because ELQ always links a mention to only one entity, we do not need an entity-disambiguation step for WEBQSP dataset. Similarly, we initiate the logical form ranker using BERT-base-uncased, and the generator using T5-base. We also sample 96 negative candidates for each question, and feed the top-5 candidates to the generation model. The ranker is trained for 10 epochs and we run bootstrapping every 2 epochs; the generator is trained for 20 epochs.

Metrics F1 is used as the main evaluation metric. In addition, for approaches that are able to select entity sets as answers, we report the exact match (EM) used in the official evaluation. For information-retrieval based approaches that can only predict a single entity, we report Hits[@1] (if the predicted entity is in the ground truth entity set), which is considered as a loose approximation of EM.

Results For baseline approaches, we directly take the results reported in corresponding original paper. As shown in Table 1, RNG-KBQA achieves 75.6 F1, surpassing the prior state-of-the-art (QGG) by 1.6. Our approach also achieves the best EM score of 71.1, surpassing CBR (Das et al., 2021). The performance of our approach obtained using ELQ-predicted entity linking outperforms all the prior methods, even if they are allowed to use oracle entity linking annotations (denoted as * in the top section). It is also noteworthy that both CBR and QGG, the two methods achieving strong performance closest to ours, use an entity linker with equal or better performance compared to ours. In particular, CBR also uses ELQ for entity linking. QGG uses an entity linker achieving 85.2 entity linking F1 (calculated using public available code) which is slightly better than ours achieving 84.8 entity linking F1. To summarize, the results on WEBQSP suggest that, in addition to outstanding generalization capability, our approach is also as strong in solving simpler questions in i.i.d. setting.

3.3 Analysis

Ablation Study We first compare the performance of our full model against incomplete ablations in Table 3. We derive a generation-only (Gen Only) model from our base model by replacing the trained ranker with a random ranker, which leads to a performance drop of 27.5 and 5.7 on GRAILQA and WEBQSP, respectively. The performance deterioration is especially sharp on GRAILQA as it requires generalizing to unseen KB schema items,

	GRAILQA	WEBQSP
Full Model	75.1	75.6
Gen Only (Rand Rank)	47.6	69.9
Rank Only	69.8	72.7
Rank Only (w/o Bootstrap)	68.6	71.3

Table 3: F1 scores on GRAILQA (dev set) and WE-BQSP of three ablations, including a generation-only variant (Gen Only, which uses randomly selected logical form candidates), a ranking-only variant (Rank Only), and a ranking-only variant without using bootstrap training strategy (w/o Bootstrap). Removing either component leads to performance deterioration.
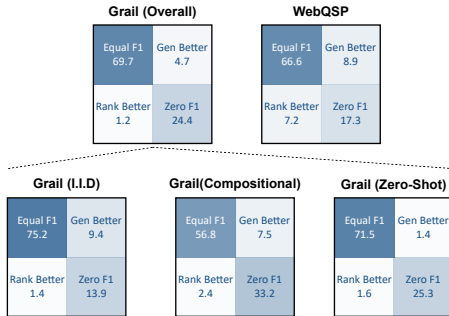

Figure 6: Comparison between the ranker's top predictions and the generator's top predictions. Generation model mostly keeps or improves the prediction while occasionally introducing errors.

for which the generator typically needs to be based on a good set of candidates to be effective.

To test the effects of our generation step, we compare the performance of a ranking-only variant (directly using the top-ranked candidate) against the performance of the full model. As shown in Table 3, the generation model is able to remedy some cases not addressable by the ranking model alone, which boosts the performance by 5.3 on GRAILQA and 2.9 on WEBQSP.

We additionally evaluate the performance of a ranking model trained without bootstrapping strategy introduced in Section 2.2. The performance of this variant lags its counterpart by 1.2 and 1.4 on GRAILQA and WEBQSP, respectively. The bootstrapping strategy is indeed helpful for training the ranker to better distinguish spurious candidates.

Comparing Outputs of Ranking Model and Generation Model We have demonstrated the benefit of adding a generation stage on top of the ranking step on previous result sections. Here, we present a more detailed comparison between the outputs of ranking model and generation model. Figure 6 presents the "comparison matrices" show-

Generation Better Than Ranking
(a) Q *what is the shortest recording by samuel ramey?*
R `(AND music.recording (JOIN recording.artist ramey))`
G `(ARGMIN (AND music.recording (JOIN recording.artist ramey)) recording.length)`
(b) Q *where did kevin love go to college?*
R `(JOIN education.institution (JOIN person.education love))`
G `(AND (JOIN topic.notable_types college) (JOIN edu.institution (JOIN person.education love)))`
Ranking Better Than Generation
(c) Q what song for tv or television did benny davis compose?
R `(AND tv.tv_song (JOIN composition.lyricist davis))`
G `(AND tv.tv_song (JOIN composition.song (JOIN composition.composer davis)))`
(d) Q what team does heskey play for?
R `(JOIN sports_team_roster.team (JOIN pro_athlete.teams heskey))`
G `(JOIN sports_team_roster.team (AND (JOIN sports_team_roster.from 2015) (JOIN pro_athlete.teams heskey)))`

Figure 7: Examples of outputs from the generator (G) and ranker (R). A generation step is able to compensate some missing operators not supported in the enumeration (a), or patch some implicit clue (b). However, generator does introduce errors if it produces another prediction when there is inherent ambiguity in the question and the top-ranked one is indeed correct (c). Generator also adds unnecessary constraint sometimes (d).

ing the fractions of questions where ∘ top left: the top ranking prediction and top generation prediction achieves a equal nonzero F1, ∘ top right: the top generation prediction is better, ∘ bottom left: the top ranking prediction is better, ∘ bottom right: they both fail (achieving a 0 F1).

The generator retains the ranking predictions without any modifications for most of the time. For 4.7% and 8.9% of the questions from GRAILQA and WEBQSP, respectively, the generator is able to fix the top-ranked candidates and improves the performance. Although generator can make mistakes in non-negligible fraction of examples on WEBQSP, it is mostly caused by introducing false constraints (e.g., Figure 7 (d)). Thanks to our execution-guided inference procedure, we can still turn back to ranker-predicted results when the generator makes mistakes, which allows tolerating generation errors to some extent.

We also show the break down by types of generalization on GRAILQA (bottom row in Figure 6). Generation stage is more helpful in i.i.d. and compositional setting, but less effective in zero-shot

	GRAILQA		WEBQSP	
	EXEC	VALID	EXEC	VALID
Top-1	99.7	88.1	98.7	91.1
Top-3	99.7	89.4	99.5	94.5
Top-5	99.7	89.8	99.5	94.6
Top-10	99.7	90.4	99.5	95.4

Table 4: The chances of finding an executable (EXEC) and a valid (VALID) logical form among the top-k generated. logical forms.

setting, as it involves unseen relations that are usually hard to generate.

Executability We use executability to further measure the quality of generated outputs. Table 4 shows executable rate (producing an executable logical forms) and valid rate (producing a logical form that yields non-empty answer) among the top-k decoded list. Nearly all the top-1 logical forms are executable. This suggests that the generation model can indeed produce high-quality predictions in terms of syntactic correctness and consistency with KB. As the beam size increases, more valid logical forms can be found in the top-k list, which our inference procedure can benefit from.

Output Examples of Ranking Model and Generation Model For more intuitive understanding of how the generator works, we attach several concrete examples (Figure 7). As suggested by example (a), the generation model can remedy some missing operations (*ARGMIN*) not supported when enumerating. It can also patch the top-ranked candidate with implicit constraints: the `(JOIN topic.notable_types college)` in (b) is not explicitly stated, and our NER system fails to recognize *college* as an entity.

As in example (c), the generation model makes a worse prediction sometimes because it prefers another prediction in the top-ranked list due to inherent ambiguity in the question. It can also fail when falsely adding a constraint which results in empty answer (d).

4 Related Work

KBQA is a promising technique for users to efficiently query over large KB, which has been extensively studied over the last decade. Past work has collected a series of datasets (Yih et al., 2016; Bordes et al., 2015; Zhang et al., 2018; Su et al., 2016; Gu et al., 2021) as well as proposed a diversity of approaches for this task.

One line of KBQA approaches first constructs a query-specific subgraph with information retrieved from the KB and then rank entity nodes to select top entities as the answer (Sun et al., 2018, 2019; Saxena et al., 2020; Cohen et al., 2020; Shi et al., 2021). The subgraph can either be retrieved in one-shot using heuristic rules (Sun et al., 2018), or iteratively built using learned models (Sun et al., 2019; Shi et al., 2021; Cohen et al., 2020; Saxena et al., 2020). Later, a neural model operating over subgraph is employed to determine the answer nodes (Sun et al., 2018, 2019; Shi et al., 2021). Such information retrieval based approaches are usually less interpretable as they do not produce the inference path reaching the answer, whereas our approach is more transparent since we are able to produce logical forms.

More closely related to our approach, another line answers a question by parsing it into an executable logical form in various representations, including lambda-DCS (Liang, 2013; Berant et al., 2013), sparql query (Das et al., 2021), graph query (Yih et al., 2015; Su et al., 2016; Lan and Jiang, 2020), and s-expression (Gu et al., 2021). Past work has attempted to generate logical forms using grammar-based parsera (Berant et al., 2013) or seq-to-seq parsers (Zhang et al., 2019). There has also been an alternative way that first enumerates a list of logical form candidates and then choose one that best matches the intents in the question (Lan and Jiang, 2020; Luo et al., 2018; Yih et al., 2015; Yavuz et al., 2016, 2017; Reddy et al., 2017; Sun et al., 2020). Our approach differs in that we employ a generation stage to remedy the coverage issue which these approaches often suffer from.

5 Conclusion

We have presented RNG-KBQA for question answering over knowledge base. RNG-KBQA consists of a ranking step and a generation step. Our ranker trained with iterative bootstrapping strategy can better distinguish correct logical forms from spurious ones than prior seq-to-seq ranker. Our generator can further remedy uncovered operations or implicitly mentioned constraints in the top-ranked logical forms. The experimental results on two datasets, GRAILQA and WEBQSP, suggest the strong performance of our approach: RNG-KBQA achieves new state-of-the-art performance on both datasets, and particularly outperforms prior methods in generalization setting by a large margin.

References

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. UHop: An unrestricted-hop relation extraction framework for knowledge-based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

William W. Cohen, Haitian Sun, R. Alex Hofer, and Matthew Siegler. 2020. Scalable neural methods for reasoning with a symbolic knowledge base. In *International Conference on Learning Representations*.

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. *arXiv preprint arXiv:2104.08762*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. 2017. Robustfill: Neural Program Learning under Noisy I/O. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of clueweb corpora, version 1.

Yu Gu, Sue E. Kase, M. 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.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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*.

Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019. Knowledge base question answering with topic units. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5046–5052. International Joint Conferences on Artificial Intelligence Organization.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *EMNLP*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

P. Liang. 2013. Lambda dependency-based compositional semantics. *Technical Report, ArXiv*.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Ré. 2021. Bootleg: Chasing the tail with self-supervised named entity disambiguation. *ArXiv*, abs/2010.10363.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

9

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark.

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

Jiaxin Shi, Shulin Cao, Lei Hou, and Hanwang Li, Juanziand Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. In *EMNLP*.

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*.

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

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. SPARQA: skeleton-based semantic parsing for complex questions over knowledge bases. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

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

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2017. Recovering question answering errors via query revision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2020a. Benchmarking multimodal regex synthesis with complex structures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xi Ye, Qiaochu Chen, Xinyu Wang, Isil Dillig, and Greg Durrett. 2020b. Sketch-Driven Regular Expression Generation from Natural Language and Examples. In *Transactions of the Association for Computational Linguistics (TACL)*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

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)*.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019. Complex question decomposition for semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

10

	Linking F1	KBQA F1
Bert Ranking (Gu et al., 2021)	72.2	58.0
ReTrack (Chen et al., 2021)	77.4	65.3
RnG-KBQA (Ours)	79.6	74.4
w/o Entity Disambiguation	72.2	67.4

Table 5: The entity linking F1 (on dev set) and the corresponding final F1 scores (on leaderboard) on GRAILQA of various methods.

A Details of Entity Linking Performance

Table 5 shows the entity linking performance and KBQA performance on GRAILQA of various methods. Compared to the popularity-based baseline (Bert Ranking), Our entity disambiguation model is effective and successfully improves the entity linking F1 by 7.4, which boosts the final KBQA F1 score by 7.0. Our entity linking model is also better than the Bootleg approach (Orr et al., 2021) used in ReTrack (Chen et al., 2021).

Furthermore, our method without the entity disambiguation modules outperforms Bert Ranking with a substantially large margin (11.4 F1 score). Our method even beats ReTrack when it is built upon a much better entity linking model. The results suggest the strong effectiveness of our rank-and-generate framework.