RetinaQA: A Robust Knowledge Base Question Answering Model for both Answerable and Unanswerable Questions

Prayushi Faldu[†], **Indrajit Bhattacharya**[‡], **Mausam**[†]
[†]Indian Institute of Technology Delhi, [‡]TCS Research
prayushifaldu123@gmail.com, b.indrajit@tcs.com, mausam@cse.iitd.ac.in

Abstract

An essential requirement for a real-world Knowledge Base Question Answering (KBQA) system is the ability to detect answerability of questions when generating logical forms. However, state-of-the-art KBQA models assume all questions to be answerable. Recent research has found that such models, when superficially adapted to detect answerability, struggle to satisfactorily identify the different categories of unanswerable questions, and simultaneously preserve good performance for answerable questions. Towards addressing this issue, we propose RetinaQA, a new KBQA model that unifies two key ideas in a single KBQA architecture: (a) discrimination over candidate logical forms, rather than generating these, for handling schema-related unanswerability, and (b) sketch-filling-based construction of candidate logical forms for handling data-related unaswerability. Our results show that RetinaQA significantly outperforms adaptations of state-of-the-art KBQA models in handling both answerable and unanswerable questions and demonstrates robustness across all categories of unanswerability. Notably, RetinaQA also sets a new state-of-the-art for answerable KBQA, surpassing existing models. We release our code-base[1] for further research.

1 Introduction

Question answering over knowledge bases (KBQA) (Saxena et al., 2022; Zhang et al., 2022; Mitra et al., 2022; Wang et al., 2022; Das et al., 2022; Cao et al., 2022; Ye et al., 2022; Chen et al., 2021; Das et al., 2021; Shu et al., 2022; Gu et al., 2023) requires answering natural language questions over a knowledge base (KB), most commonly via generating formal queries or logical forms that are then executed over the knowledge base to retrieve the answers. When users interact with KBs in real-world settings, unanswerability of questions arises naturally. Users are typically unfamiliar with the schema and data of the underlying KB. Further, KBs are also often incomplete. While specialized models for handling unanswerability have been proposed for other question answering tasks (Rajpurkar et al., 2018; Choi et al., 2018; Reddy et al., 2019; Sulem et al., 2022; Raina and Gales, 2022), all existing models for KBQA assume answerability of questions over the given KB.

Recently, Patidar et al. (2023) published a benchmark dataset called GrailQAbility incorporating different categories of unanswerable questions. This work also demonstrated that state-of-the-art KBQA models perform poorly off-the-shelf for unanswerable questions. This performance improves with superficial adaptations for unaswerability, such as adding unanswerable questions during training and thresholding. However, such adaptations significantly hurt performance for answerable questions. Additionally, different state-of-the-art models struggle with different categories of unaswerability, such as (a) questions for which schema elements (i.e. relations or entity types) are missing in the KB, and which therefore do not have valid logical forms, and (b) questions for which data elements (i.e. entities or facts) are missing in the KB, and which therefore have logical forms that are valid, but return empty answers on execution. This work demonstrated that there is no single model which performs well for all categories of unanswerable questions and answerable ones.

Our analysis of existing models reveals two key drawbacks that make KBQA models less robust – imperfect calibration and over-reliance on path-based retrieval. To overcome these, we propose a new multi-stage **Retri**eve, ge**n**erate, and r**a**nk architecture for KBQA named **RetinaQA**, which brings together several ideas from the literature.

First, a robust KBQA model has to separate questions that are answerable (having a valid logical form) from those that are unanswerable due

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

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6643–6656
August 11-16, 2024 ©2024 Association for Computational Linguistics

to missing schema elements (not having a valid logical form). This requires good model calibration. Most KBQA models use *generation* of logical form as their final step, which has calibration challanges. Instead, since discriminative models are generally better calibrated, RetinaQA's final stage *discriminates* among candidate logical forms, better separating questions without any logical form.

Secondly, existing models assume answerability, where all logical forms have accompanying paths in the KB. These models learn to rely on retrieved paths from the KB to generate logical forms. However, missing data elements may break any valid path, while the question, even though unanswerable, still has a valid logical form. Existing models falter here, and end up generating some other logical form corresponding to paths that do exist in the KB. To handle such unanswerable questions, RetinaQA additionally employs sketch-filling-based construction. It first generates schema-independent high-level sketches, and then grounds these for the specific KB using relevant schema elements retrieved for the question.

Using experiments over GrailQAbility, we demonstrate that the performance of RetinaQA is not only stable but also significantly better than that of adaptations of multiple state-of-the-art KBQA models that assume answerability, not only across different categories of unanswerable questions, but also for answerable ones. Interestingly, we demonstrate that the RetinaQA architecture performs strongly for fully answerable KBQA benchmarks as well, and establishes a new state-of-the-art performance on the GrailQA dataset.

2 Related Work

The predominant approach for supervised KBQA uses the question to construct a logical form, which is then executed to retrieve the answer (Cao et al., 2022; Ye et al., 2022; Chen et al., 2021; Das et al., 2021). State-of-the-art models use k-hop path traversal to retrieve data paths from linked entities in the question (Ye et al., 2022; Shu et al., 2022). Some models, instead (Chen et al., 2021) or additionally (Shu et al., 2022), retrieve schema elements (namely, entity types and relations) based on the question. These utilize transformer-based decoders to generate the target logical forms. We have observed that generative models are not adequately calibrated to separate answerable and unanswerable questions correctly.

In contrast, Pangu (Gu et al., 2023) uses language models to incrementally enumerate and discriminate between partial logical forms. Though this approach is better calibrated, its path-level enumeration makes it brittle for questions with missing data elements. Some retrieval-based methods (Saxena et al., 2020, 2022) also perform ranking of answer paths, but directly select answer nodes. These methods maximize the similarity score between a relation and the question. Adapting such techniques to detect unanswerable questions is difficult. In contrast, we perform contrastive-learning-based one-shot discrimination on fully-formed logical form candidates in the final stage.

In addition to supervised in-domain settings, transfer (Cao et al., 2022; Ravishankar et al., 2022) and few-shot (Li et al., 2023) settings have also been studied for KBQA. Here, test questions involve unseen KB relations and entity types. Approaches for these settings first generate high-level sketches (also called drafts or skeletons) that capture the syntax of the target language. These sketches are then filled in with KB-specific arguments to construct complete programs, which are finally scored and ranked. Notably, these transfer architectures do not involve any traversal-based component to retrieve logical forms. In contemporaneous work on few-shot transfer, Patidar et al. (2024) propose the FuSIC-KBQA framework accommodating one or more supervised retrievers coupled with an LLM for retrieval reranking and logical form generation. This demonstrates how retrievers from existing supervised models (Shu et al., 2022; Gu et al., 2023) can be adapted and augmented using LLMs for low-supervision settings. In principle, RetinaQA can also be accommodated as a retriever in this framework.

Unanswerability and specialized models for detecting unanswerable questions have been studied for many question-answering tasks (Rajpurkar et al., 2018; Choi et al., 2018; Reddy et al., 2019; Sulem et al., 2022; Raina and Gales, 2022). However, no specialized models have been proposed for detecting unanswerable questions in KBQA. All existing KBQA models assume that questions have valid logical forms with non-empty answers. Even in the transfer setting for KBQA (Cao et al., 2022; Ravishankar et al., 2022), though the target logical forms may involve schema elements unseen during training, questions are still assumed to be answerable. Recent work (Patidar et al., 2023)

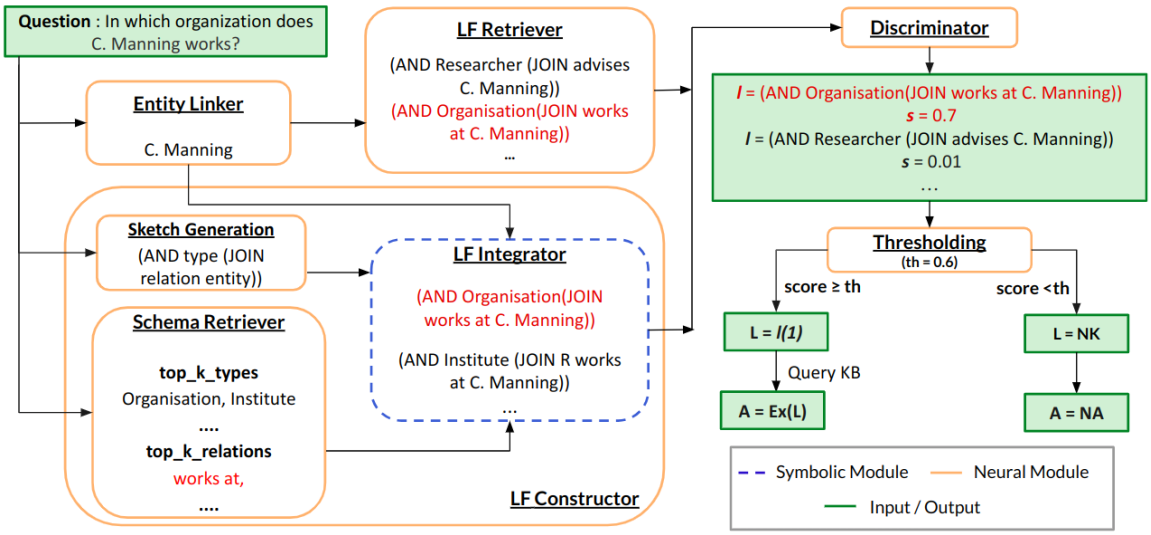

Figure 1: RetinaQA Architecture showing different components illustrated with an example question. Symbols l and s represent candidate logical form and its score as Discriminator output, L the output logical form, A the final answer, $l(1)$ the top ranked logical form, Ex(l) the answer obtained by executing logical form l. NK and NA are special symbols indicating No Knowledge (for logical form) and No Answer. The logical form in red under LF Retriever would not be found if data element *(C. Manning, works at, Stanford)* is missing in the KB, and additionally that in red under LF Integrator would not be found if relation *works at* is missing in KB schema and therefore not retrieved by the Schema Retriever. The candidate logical form in red under Discriminator would not be found if both of these are missing.

has published the GrailQAbility benchmark by modifying the popular GrailQA dataset (Gu et al., 2021) to incorporate multiple categories of unanswerable questions. This work also demonstrates that answerable-only KBQA models, when superficially adapted for handling unanswerability, fall short in many ways.

3 The Problem and Our Solution

We first briefly define the KBQA with unanswerability task, and then describe the architecture of our proposed RetinaQA model.

3.1 KBQA with Unanswerability

A Knowledge Base G consists of a schema S with data D stored under it. The schema consists of entity types T and binary relations R defined over pairs of types. Together, we refer to these as schema elements. The data D consists of entities E as instances of types T, and facts $F \subseteq E \times R \times E$. Together, we refer to these as data elements. We follow the definition of (Patidar et al., 2023) for defining the task of Knowledge Base Question Answering (KBQA) with unanswerability. A natural language question q is said to be answerable for a KB G if it has a corresponding logical form l, which, when executed over G, returns a non-empty answer A. In contrast, a question q is unanswerable for G, if it either (a) does not have a corresponding logical form that is valid for G, or (b) it has a valid logical form l for G, but which on executing returns an empty answer. The first case indicates that G is missing some schema element necessary for capturing the semantics for q. The second case indicates that the schema S is sufficient for q, but G is missing some necessary data elements for answering it. In the KBQA with unanswerability task, given a question q, if it is answerable, the model needs to output the corresponding logical form l and the non-empty answer A entailed by it, and if it is unanswerable, the model either needs to output NK (meaning No Knowledge) for the logical form, or a valid logical form l with NA (meaning No Answer) as the answer. While different formalisms have been proposed for logical forms, we use *s-expressions* (Gu et al., 2021). These have set-based semantics, functions with arguments and return values as sets.

3.2 The RetinaQA Model

Fig. 1 shows the architecture of RetinaQA. At a high level, RetinaQA has two stages – **logical form enumeration**, followed by **logical form ranking**. For logical form enumeration, RetinaQA follows

two complementary approaches and then takes the union. The first is **path-traversal based retrieval**. Starting from linked entities in the question, RetinaQA traverses *data-level KB paths* and transforms these to logical forms. The second is **sketch-filling based construction**, which is critical when the KB has missing data elements for the question. Here, RetinaQA first *generates* **logical form sketches** corresponding to the question, and then *enumerates* semantically valid groundings for these by *retrieving* relevant KB schema elements for filling in the sketch arguments. Note that this approach utilizes only the KB schema and avoids data. Once candidate logical forms are so identified, RetinaQA uses *discriminative* scoring to rank these logical forms with respect to the question. We next explain each of these components in more detail.

Entity Linker: The pipeline starts with linking mentioned entities in the question with KB entities E. This is required for both logical form retrieval and logical form construction. We use an off-the-shelf entity linker (Ye et al., 2022) previously used in the KBQA literature (Shu et al., 2022; Gu et al., 2023). More details are in the Appendix (A.1). If the mentioned entities are missing in the KB, the entity linker returns an empty set.

Logical Form Retriever: As the first approach for enumerating logical forms, RetinaQA uses KB data path traversal (Ye et al., 2022). RetinaQA traverses 2-hop paths starting from the linked entities and transforms these to logical forms in s-expression. These logical forms are then scored according to their similarity with the question, and the top-10 logical forms are selected for the next stage, as illustrated under LF Retriever in Fig. 1. Following (Ye et al., 2022), we score a logical form l and question q as:

$$s(l, q) = \textsc{Linear}(\textsc{BertCls}([l; q])) \quad (1)$$

and optimize a contrastive objective for ranking:

$$\mathcal{L}_{ret} = -\frac{\exp(s(l^*, q))}{\exp(s(l^*, q)) + \sum_{l \in L \wedge l \neq l^*} \exp(s(l, q))} \quad (2)$$

where l^* is the gold-standard logical form for q, and L is the set of logical forms similar to l^*. Note that the transformation to logical forms from KB-paths only covers certain operators (such as *count*), but not some others (such as *argmin*, *argmax*), so that this enumeration approach is not guaranteed to cover all logical forms. More importantly for unanswerability, as illustrated in Fig. 1, this approach cannot retrieve the logical form in red when the relevant data path in the KB is broken, as by the absence of the data element *(C. Manning, works at, Stanford)* in our example.

Logical Form Constructor: The second approach used by RetinaQA for logical form enumeration is sketch-filling. Drawing inspiration from the transfer approaches for KBQA (Cao et al., 2022; Ravishankar et al., 2022; Li et al., 2023), RetinaQA uses logical form sketches. Sketches capture KB-independent syntax of s-expressions with functions, operators and literals, and replace KB-specific elements, specifically entities, entity types and relations, with arguments. RetinaQA first generates sketches using a **sketch generator**, and in parallel retrieves relevant schema elements as candidates for sketch arguments using a **schema retriever**, and finally fills in arguments for each candidate sketch using the retrieved argument candidates in all possible valid ways using a **logical form integrator**. By avoiding path-based retrieval, this approach can construct valid logical forms when these exist, even when some relevant data element for the question is missing in the KB, for example, when the data element *(C. Manning, works at, Stanford)* is missing in the KB but the relation *works at* is present in the KB schema.

Sketch Generator: The sketch generator takes the question q as input and outputs a sketch s, optimizing a cross-entropy-based objective:

$$L_{sketch} = -\sum_{t=1}^{n} \log(p(s_t | s_{<t}, q))$$

Specifically, we fine-tune T5 (Raffel et al., 2020) as the Seq2Seq model. We also perform constrained decoding during inference to ensure syntactic correctness of the generated sketch. This step is unaffected by any KB incompleteness.

Schema Retriever: To retrieve candidate arguments for generated sketches, we follow the schema retriever of TIARA (Shu et al., 2022). It is a cross encoder and uses the form of Eqn.1 to score a schema element x and the question q, and optimizes the same form of the objective as for the sentence-pair classification task (Devlin et al., 2019). We train two retriever models, one for relations and one for types, and use the top-10 types and top-10 relations as candidate arguments for

each question. As illustrated in Fig. 1, this step breaks when relevant relations, such as *works at*, or entity types are missing from the KB schema.

Logical Form Integrator: This component grounds each generated candidate sketch using the retrieved candidate arguments and the linked entities to construct complete logical form candidates. Each candidate sketch is grounded using every possible combination of candidate arguments. A symbolic checker ensures type-level validity of the grounded logical forms for the KB G. This also avoids a combinatorial blow-up and restricts the space of logical form candidates. This component does not involve any trainable parameters.

Logical Form Discriminator: This component finally scores and ranks all the logical form candidates provided by the retriever and constructor components. A T5 encoder-decoder model is trained to score a logical form candidate. Following Zhuang et al. (2022), we feed a (question, logical form) pair to the encoder, and use decoding probability for a special token as the ranking score.[2] This component uses a contrastive learning-based optimization objective similar to Eqn.2. We sample negative examples randomly, but this mostly covers the set of all negative candidates given its small size.

For a test question, the candidate logical forms are ranked according to the predicted discriminator scores. If the score of the top-ranked candidate is below a threshold (tuned on the validation set), it is classified as unanswerable i.e. $L = \text{NK}$ and $A = \text{NA}$. This helps in identifying questions for which valid logical forms do not exist due to missing schema elements. For example, in Fig. 1, if the logical form in red is missing from the candidate list, the discriminator assigns a low score to the rank 1 logical form candidate, and NK is output after thresholding. Otherwise, the top-ranked candidate is predicted as the logical form.

The predicted logical form is then converted to SPARQL and executed over KB. If a non-empty answer is obtained, then the question is considered answerable with the output as the desired answer. On the other hand, if the answer is empty, the question is classified as unanswerable under the missing data elements category, i.e., $A = \text{NA}$.

[2]We use $< extra\_id\_6 >$ token of T5 for tuning the ranking score.

4 Experiments

We address the following research questions: **(1)** How does RetinaQA compare against baselines for answerable and unanswerable questions in two different training settings: one with only answerable questions and another with both answerable and unanswerable questions? **(2)** How does RetinaQA perform for different categories of unanswerable questions? **(3)** How does RetinaQA perform in the answerable-only setting? **(4)** To what extent do the different components of RetinaQA contribute towards its performance in (1), (2) and (3) above?

4.1 Experimental Setup

Datasets: For research questions (1) and (2) above, we use the GrailQAbility dataset, which is the only KBQA dataset that contains both answerable and unanswerable questions. For research question (3), we use the two most popular KBQA datasets with only answerable questions, namely GrailQA and WebQSP.

GrailQA (Gu et al., 2021) is a popular KBQA dataset that contains only answerable questions. It contains questions at various levels of generalization: IID (seen schema elements), compositional (unseen combination of seen schema elements) and zero-shot (unseen schema elements). **WebQSP** (Yih et al., 2016) also has only answerable questions with Freebase as the KB, but unlike GrailQA, where the questions are synthetically constructed, it contains real user queries annotated with logical forms. It only has IID test questions. **GrailQAbility** (Patidar et al., 2023) is a recent dataset that adapts GrailQA to additionally incorporate unanswerable questions. The unanswerable questions are constructed by systematically dropping data and schema elements from the KB. More details are added in the Appendix (A.4).

Evaluation Metrics: We primarily focus on evaluating the logical form using the Exact Match (EM) metric, which checks whether the predicted logical form is same as the gold logical form (which is NK for unanswerable questions with missing schema elements). We also evaluate the answers using the F1 score, which compares the predicted answer set with the gold answer set. For unanswerable questions, similar to prior work (Patidar et al., 2023), we report two F1 scores – the regular score F1(R) compares the list of answers based on the given incomplete KB, and the lenient score F1(L) which does not penalize a model for returning ideal

Train	Model	Overall			Answerable			Unanswerable		
		F1(L)	F1(R)	EM	F1(L)	F1(R)	EM	F1(L)	F1(R)	EM
A	RnG-KBQA	67.80	65.60	51.60	78.10	78.10	74.20	46.90	40.10	5.70
	RnG-KBQA + T	67.60	65.80	57.00	71.40	71.30	68.50	59.90	54.50	33.60
	Tiara	75.05	72.84	53.69	80.03	80.00	75.63	64.95	58.31	9.20
	Tiara + T	73.26	71.62	55.23	74.08	74.05	70.56	71.60	66.68	24.15
	Pangu	63.09	60.06	54.55	78.72	78.7	74.00	31.40	22.25	15.13
	Pangu + T	79.14	77.89	66.53	75.52	75.51	72.37	86.48	82.70	54.68
	RetinaQA	76.83	75.24	64.54	**81.22**	**81.2**	**77.41**	67.93	63.16	38.45
	RetinaQA + T	**83.30**	**82.18**	**73.76**	81.19	81.17	75.01	**87.59**	**84.22**	**71.20**
A+U	RnG-KBQA	80.50	79.40	68.20	75.90	75.90	72.60	89.70	86.40	59.40
	RnG-KBQA + T	77.80	77.10	67.80	70.90	70.80	68.10	92.00	89.80	67.20
	Tiara	78.29	77.43	66.29	71.33	71.32	68.29	92.4	89.82	62.24
	Tiara + T	77.67	76.94	66.87	69.89	69.88	66.98	93.43	91.24	66.65
	Pangu	63.59	60.42	53.79	79.45	79.42	73.49	31.42	21.89	13.85
	Pangu + T	78.29	76.91	66.14	75.25	75.22	71.62	80.46	80.32	55.03
	RetinaQA	77.31	75.71	64.79	**80.98**	**80.97**	**76.95**	69.87	65.04	40.14
	RetinaQA + T	**83.30**	**82.69**	**77.45**	77.91	77.91	75.16	**94.21**	**92.38**	**82.10**
	RetinaQA - LFR + T	77.36	76.37	65.37	73.40	73.39	70.90	85.38	82.43	54.17
	RetinaQA - LFI + T	74.89	73.53	53.89	70.89	70.85	68.07	83.01	78.95	25.13
	RetinaQA - (SG ∪ SR) + T	64.68	62.58	52.46	72.99	72.95	68.13	47.84	41.54	20.70

Table 1: Performance of different models on the GrailQAbility dataset: overall and for answerable and unanswerable questions. A indicates training with answerable questions, A+U with answerable and unanswerable questions, +T indicates thresholding. Ablations of RetinaQA are named as RetinaQA - X, where we denote logical form retriever as LFR, logical form integrator as LFI and sketch generator and schema retriever together as (SG ∪ SR).

Train	Model	Schema Element Missing						Data Element Missing			
		Type		Relation		Mention Entity		Other Entity		Fact	
		F1(R)	EM	F1(R)	EM	F1(R)	EM	F1(R)	EM	F1(R)	EM
A	RnG-KBQA + T	55.50	49.50	57.10	46.60	44.70	40.30	56.00	11.50	58.60	13.90
	Tiara + T	66.27	21.70	70.21	28.06	61.01	23.43	68.91	22.97	68.29	23.63
	Pangu + T	**87.97**	**87.50**	**80.07**	**79.63**	90.57	**90.41**	83.19	0.00	76.48	1.07
	RetinaQA + T	86.32	80.31	79.41	62.08	**90.72**	77.83	**85.71**	**68.07**	**84.68**	**71.14**
A+U	RnG-KBQA + T	93.40	86.80	89.70	**85.50**	92.10	89.60	87.10	30.80	86.00	32.50
	Tiara + T	91.63	83.84	**90.90**	72.37	**94.50**	71.38	91.60	50.42	90.38	52.85
	Pangu + T	90.80	**90.68**	78.66	78.44	90.41	**90.25**	74.51	0.00	69.71	0.95
	RetinaQA + T	**94.22**	90.21	88.52	81.91	94.34	86.64	**93.84**	**75.91**	**94.30**	**76.13**

Table 2: Performance of different models for the unanswerable questions in GrailQAbility, grouped by categories of KB incompleteness. Note that missing mention entities result in invalid logical form, while other missing entities lead to valid logical form with no answer.

answers. Specifically, F1(L) accepts answers considering both KBs – the ideal or complete KB and new or incomplete KB. In a way, it evaluates the model's ability to infer missing KB elements and predict the correct answer.

Baselines: We compare RetinaQA against existing state-of-the-art KBQA models, as per the GrailQA leaderboard and code availability. These are TIARA (Shu et al., 2022), RnG-KBQA (Ye et al., 2022), and Pangu (Gu et al., 2023). Of these, the first two are shown to the best performing models on GrailQAbility, and Pangu is a SoTA model for GrailQA[3] and WebQSP (Gu et al., 2023). For fair comparison, we use the same entity linker (Ye et al., 2022) and T5-base as base LLM for all models. For GrailQAbility, we adapt all models appropriately for unanswerability. Specifically, we perform thresholding (denoted as "+ T") on logical form generation to output NK. The thresholds are tuned on the dev set. Additionally, instead of training the models only on answerable questions (denoted as A training), we train the models using both answerable and unanswerable questions (denoted as A+U training). Further implementation details are in the Appendix (A.2).

4.2 Results for KBQA with Unanswerability

We first address research question (1). In Table 1, we report high-level performance of different models on GrailQAbility. With A+U training, RetinaQA+T outperforms all models overall and is about 9 pct points ahead of the closest competi-

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

Train	Model	IID			Compositional			Zero-Shot		
		F1(L)	F1(R)	EM	F1(L)	F1(R)	EM	F1(L)	F1(R)	EM
A	RnG-KBQA	85.50	85.40	83.20	65.90	65.90	60.20	72.70	72.70	67.30
	TIARA	86.53	86.47	84.52	72.02	72.02	64.93	74.24	74.24	67.60
	Pangu	82.00	81.97	79.09	71.63	71.63	65.95	**77.02**	**77.02**	**70.18**
	RetinaQA	**87.94**	**87.90**	**85.85**	**73.92**	**73.92**	**67.48**	74.84	74.84	69.68
A+U	RnG-KBQA	85.40	85.30	83.30	65.80	65.80	60.80	66.90	66.90	62.60
	TIARA	82.38	82.36	80.57	65.16	65.16	59.84	58.50	58.50	54.65
	Pangu	81.08	81.01	76.85	**77.43**	**77.43**	**69.52**	**78.01**	**78.01**	**70.42**
	RetinaQA	**89.00**	**88.98**	**87.06**	71.69	71.69	65.55	73.59	73.59	67.51

Table 3: Performance of different models for answerable questions in the GrailQAbility dataset, for IID, compositional, and zero-shot test scenarios. Names have the same meanings as in Table 1.

tor (Pangu+T) in terms of EM. For unanswerable questions, RetinaQA achieves a 16 pct points improvement, while being consistently better for answerable questions. Unsurprisingly, thresholding helps all models for unanswerable questions and hurts slightly for answerable ones. This drop is relatively small for Pangu and RetinaQA, suggesting that these are better calibrated due to their discriminative training.

Next, we address research question (2). Table 2 drills down on performance for different categories of unanswerability. First, we observe that for the baselines, performance varies significantly across different categories. Pangu performs well for missing schema elements but is the worst model for missing data elements. TIARA is the best baseline for missing data elements but does not perform as well for missing schema elements. We further analyze such variability in performance for the baselines in the Appendix (Sec. A.3.1). RetinaQA performs the best by a large margin for questions with missing data elements, and compares favorably with Pangu for missing schema elements, making it the overall model of choice across different categories of unanswerability. We also note that RetinaQA+T results shows little degradation for questions with missing data (which have valid logical forms), and huge gains for questions with missing schema elements.

In the A training setting, as a testimony to its robustness, RetinaQA achieves comparable performance for answerable and unanswerable questions, with an EM gap of only 4 pct points. In contrast, this gap ranges 18 to 45 pct points for other models. Other trends are very similar to the A+U setting.

Additionally, we see that RetinaQA largely outperforms existing models across different generalization settings for answerable questions (Table 3). RetinaQA performs the best for IID and compositional generalization, but for zero-shot generalization, RetinaQA performs slightly worse than Pangu. This is mainly because of the trade-off related to path traversal, as we explain in Section 4.4.

For unanswerable questions (Table 4), RetinaQA outperforms all existing models for both IID and zero-shot generalization. Furthermore, we observe that the difference between EM and F1(R) scores is the smallest by far for RetinaQA, indicating that it not only predicts unanswerability correctly but also does so for the right reason.

Model	IID		Zero-Shot	
	F1(R)	EM	F1(R)	EM
RnG-KBQA + T	94.30	75.90	85.90	59.50
TIARA + T	95.10	77.77	87.86	56.88
Pangu + T	80.51	57.52	80.15	52.85
RetinaQA + T	**97.01**	**89.94**	**88.31**	**75.22**

Table 4: Performance of different models for IID and zero-shot test scenarios for unanswerable questions in GrailQAbility for A+U training. Names have the same meanings as in Table 1.

Additionally, In Table 5, we further scrutinize the performance of the models for sub-categories of zero-shot unanswerable questions. We observe that RnG-KBQA and Pangu perform better in terms of both EM and F1(R) for these categories. However, this is a natural benefit of mostly predicting $L = NK$ for unanswerable questions, which results in extremely poor performance for the missing data element category, as we have already seen in Table 2. As seen earlier, here too all the baseline models have huge variations in performance across full z-shot and partial z-shot categories. This is in contrast to RetinaQA, which achieves comparable performance. It does so without compromising on performance for missing data elements category. This further establishes the stability of RetinaQA across various generalization categories and makes it more reliable for real-world scenarios.

Model	Full Z-Shot		Partial Z-Shot	
	F1(R)	EM	F1(R)	EM
RnG-KBQA + T	89.70	86.70	**83.10**	71.00
TIARA + T	**90.64**	78.82	82.64	54.14
Pangu + T	89.66	**89.66**	79.94	**79.46**
RetinaQA + T	88.67	77.83	80.89	70.54

Table 5: Performance of different models for missing schema elements - partial zero-shot and full-zero test scenarios in GrailQAbility for A+U training. Names have the same meanings as in Table 1.

4.3 Results for Answerable-only KBQA

We designed RetinaQA for stability across answerable and unanswerable questions, and this naturally trades off performance across the two categories. However, we observed that RetinaQA performed the best for answerable questions as well in GrailQAbility. Motivated by this, we next address research question (3), by evaluating it for traditional KBQA benchmarks with only answerable questions. Since all questions are answerable in this setting, we (re-)introduce Execution Guided Check (EGC) as the final step for all models including RetinaQA. With EGC, models output the highest-ranked logical form which when executed over the KB returns a non-empty answer.

In Table 6, we report results on GrailQA. We find that overall RetinaQA beats previous the state-of-the art by around 1.2 pct points in terms of F1 and 1.8 pct points in terms of EM, establishing a new state-of-the-art for this dataset. We also see that, as for answerable questions in GrailQAbility, here too RetinaQA performs the best for IID and compositional generalization, and performs almost at par with Pangu for zero-shot generalization.

Further analysis shows that RetinaQA performs well across questions of various complexities. It is the best model for 1, 2, and 4-hop questions, while it is outperformed by TIARA for 3-hop questions. More details are in the Appendix (A.3.2).

In Table 7, we record results for WebQSP. Here, RetinaQA outperforms Pangu by 0.6 pct points but ranks below TIARA by 0.2 pct points. These numbers further establish the usefulness of its architecture for answerable-only KBQA as well.

4.4 Ablation Study

Finally, we address the research question (4). Here, we assess the individual contributions of the different components in RetinaQA. First, we remove (one at a time) the three key components: the logical form integrator (LFI), the logical form retriever (LFR), and the coupled sketch generator (SG) and schema retriever (SR).

Our ablation study focuses on specific question categories. First, we study the recall of the correct logical form within the candidate set for unanswerable questions for missing data elements (see Table 12 in Appendix). We see that if we remove SR and SG, there is a significant 65 pct point decrease in recall. In contrast, removing LFR has little impact. This is because LFR relies on path traversal to retrieve candidate logical forms This fails when paths are incomplete, preventing RetinaQA from effectively enumerating logical forms. This agrees with our intuition that when essential data elements are missing, we need retrieval techniques that are independent of path traversal.

Next, we do a similar study of recall for answerable questions (Table 11 in Appendix). We observe that eliminating LFR leads to a significant reduction in recall for zero-shot questions. This is unlike removing SG and SR, which only affects i.i.d. questions. This is because retrieval techniques that do not rely on path traversal have a larger search space, resulting in more errors when encountering unseen schema elements. In contrast, path traversal-based methods are grounded at the fact level. This results in a smaller search space and therefore fewer errors for zero-shot questions. Thus, path-traversal-dependent logical form retrieval is important for zero-shot generalization of answerable questions.

Finally, we evaluate the impact of LFI and EGC. Both reduce the space of logical form candidates for the discriminator, by pruning out invalid logical forms. Table 10 in the Appendix records performance of ablations of RetinaQA in the answerable setting on GrailQA. By switching off LFI and EGC separately, we see drops in performance by about 4 pct points and 2 pct points respectively. However, on switching off both together, we observe a 17 pct point drop. This shows that these components can compensate for each other, but RetinaQA needs at least one of them for good performance.

Additional ablations show that SG and SR are more helpful for complex (3-hop and 4-hop) questions. See Sec. A.3.2 for more details.

4.5 Error Analysis

We have done detailed error analysis for RetinaQA on GrailQAbility. Here, we present a brief summary of it. For this, we used the the best version RetinaQA, which is RetinaQA+T with A+U

Model	Overall		IID		Compositional		Zero-Shot	
	F1	EM	F1	EM	F1	EM	F1	EM
RnG-KBQA	76.80	71.40	89.01	86.70	68.90	61.70	74.70	68.80
TIARA	81.90	75.30	**91.20**	88.40	74.80	66.40	80.70	73.30
Pangu	82.16	75.90	86.38	81.73	76.12	68.82	**82.82**	**76.29**
RetinaQA'	**83.33**	**77.84**	**91.22**	**88.58**	**77.49**	**70.48**	82.32	76.20

Table 6: Performance of different models on GrailQA (validation set) for IID, compositional, and zero-shot test scenarios. RetinaQA' denotes RetinaQA with EGC

Model	F1
TIARA	**75.80**
Pangu	75.00
RetinaQA'	75.60

Table 7: Performance of different models on WebQSP (test set) containing only IID answerable questions. We use the WebQSP evaluation script that only reports F1. RetinaQA' = RetinaQA + EGC

training. We found three main error categories: (1) *thresholding error*, where, due to thresholding, RetinaQA incorrectly predicts NK for a question with a valid logical form; (2) *reranking error*, where the discriminator makes a mistake in scoring, though the candidates contain the correct logical form, and (3) *recall error*, where the correct logical form is not in the set of discriminator candidates. This may be due to errors in entity linking, logical form retrieval or logical form construction.

On the subset of answerable questions, thresholding and reranking errors occur for around 38%, and 30% of the total errors respectively. Improving discriminator calibration can potentially reduce these errors. The most frequent error is recall error (70%). Among these, entity linking errors occur 80% of the time. This clearly suggests that improving entity linkers can significantly improve the overall performance of KBQA models. Unsurprisingly, the majority of the errors across categories occur for the zero-shot generalization questions. Detailed statistics are in Table 8 in the Appendix.

For unanswerable questions, we first look at those with missing data elements (Table 9). We find that the vast majority of errors (around 90%) are recall errors, out of which about 72% are attributed to the entity linker. Next, thresholding error accounts for 45% of the total errors. Reranking errors only occur in 5% of total error. This suggests that while the ranking of the logical forms according to discriminator assigned scores is correct, the absolute scores have errors.

Finally, we look at the subset of unanswerable questions with missing schema elements. Since for these the gold logical form is NK, thresholding error is the only source of error. This occurs for only 14% of the questions (under the same category), out of which 90% of errors occur for zero-shot generalization. This suggests that RetinaQA broadly performs well in this setting but the scores assigned to logical forms with unseen schema elements have occasional errors.

5 Conclusions

We have presented RetinaQA, the first specialized KBQA model that shows robust performance for both answerable and unanswerable questions. To address this challenging task, RetinaQA identifies candidate logical forms using data-traversal-based retrieval, as well as schema-based generation via sketch-filling that bridges over data gaps that break traversal. RetinaQA also discriminate between fully formed candidate logical forms at the final stage instead of generating these. This enables it to better differentiate between valid and invalid logical forms. In doing so, RetinaQA unifies key aspects of different existing KBQA models that assume answerability in IID and transfer settings.

Unlike superficial adaptations of existing SoTA models for unanswerability, we show that RetinaQA demonstrates stable performance across adaptation strategies, question categories (answerable and different categories of unanswerable questions) and different generalization settings. We also show that RetinaQA performs significantly better for unanswerable questions and almost at par for answerable ones. RetinaQA also performs well with only answerable training, which is a likely real-world scenario. RetinaQA also retains this stability and performance for answerable-only KBQA benchmarks, achieving a new state-of-the-art performance on the answerable-only GrailQA dataset. We release our code-base [4] for further research.

[4] https://github.com/dair-iitd/RetinaQA

Limitations

A sketch, while free of references to the KB, still specifies the length of the path to be traversed in the KB. The subsequent grounding step is limited by this and cannot adapt the path length after retrieving schema elements from the KB. RetinaQA inherits this limitation from existing sketch generation approaches (Cao et al., 2022; Ravishankar et al., 2022). We hope to improve this in future work.

For unanswerable questions without valid logical forms for the given KB, RetinaQA only outputs l =NK. However, this does not explain the gap in the schema, which, if bridged, would have make this question answerable. The situation is similar for unanswerable questions with valid logical forms but missing data elements. This is also an important area of future work.

Risks

Our work does not have any obvious risks.

Acknowledgements

Prayushi is supported by a grant from Reliance Foundation. Mausam is supported by a contract with TCS, grants from IBM, Wipro, Verisk, and the Jai Gupta chair fellowship by IIT Delhi. We thank the IIT Delhi HPC facility for its computational resources.

References

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

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

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

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949, Toronto, Canada. Association for Computational Linguistics.

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

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang,

Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Mayur Patidar, Prayushi Faldu, Avinash Singh, Lovekesh Vig, Indrajit Bhattacharya, and Mausam . 2023. Do I have the knowledge to answer? investigating answerability of knowledge base questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10341–10357, Toronto, Canada. Association for Computational Linguistics.

Mayur Patidar, Riya Sawhney, Avinash Kumar Singh, Mausam Biswajit Chatterjee, and Indrajit Bhattacharya. 2024. Few-shot transfer learning for knowledge base question answering: Fusing supervised models with in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

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. *J. Mach. Learn. Res.*, 21(1).

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Srinivas Ravishankar, Dung Thai, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Tahira Naseem, Pavan Kapanipathi, Gaetano Rossiello, and Achille Fokoue. 2022. A two-stage approach towards generalization in knowledge base question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

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

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

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

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

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

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

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

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

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

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. Rankt5: Fine-tuning t5 for text ranking with ranking losses.

A Appendix

A.1 Entity Linker

We use an off-the-shelf entity linker (Ye et al., 2022) previously used in the KBQA literature (Shu et al., 2022; Gu et al., 2023), which uses a standard 3-staged pipeline - Mention Detection, Candidate Generation, and Entity Disambiguation. Mention Detector first identifies span of text from question which corresponds to name of an entity. For each mention a set of candidates entities are generated using alias mapping of FACC1 (Gabrilovich et al., 2013). Final stage is a neural disambiguator which rank candidates given the question and context of entities.

Components	Overall	IID	Compositional	Zero-shot
#questions	6808	3386	981	2441
total_errors	1691	445	347	899
thresholding_error	637	161	113	363
reranking_error	508	49	134	325
recall_error	1183	396	213	574
entity_linking_error	949	343	136	470
schema_retriever_error	460	61	77	322
sketch_parser_error	420	43	154	22

Table 8: Component wise errors of RetinaQA + T (A+U) for answerable questions. Note that reranking errors and recall errors are non-intersecting, while thresholding errors are a subset of total errors i.e. union of reranking and recall errors. Also, recall error is the union of entity_linking_error, schema_retriever_error, and sketch_parser_error while individually these are intersecting.

Components	Overall	IID	Zero-shot
#questions	1196	530	666
total_errors	287	127	160
thresholding_error	131	59	72
reranking_error	16	5	11
recall_error	271	122	149
entity_linking_error	195	77	118
schema_retriever_error	56	29	27
sketch_parser_error	49	27	22

Table 9: Component wise errors of RetinaQA+ T (A+U) for data element missing unanswerable questions. Names have the same meanings as in Table 8

A.2 Implementation Details

To perform experiments for GrailQAbility, we first update the original Freebase KG using codebase[5]. To test baselines for GrailQAbility, we use the existing codebases[6] [7] [8] and make changes in code to adapt for answer-ability detection. All of the baselines assumes answerability and employs Execution Guide Check i.e. if a logical form returns an empty answer upon execution then they select next best logical form. We have removed this constraint while performing experiments for GrailQAbility. Also for A+U training we have made code changes so that models can be trained to predict logical form as NK unanswerable questions. We implement our model using Pytorch (Paszke et al., 2019) and Hugging Face[9]. All the experiments of RetinaQA are performed using an NVIDIA A100 GPU with 80 GB RAM. Above mentioned configurations are the maximum ones, since we have different components and all do not require same compute configurations. For Sketch Generation we fine tune Seq2Seq t5-base model for 10 epochs (fixed). We use learning rate of 3e-5 and batch size of 8. We use beam search during decoding with $beamsize = 10$. We also check syntactic correctness while selecting top ranked sketch. For WebQSP sketch generator is trained for 15 epochs with batch size of 2 Training time for sketch parser is around 3 hours. LF Integrator is a parameter free module and does not require any training. Since, LF Integrator converts logical forms into query-graphs and validates type-level constraints, it is a costly operation. So we employ parallel processing(with cache) for this stage i.e. we use 4-6 CPUs (each with 2 cores) to create pool of valid logical forms. It takes around 5 hours to generate candidates for all train, dev and test data. Finally we train Discriminator which fine-tune t5-base Seq2Seq model. We train Discriminator with learning rate 1e-4 and batch size 4 for 10 epochs. For discriminator training we use AdmaW (Loshchilov and Hutter, 2019) optimizer and linear scheduler with warm up ratio of 0.01. We use 64 negative samples per question for contrastive training. Generally discriminator model converges in 2 epochs of training so we use patience of 2 i.e. if best model does not change for consequent 2 epochs then we assume model has converged and will stop training. It takes around 7-8 hours to train a discriminator. We train WebQSP for 15 epochs with patience equal to 10 and 32 negative samples. Inference time for discriminator is few minutes.

For A+U training components like Entity Linker, Schema Retriever, LF Retriever are trained only on question where logical form is known. While training for questions with l ="NK" is performed only at last step.

All the results presented for single run (however the reproducibility of results is already verified).

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

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

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

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

[9] https://huggingface.co/

Model	Overall		IID		Compositional		Zero-Shot	
	F1	EM	F1	EM	F1	EM	F1	EM
RetinaQA'	83.33	77.84	91.22	88.58	77.49	70.48	82.32	76.2
RetinaQA' - EGC	80.62	75.68	89.81	87.7	74.78	68.1	79.03	73.58
RetinaQA' - LFI	78.65	73.1	88.1	84.81	75	67.31	76.04	70.4
RetinaQA' - LFR	71.8	68.33	87.33	85.56	69	63.47	66.19	62.83
RetinaQA' - (SG ∪ SR)	73.2	66.78	77.06	72.13	60.63	54.43	76.73	69.56
RetinaQA' - LFI - EGC	63.29	59.99	79.84	77.84	59.33	54.03	57.73	54.68

Table 10: Ablation experiment on GrailQA dev set. EGC refers to Execution Guided Check and LFI refers to Logical Form Integrator, RetinaQA' = RetinaQA + EGC

Model	Overall	IID	Compositional	Zero-shot
RetinaQA	82.62	88.3	78.29	76.49
RetinaQA- LFR	74.24	85.91	67.38	60.79
RetinaQA- (SG ∪ SR)	71.94	74.22	65.24	71.49

Table 11: Ablation experiment of Logical Form Recall(%) on GrailQAbility test set for Answerable questions. LFR refers to Logical Form Retriever, SG refers to Sketch Generation and SR refers to Schema Retriever.

Model	Overall	IID	Zero-shot
RetinaQA	77.34	76.98	77.63
RetinaQA- LFR	77.17	76.79	77.48
RetinaQA- (SG ∪ SR)	12.29	10	14.11

Table 12: Ablation experiment of Logical Form Coverage(%) on GrailQAbility test set (Unanswerable questions - Data Element Missing). LFR refers to Logical Form Retriever, SG refers to Sketch Generation and SR refers to Schema Retriever.

#relation	1	2	3	4
Pangu	82.8	63.5	24.7	0.0
TIARA	81.2	64.7	**29.3**	**50.0**
RetinaQA	**83.7**	**68.5**	26.9	**50.0**
RetinaQA - LFR	76.3	50.9	25.1	50.0
RetinaQA - (SP ∪ SR)	72.9	56.9	12.0	0.0

Table 13: Performance for different types of questions on the GrailQA validation set in terms of EM. #relation denotes the number of relations in the s-expression.

We release our code-base[10] for the community.

A.3 In Depth Analysis

A.3.1 Trade-off Analysis

Sec 4.4 describes how individual components strengthens performance for different types of answerabilties and unanswerabilties. This section discusses an important trade-off i.e. **Traversal dependent Retrieval Vs Traversal independent Retrieval** : Traversal based Retrieval methods perform step by step enumeration over KB to retrieve next possible set of candidates(which is retrieval at data level). While Traversal independent Retrieval based method generate candidates based on semantic similarity with the question(which is at schema level). So for Data Element Missing unanswerability where data paths are missing, Traversal based methods will never find correct path during enumeration and hence will not be able to reach to a correct logical form. While Traversal independent method can generate correct logical form. Hence Traversal independent methods performs well for data element missing.

At the same time the search space for Traversal independent methods is much larger as it lacks KB grounding information. So for zero-shot generalisation where schema elements are unseen Traversal dependent tends to get confused between similar schema elements.

A.3.2 Complexity Analysis

Tab. 13 records the performance of RetinaQA for queries of different complexities represented by number of relations in s-expression (or number of hops in answer path). We can see that for 1-hop and 2-hop questions RetinaQA is better than both baselines, while for 3-hop questions RetinaQA is not the best but is better than Pangu. Further by comparing ablations i.e. without Logical Form Retriever and without (Sketch Generation and Schema Retriever) we can see that Sketch Generation and Schema Retriever contribute more to the performance of 3-hop and 4-hop questions.

A.4 GrailQAbility - Dataset Creation

We summarise the dataset creation algorithm of GrailQAbility (Patidar et al., 2023) here. In a nutshell, the authors start with a standard KBQA

[10] https://anonymous.4open.science/r/RETINAQA-122B

dataset containing only answerable questions for a given KB. Then they introduce unanswerability in steps, by deleting schema elements (entity types and relations) and data elements (entities and facts) from the given KB. They mark questions that become unanswerable as a result of each deletion with appropriate unanswerability labels. So starting from a larger set of all answerable questions, the authors create two subsets of data - one set of answerable questions and another set of unanswerable questions (which are unanswerable due to missing structures in graph/KB).